Predict Bike Sharing Demand with EDA Template

import libraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import sys
from scipy.special import boxcox

from sklearn import preprocessing, metrics, feature_selection, model_selection # Import
from sklearn.linear_model import LinearRegression # Import the Lin
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
warnings.filterwarnings("ignore")

import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)

import plotly.express as px
import datetime as dt
sns.set()
%matplotlib inline
```

Exploratory Data Analysis

load Data

In [2]:

```
train = pd.read_csv("train.csv" ,parse_dates= ['datetime'])
test = pd.read_csv("test.csv" ,parse_dates= ['datetime'])
```

Data preparation for EDA

In [3]:

```
train.head()
```

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regist
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regist
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [4]:

```
test.head()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

In [5]:

```
# Train Data
train['dayofweek'] = train.datetime.dt.dayofweek
train['hour'] =  train.datetime.dt.hour
train['day'] =  train.datetime.dt.day
train['month'] =  train.datetime.dt.month
train['year'] =  train.datetime.dt.year
```

In [6]:

```
# test Data
test['dayofweek'] = test.datetime.dt.dayofweek
test['hour'] =  test.datetime.dt.hour
test['day'] =  test.datetime.dt.day
test['month'] =  test.datetime.dt.month
test['year'] =  test.datetime.dt.year
```

In [7]:

```
train.describe().T.style.background_gradient(cmap='PuBuGn')
```

Out[7]:

	count	mean	std	min	25%	50%	75%
season	10886.000000	2.506614	1.116174	1.000000	2.000000	3.000000	4.000000
holiday	10886.000000	0.028569	0.166599	0.000000	0.000000	0.000000	0.000000

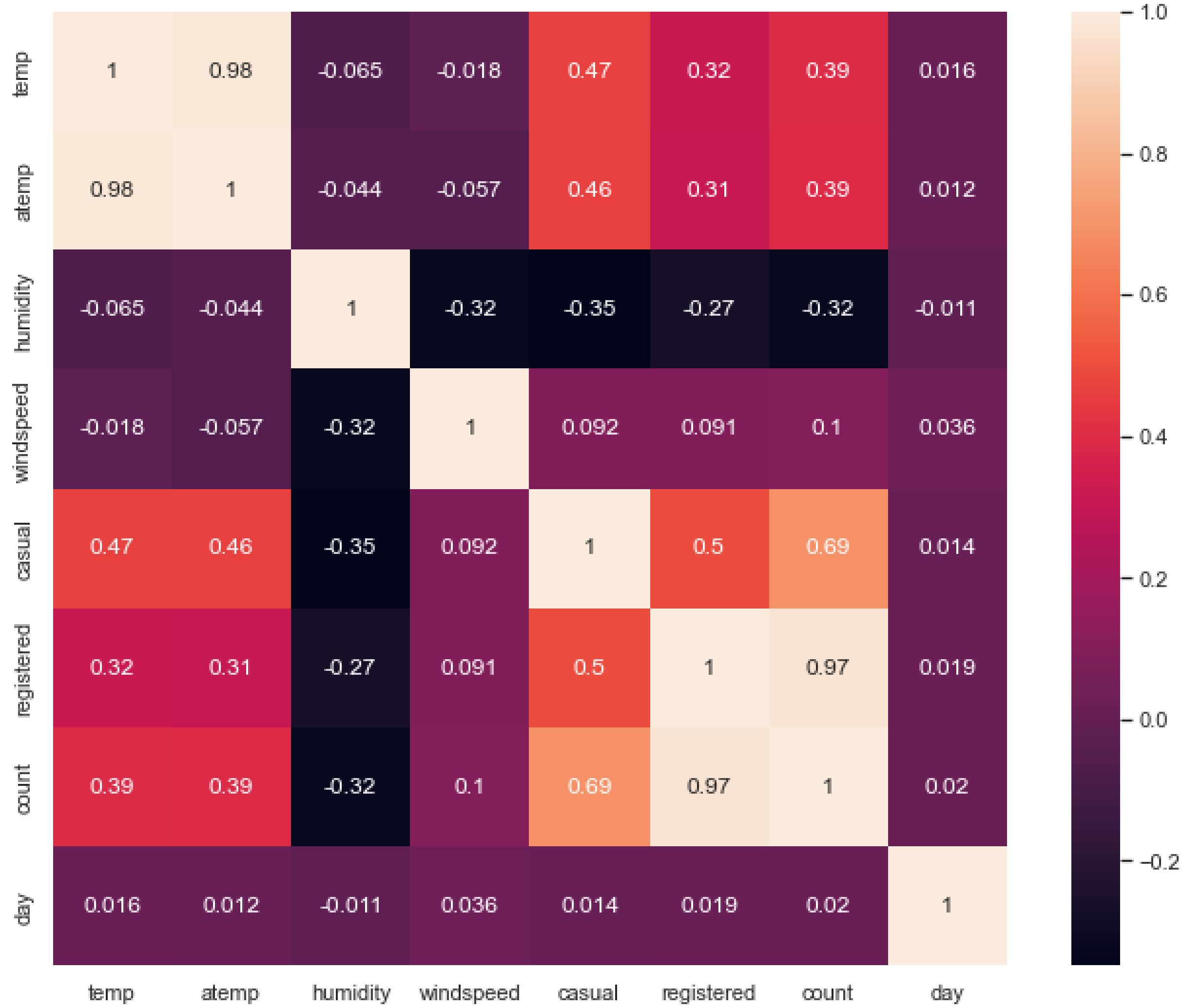
	count	mean	std	min	25%	50%	75%
workingday	10886.000000	0.680875	0.466159	0.000000	0.000000	1.000000	1.000000
weather	10886.000000	1.418427	0.633839	1.000000	1.000000	1.000000	2.000000
temp	10886.000000	20.230860	7.791590	0.820000	13.940000	20.500000	26.240000
atemp	10886.000000	23.655084	8.474601	0.760000	16.665000	24.240000	31.060000
humidity	10886.000000	61.886460	19.245033	0.000000	47.000000	62.000000	77.000000
windspeed	10886.000000	12.799395	8.164537	0.000000	7.001500	12.998000	16.997900
casual	10886.000000	36.021955	49.960477	0.000000	4.000000	17.000000	49.000000
registered	10886.000000	155.552177	151.039033	0.000000	36.000000	118.000000	222.000000
count	10886.000000	191.574132	181.144454	1.000000	42.000000	145.000000	284.000000
dayofweek	10886.000000	3.013963	2.004585	0.000000	1.000000	3.000000	5.000000
hour	10886.000000	11.541613	6.915838	0.000000	6.000000	12.000000	18.000000
day	10886.000000	9.992559	5.476608	1.000000	5.000000	10.000000	15.000000
month	10886.000000	6.521495	3.444373	1.000000	4.000000	7.000000	10.000000
year	10886.000000	2011.501929	0.500019	2011.000000	2011.000000	2012.000000	2012.000000

In [217…

```
# Plot the correlation matrix using a heatmap

corrmat = train.corr()
fig, ax = plt.subplots(figsize=(12, 9))

sns.heatmap(corrmat, square=True, annot=True);
```



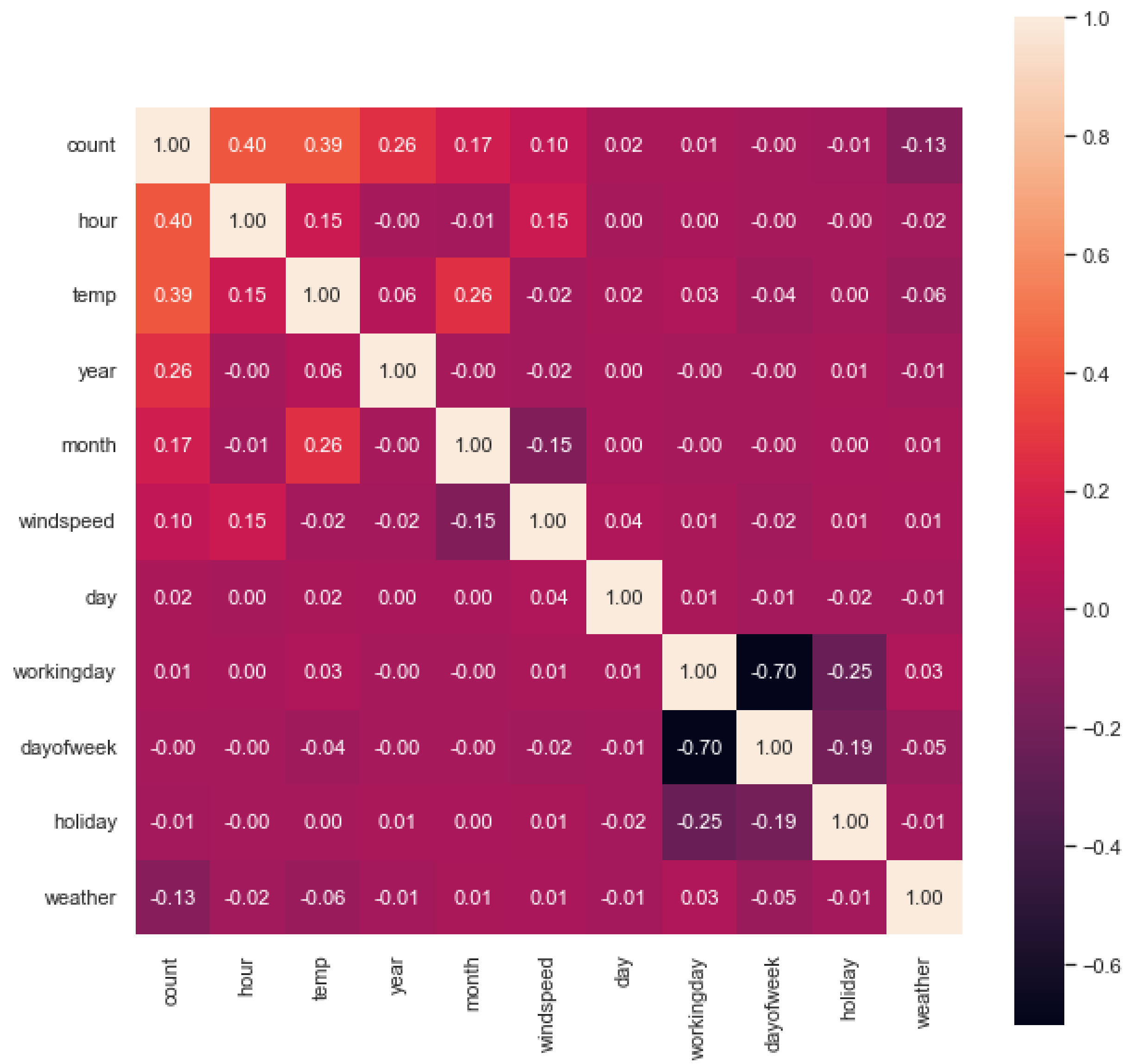
From the heatmap, 'registered’ and 'casual' are highly correlated to ‘count’, 'season’ is highly correlated to ‘month’, 'atemp’ is highly correlated to ‘temp’. In addition, we know that casual + registered = count. Hence we can drop columns ‘registered’, ‘casual’, "atemp" and ‘season’.

In [9]:

```
corrmat.drop(["season"],axis=0,inplace=True)
corrmat.drop(["season"],axis=1,inplace=True)
corrmat.drop(["atemp"],axis=0,inplace=True)
corrmat.drop(["atemp"],axis=1,inplace=True)
corrmat.drop(["casual"],axis=0,inplace=True)
corrmat.drop(["casual"],axis=1,inplace=True)
corrmat.drop(["registered"],axis=0,inplace=True)
corrmat.drop(["registered"],axis=1,inplace=True)
```

In [10]:

```
# Let's now plot a "zoomed" correlation matrix, with respect to our response variable
fig, ax = plt.subplots(figsize=(10,10))
k = 11 #number of variables for heatmap
cols = corrmat.nlargest(k, 'count').loc[:,'count'].index
cm = train.loc[:,cols].corr()
sns.set(font_scale=1)
sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 11},
plt.show()
```



From the above matrix, we noted that 'day', 'workingday', 'dayofweek' and 'holiday' seem to have little or no correlation to 'count'. Nevertheless, lets visualize their relationship with some plots.

In [11]:
```
train.sample(10)
```

Out[11]:

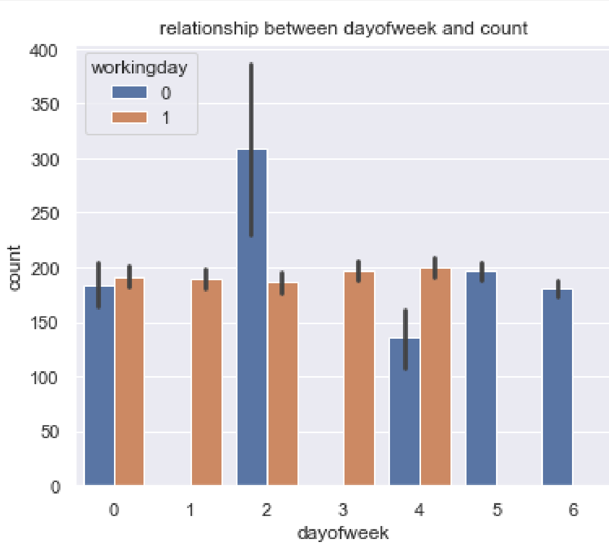
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
9070	2012-09-01 07:00:00	3	0	0	2	30.34	34.090	58	8.9981	8	
102	2011-01-05 11:00:00	1	0	1	1	10.66	11.365	33	22.0028	12	
8021	2012-06-14 14:00:00	2	0	1	1	28.70	32.575	54	19.0012	84	

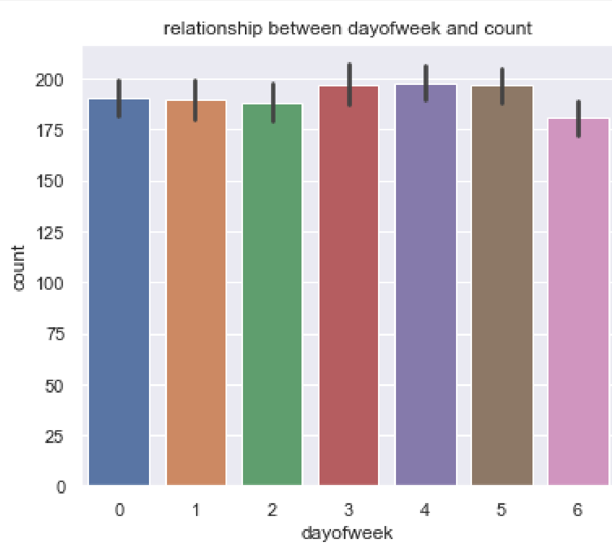
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
8910	2012-08-13 15:00:00	3	0	1	1	33.62	36.365	34	15.0013	80	
9456	2012-09-17 09:00:00	3	0	1	1	23.78	27.275	68	8.9981	34	
7438	2012-05-09 07:00:00	2	0	1	2	22.96	26.515	94	12.9980	17	
8507	2012-07-15 20:00:00	3	0	0	1	29.52	34.850	79	16.9979	95	
3581	2011-08-19 03:00:00	3	0	1	1	26.24	29.545	78	0.0000	2	
2451	2011-06-10 01:00:00	2	0	1	3	28.70	32.575	65	15.0013	5	
518	2011-02-04 18:00:00	1	0	1	2	9.84	12.880	60	7.0015	3	

In [12]:

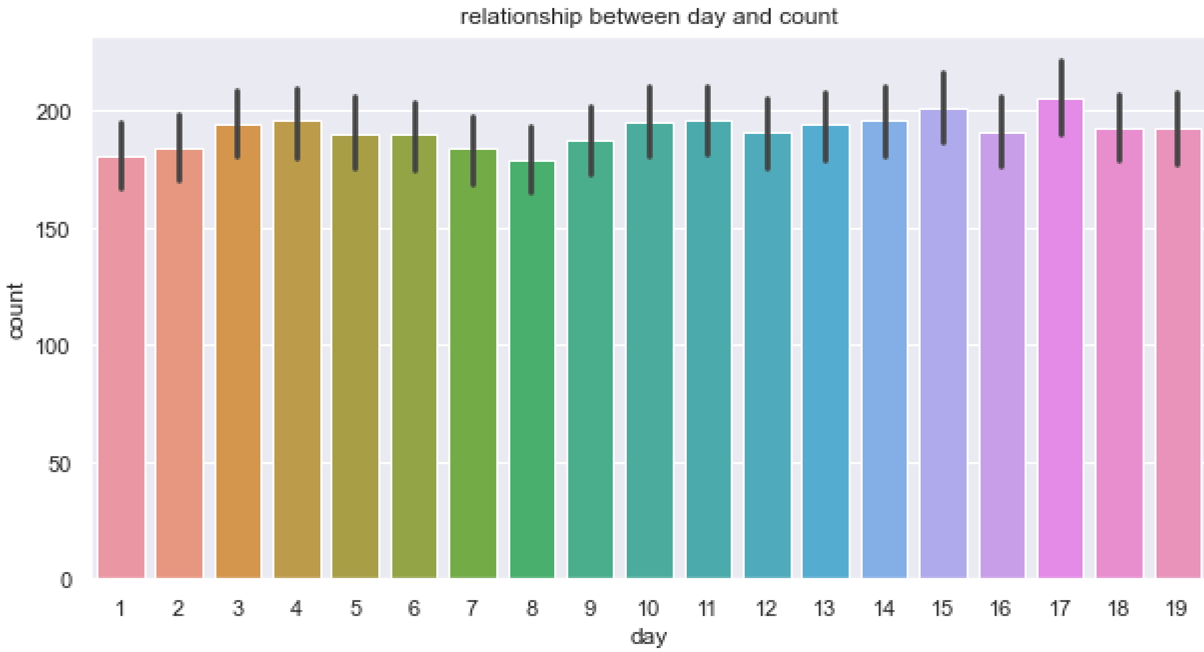
```
#base_color = sns.color_palette()[9]
plt.figure(figsize= (13, 5))
plt.subplot(1, 2, 1)
sns.barplot(data=train, x="dayofweek", y="count", hue='workingday')
plt.title("relationship between dayofweek and count")
plt.subplot(1, 2, 2)
sns.barplot(data=train, x="dayofweek", y="count")
plt.title("relationship between dayofweek and count")

plt.show()
```



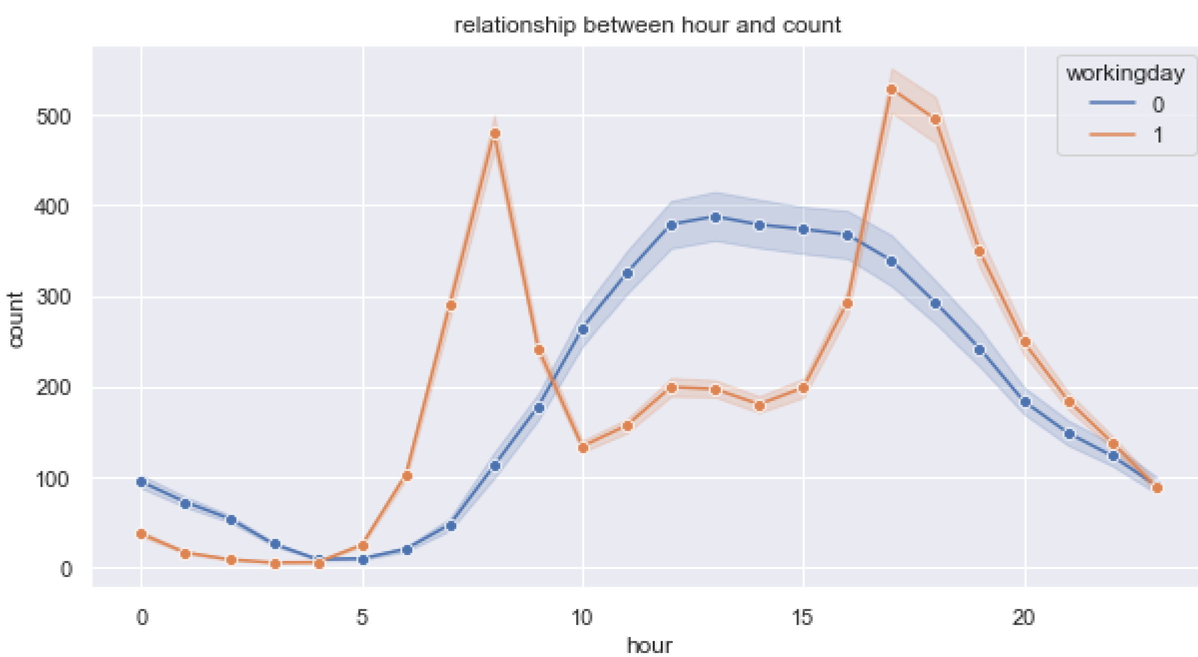


In [13]:
```
plt.figure(figsize= (10, 5))
sns.barplot(data=train, x="day", y="count")
plt.title("relationship between day and count")
plt.show()
```



Based on the above plots, there is no obvious relationship between 'count' and 'dayofweek' and between 'count' and 'day' as per the coefficient coefficient. Whether 'dayofweek' is a working or non-working day also doesn't seem to matter.

In [14]:
```
plt.figure(figsize= (10, 5))
sns.lineplot(data=train, x="hour", y="count", hue='workingday', marker='o', markers=Tru
plt.title("relationship between hour and count")
plt.show()
```



What is obvious is that 'count' is dependent on 'hour' of the day, and whether it is a working or non-working day. When it is a working day, the morning and evening rush hours tend to have a higher count. However, when it is a non-working day, the count is higher from ~10am to ~8pm.

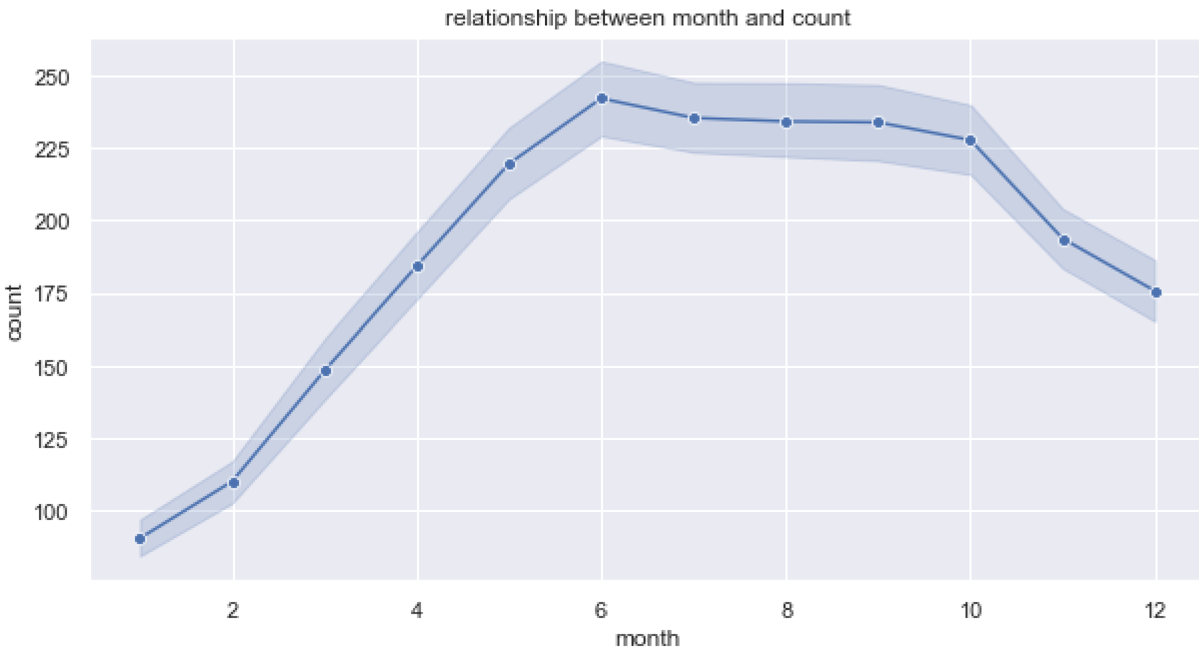
In [ ]:

```

```

In [15]:

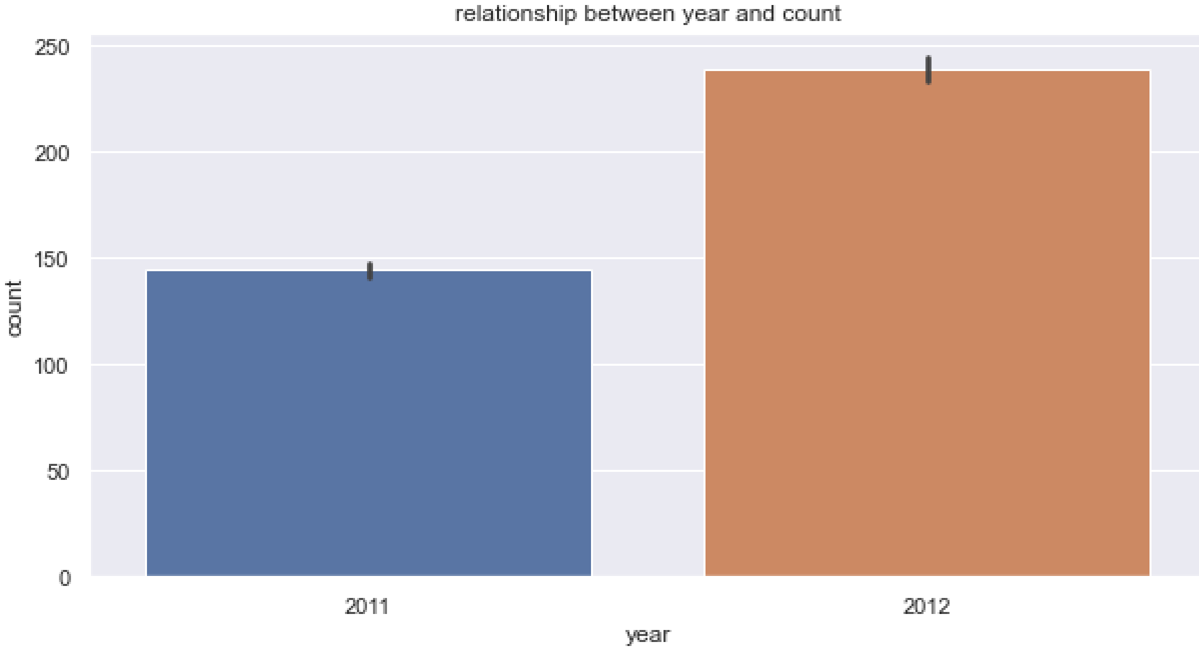
```
plt.figure(figsize= (10, 5))
sns.lineplot(data=train, x="month", y="count", marker='o', markers=True, dashes=False)
plt.title("relationship between month and count")
plt.show()
```



'count' is also dependent on the 'month' of the year, where 'count' is higher from May to Oct (presumably due to warmer spring/autumn months)

In [16]:

```
plt.figure(figsize= (10, 5))
sns.barplot(data=train, x="year", y="count")
plt.title("relationship between year and count")
plt.show()
```



the most year in which users rent bicycles is 2012 .

In [ ]:

In [17]:

```
train.var()
```

Out[17]:

```
season            1.245845
holiday           0.027755
workingday        0.217304
weather           0.401751
temp             60.708872
atemp            71.818856
humidity        370.371306
windspeed        66.659670
casual         2496.049219
registered    22812.789514
count         32813.313153
dayofweek         4.018363
hour             47.828815
day              29.993238
month            11.863709
year              0.250019
dtype: float64
```

In [18]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
```

```
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
 12  dayofweek   10886 non-null  int64
 13  hour        10886 non-null  int64
 14  day         10886 non-null  int64
 15  month       10886 non-null  int64
 16  year        10886 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(13)
memory usage: 1.4 MB
```

In [19]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6493 entries, 0 to 6492
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    6493 non-null   datetime64[ns]
 1   season      6493 non-null   int64
 2   holiday     6493 non-null   int64
 3   workingday  6493 non-null   int64
 4   weather     6493 non-null   int64
 5   temp        6493 non-null   float64
 6   atemp       6493 non-null   float64
 7   humidity    6493 non-null   int64
 8   windspeed   6493 non-null   float64
 9   dayofweek   6493 non-null   int64
 10  hour        6493 non-null   int64
 11  day         6493 non-null   int64
 12  month       6493 non-null   int64
 13  year        6493 non-null   int64
dtypes: datetime64[ns](1), float64(3), int64(10)
memory usage: 710.3 KB
```

Feature Engineering

- add new features

In [20]:
```
train['weather'] = train['weather'].map({1: 'clear',2:  'few clouds', 3: 'partly cloudy
train['season'] = train['season'].map({1 : 'spring', 2 : 'summer', 3 : 'fall', 4 : 'win
#train['holiday'] = train['holiday'].map({0 : 'no', 1 : 'yes'})
#train['workingday'] = train['workingday'].map({0 : 'no', 1 : 'yes'})
test['weather'] = test['weather'].map({1: 'clear',2:  'few clouds', 3: 'partly cloudy',
test['season'] = test['season'].map({1 : 'spring', 2 : 'summer', 3 : 'fall', 4 : 'winte
#test['holiday'] = test['holiday'].map({0 : 'no', 1 : 'yes'})
#test['workingday'] = test['workingday'].map({0 : 'no', 1 : 'yes'})
```

In [21]:
```
train['holiday'].value_counts()
```

Out[21]:
```
0    10575
```

```
1      311
Name: holiday, dtype: int64
```

In [22]:
```
train['workingday'].value_counts()
```

Out[22]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

In [23]:
```
train.head()
```

Out[23]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regist
0	2011-01-01 00:00:00	spring	0	0	clear	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	spring	0	0	clear	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	spring	0	0	clear	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	spring	0	0	clear	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	spring	0	0	clear	9.84	14.395	75	0.0	0	

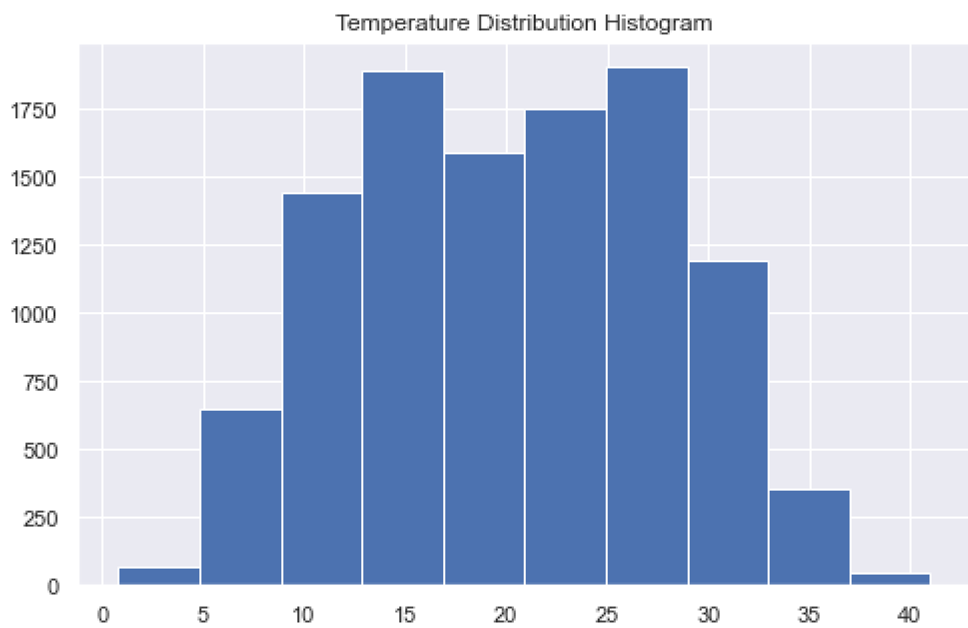
In [24]:
```
test.head()
```

Out[24]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	dayofweek	I
0	2011-01-20 00:00:00	spring	0	1	clear	10.66	11.365	56	26.0027	3	
1	2011-01-20 01:00:00	spring	0	1	clear	10.66	13.635	56	0.0000	3	
2	2011-01-20 02:00:00	spring	0	1	clear	10.66	13.635	56	0.0000	3	
3	2011-01-20 03:00:00	spring	0	1	clear	10.66	12.880	56	11.0014	3	
4	2011-01-20 04:00:00	spring	0	1	clear	10.66	12.880	56	11.0014	3	

In [25]:
```
# Feature that categorizes hours
bins = [-np.inf, 0, 11, 12, 17, np.inf]
labels = ['MIDNIGHT', 'MORNING', 'MIDDAY', 'AFTERNOON', 'EVENING']
train['times of the day'] = pd.cut(train['hour'], bins= bins, labels= labels)
train['times of the day'] = train['times of the day'].str.lower()
```

In [26]:
```
train['temp'].hist(figsize= (8, 5));
plt.title('Temperature Distribution Histogram')
plt.show()
```

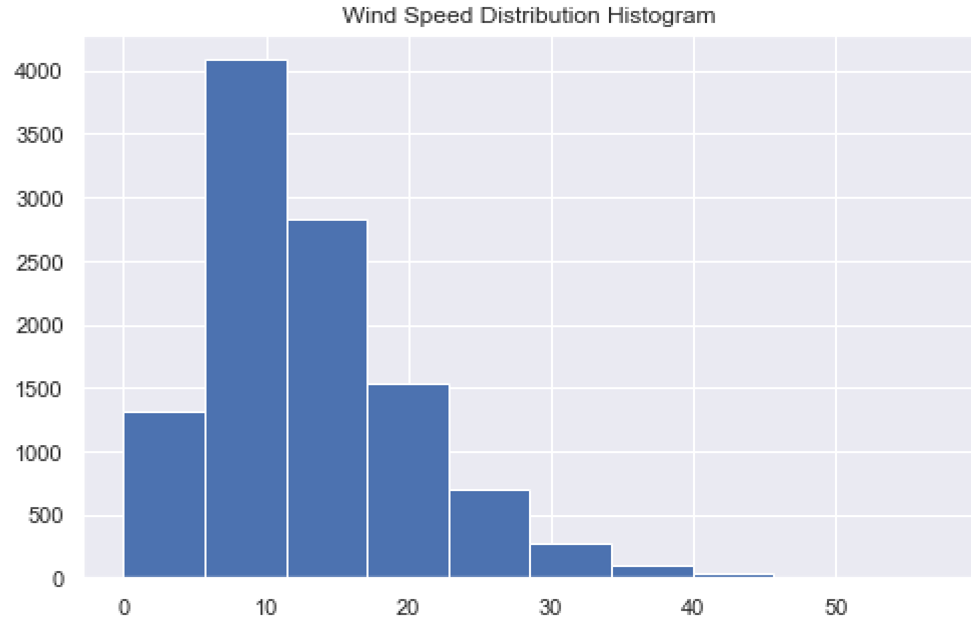


In [27]:
```
#Feature that categorizes hot/cold/mild temps from temp

bins = [-np.inf, 20, 30, np.inf]
labels = ['cold', 'mild', 'hot']
train['temp of the day'] = pd.cut(train['temp'], bins= bins, labels= labels)
train['temp of the day'].value_counts()
```

Out[27]:
```
cold    5308
mild    4334
hot     1244
Name: temp of the day, dtype: int64
```

In [28]:
```
train['windspeed'].hist(figsize= (8, 5));
plt.title('Wind Speed Distribution Histogram')
plt.show()
```



In [29]:
```
#Feature that categories Calm, Moderate, Strong.
bins = [-np.inf, 20, 38, np.inf]
labels = ['calm', 'moderate', 'strong']
train['windspeed of the day'] = pd.cut(train['windspeed'], bins= bins, labels= labels)
train['windspeed of the day'].value_counts()
```

Out[29]:
```
calm        9391
moderate    1428
strong        67
Name: windspeed of the day, dtype: int64
```

In [30]:
```
# Feature that categorizes hours
bins = [-np.inf, 0, 11, 12, 17, np.inf]
labels = ['MIDNIGHT', 'MORNING', 'MIDDAY', 'AFTERNOON', 'EVENING']
test['times of the day'] = pd.cut(train['hour'], bins= bins, labels= labels)
test['times of the day'] = test['times of the day'].str.lower()
```

In [31]:
```
#Feature that categorizes hot/cold/mild temps from temp

bins = [-np.inf, 20, 30, np.inf]
labels = ['cold', 'mild', 'hot']
test['temp of the day'] = pd.cut(test['temp'], bins= bins, labels= labels)
test['temp of the day'].value_counts()
```

Out[31]:
```
cold    3021
mild    2601
hot      871
Name: temp of the day, dtype: int64
```

In [32]:
```
#Feature that categories Calm, Moderate, Strong.
bins = [-np.inf, 20, 38, np.inf]
labels = ['calm', 'moderate', 'strong']
test['windspeed of the day'] = pd.cut(test['windspeed'], bins= bins, labels= labels)
test['windspeed of the day'].value_counts()
```

```
calm        5585
```

Out[32]:
```
moderate     868
strong        40
Name: windspeed of the day, dtype: int64
```

Make category types for these so models know they are not just numbers

In [33]:
```
train.columns
```

Out[33]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'dayofweek', 'hour', 'day', 'month', 'year', 'times of the day',
       'temp of the day', 'windspeed of the day'],
      dtype='object')
```

In [34]:
```
train.head()
```

Out[34]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regist
0	2011-01-01 00:00:00	spring	0	0	clear	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	spring	0	0	clear	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	spring	0	0	clear	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	spring	0	0	clear	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	spring	0	0	clear	9.84	14.395	75	0.0	0	

In [35]:
```
train["season"] =train["season"].astype("category")
train["weather"] = train["weather"].astype("category")
train["holiday"] = train["holiday"].astype("category")
train["workingday"] = train["workingday"].astype("category")
train["times of the day"] = train["times of the day"].astype("category")
train["temp of the day"] = train["temp of the day"].astype("category")
train["windspeed of the day"] = train["windspeed of the day"].astype("category")
train['month'] = train['month'].astype("category")
train['year'] = train['year'].astype("category")
train['hour'] = train['hour'].astype("str")
train['dayofweek'] = train['dayofweek'].astype("str")
test["season"] = test["season"].astype("category")
test["weather"] = test["weather"].astype("category")
test["holiday"] = test["holiday"].astype("category")
```

```python
test["workingday"] = test["workingday"].astype("category")
test["times of the day"] = test["times of the day"].astype("category")
test["temp of the day"] = test["temp of the day"].astype("category")
test["windspeed of the day"] = test["windspeed of the day"].astype("category")
test['month'] = test['month'].astype("category")
test['year'] = test['year'].astype("category")
test['hour'] = test['hour'].astype("str")
test['dayofweek'] = test['dayofweek'].astype("str")
```

In [36]:

```python
# View are new feature
pd.set_option('display.max_columns', 500)
train.head()
```

Out[36]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regis
0	2011-01-01 00:00:00	spring	0	0	clear	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	spring	0	0	clear	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	spring	0	0	clear	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	spring	0	0	clear	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	spring	0	0	clear	9.84	14.395	75	0.0	0	

In [37]:

```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   datetime              10886 non-null  datetime64[ns]
 1   season                10886 non-null  category
 2   holiday               10886 non-null  category
 3   workingday            10886 non-null  category
 4   weather               10886 non-null  category
 5   temp                  10886 non-null  float64
 6   atemp                 10886 non-null  float64
 7   humidity              10886 non-null  int64
 8   windspeed             10886 non-null  float64
 9   casual                10886 non-null  int64
 10  registered            10886 non-null  int64
 11  count                 10886 non-null  int64
```

```
 12  dayofweek             10886 non-null  object
 13  hour                  10886 non-null  object
 14  day                   10886 non-null  int64
 15  month                 10886 non-null  category
 16  year                  10886 non-null  category
 17  times of the day      10886 non-null  category
 18  temp of the day       10886 non-null  category
 19  windspeed of the day  10886 non-null  category
dtypes: category(9), datetime64[ns](1), float64(3), int64(5), object(2)
memory usage: 1.0+ MB
```

In [38]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6493 entries, 0 to 6492
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   datetime              6493 non-null   datetime64[ns]
 1   season                6493 non-null   category
 2   holiday               6493 non-null   category
 3   workingday            6493 non-null   category
 4   weather               6493 non-null   category
 5   temp                  6493 non-null   float64
 6   atemp                 6493 non-null   float64
 7   humidity              6493 non-null   int64
 8   windspeed             6493 non-null   float64
 9   dayofweek             6493 non-null   object
 10  hour                  6493 non-null   object
 11  day                   6493 non-null   int64
 12  month                 6493 non-null   category
 13  year                  6493 non-null   category
 14  times of the day      6493 non-null   category
 15  temp of the day       6493 non-null   category
 16  windspeed of the day  6493 non-null   category
dtypes: category(9), datetime64[ns](1), float64(3), int64(2), object(2)
memory usage: 464.6+ KB
```

In [39]:

```
train.isnull().sum()
```

Out[39]:

```
datetime                0
season                  0
holiday                 0
workingday              0
weather                 0
temp                    0
atemp                   0
humidity                0
windspeed               0
casual                  0
registered              0
count                   0
dayofweek               0
hour                    0
day                     0
month                   0
year                    0
times of the day        0
temp of the day         0
```

```
windspeed of the day    0
dtype: int64
```

In [40]:
```
test.isnull().sum()
```

Out[40]:
```
datetime                0
season                  0
holiday                 0
workingday              0
weather                 0
temp                    0
atemp                   0
humidity                0
windspeed               0
dayofweek               0
hour                    0
day                     0
month                   0
year                    0
times of the day        0
temp of the day         0
windspeed of the day    0
dtype: int64
```

In [41]:
```
plt.figure(figsize= (10, 8))
sns.heatmap(train.corr(), annot = True, cmap = 'PuBuGn')
plt.show()
```

	temp	atemp	humidity	windspeed	casual	registered	count	day
temp	1	0.98	-0.065	-0.018	0.47	0.32	0.39	0.016
atemp	0.98	1	-0.044	-0.057	0.46	0.31	0.39	0.012
humidity	-0.065	-0.044	1	-0.32	-0.35	-0.27	-0.32	-0.011
windspeed	-0.018	-0.057	-0.32	1	0.092	0.091	0.1	0.036
casual	0.47	0.46	-0.35	0.092	1	0.5	0.69	0.014
registered	0.32	0.31	-0.27	0.091	0.5	1	0.97	0.019
count	0.39	0.39	-0.32	0.1	0.69	0.97	1	0.02
day	0.016	0.012	-0.011	0.036	0.014	0.019	0.02	1

In [42]:
```
train.describe().T.style.background_gradient(cmap='PuBuGn')
```

Out[42]:

	count	mean	std	min	25%	50%	75%	ma
temp	10886.000000	20.230860	7.791590	0.820000	13.940000	20.500000	26.240000	41.0000
atemp	10886.000000	23.655084	8.474601	0.760000	16.665000	24.240000	31.060000	45.4550
humidity	10886.000000	61.886460	19.245033	0.000000	47.000000	62.000000	77.000000	100.0000
windspeed	10886.000000	12.799395	8.164537	0.000000	7.001500	12.998000	16.997900	56.9969
casual	10886.000000	36.021955	49.960477	0.000000	4.000000	17.000000	49.000000	367.0000
registered	10886.000000	155.552177	151.039033	0.000000	36.000000	118.000000	222.000000	886.0000
count	10886.000000	191.574132	181.144454	1.000000	42.000000	145.000000	284.000000	977.0000
day	10886.000000	9.992559	5.476608	1.000000	5.000000	10.000000	15.000000	19.0000

In [43]:
```
train.sample(10)
```

Out[43]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
2568	2011-06-14 22:00:00	summer	0	1	clear	23.78	27.275	60	15.0013	29
3861	2011-09-11 21:00:00	fall	0	0	clear	26.24	30.305	69	0.0000	45
10345	2012-11-16 11:00:00	winter	0	1	clear	15.58	19.695	43	15.0013	33
4538	2011-11-02 04:00:00	winter	0	1	clear	12.30	16.665	87	0.0000	0
4258	2011-10-09 11:00:00	winter	0	0	clear	25.42	31.060	53	6.0032	189
3087	2011-07-17 13:00:00	fall	0	0	clear	32.80	37.120	49	19.9995	200
3764	2011-09-07 19:00:00	fall	0	1	few clouds	26.24	28.790	89	0.0000	14
2959	2011-07-12 05:00:00	fall	0	1	clear	28.70	33.335	79	6.0032	4

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
3156	2011-08-01 10:00:00	fall	0	1	clear	35.26	37.880	36	11.0014	27
9145	2012-09-04 10:00:00	fall	0	1	few clouds	29.52	34.850	74	16.9979	42

In [44]:
```
test.sample(10)
```

Out[44]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	dayofwe
6158	2012-11-28 22:00:00	winter	0	1	clear	11.48	13.635	45	11.0014	
2308	2011-09-27 17:00:00	winter	0	1	few clouds	27.06	29.545	89	12.9980	
4736	2012-06-26 04:00:00	fall	0	1	clear	22.14	25.760	45	19.9995	
1587	2011-07-21 03:00:00	fall	0	1	few clouds	30.34	35.605	79	8.9981	
5820	2012-10-25 08:00:00	winter	0	1	few clouds	21.32	25.000	83	11.0014	
568	2011-03-24 14:00:00	summer	0	1	few clouds	12.30	15.150	70	8.9981	
4097	2012-04-22 13:00:00	summer	0	0	partly cloudy	15.58	19.695	82	23.9994	
1897	2011-08-22 01:00:00	fall	0	1	clear	27.88	31.820	79	6.0032	
4000	2012-03-30 12:00:00	summer	0	1	few clouds	15.58	19.695	54	0.0000	
4832	2012-06-30 04:00:00	fall	0	0	few clouds	25.42	27.275	94	0.0000	

In [45]:
```
def describe_cont_feature(feature):
    print('\n*** Results for {} ***'.format(feature))
    print(train.groupby('temp of the day')[feature].describe())

describe_cont_feature('count')
```

```
*** Results for count ***
                  count        mean         std  min    25%    50%    75%  \
temp of the day
cold             5308.0  132.135268  138.794661  1.0   24.0   89.0  194.0
mild             4334.0  223.411398  195.357875  1.0   58.0  181.0  328.0
hot              1244.0  334.274116  181.823864  4.0  198.0  303.0  441.0

                   max
temp of the day
cold             837.0
mild             977.0
hot              897.0
```

In [46]:
```
def describe_cont_feature(feature):
    print('\n*** Results for {} ***'.format(feature))
    print(train.groupby('windspeed of the day')[feature].describe())

describe_cont_feature('count')
```

```
*** Results for count ***
                       count        mean         std  min   25%    50%  \
windspeed of the day
calm                  9391.0  189.033436  181.506649  1.0  38.0  141.0
moderate              1428.0  209.392157  179.130130  1.0  71.0  164.5
strong                  67.0  167.925373  149.851271  1.0  55.0  140.0

                         75%    max
windspeed of the day
calm                  282.00  977.0
moderate              302.25  890.0
strong                219.00  755.0
```

In [47]:
```
def describe_cont_feature(feature):
    print('\n*** Results for {} ***'.format(feature))
    print(train.groupby('times of the day')[feature].describe())

describe_cont_feature('count')
```

```
*** Results for count ***
                   count        mean         std  min    25%    50%     75%  \
times of the day
afternoon         2280.0  308.133333  185.142294  7.0  170.0  271.0  412.25
evening           2736.0  228.518640  173.859618  4.0   99.0  180.0  308.00
midday             456.0  256.508772  143.881880  3.0  157.0  234.5  332.00
midnight           455.0   55.138462   43.620012  2.0   24.0   41.0   74.50
morning           4959.0  124.147812  154.567065  1.0   11.0   56.0  187.50

                    max
times of the day
afternoon         970.0
```

```
evening          977.0
midday           757.0
midnight         283.0
morning          839.0
```

In [48]:

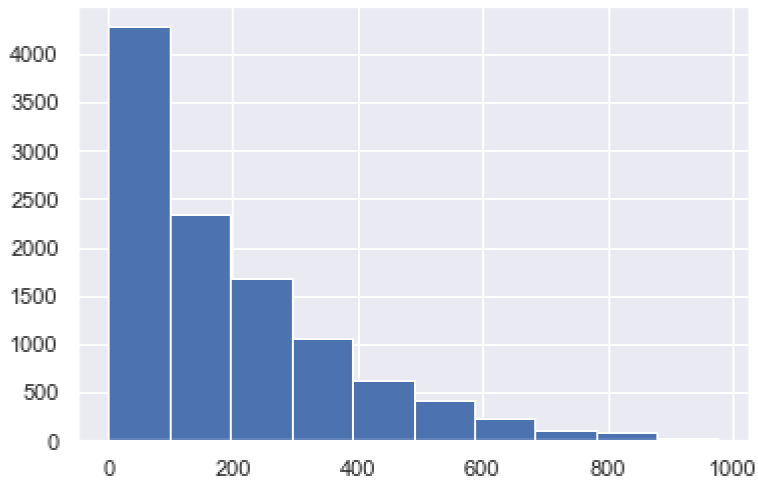
```
train.columns
```

Out[48]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'dayofweek', 'hour', 'day', 'month', 'year', 'times of the day',
       'temp of the day', 'windspeed of the day'],
      dtype='object')
```

In [49]:

```
train['count'].hist()
plt.show()
```



In [50]:

```
#Feature that categories "Low" "medium"  "above average" "high".
bins = [-np.inf, 400,  np.inf]
labels = ["normal number", "abnormal number"]
train['count_category'] = pd.cut(train['count'], bins= bins, labels= labels)
train['count_category'].value_counts()
```

Out[50]:

```
normal number      9443
abnormal number    1443
Name: count_category, dtype: int64
```

In [ ]:

In [51]:

```
pd.crosstab(train['count_category'], train['windspeed of the day']).style.background_gr
```

Out[51]:

windspeed of the day	calm	moderate	strong
count_category			
normal number	8156	1223	64
abnormal number	1235	205	3

In [52]:

```
pd.crosstab(train['count_category'], train['temp of the day']).style.background_gradien
```

Out[52]:

temp of the day	cold	mild	hot
count_category			
normal number	4992	3595	856
abnormal number	316	739	388

In [53]:

```
pd.crosstab(train['count_category'], train['times of the day']).style.background_gradie
```

Out[53]:

times of the day	afternoon	evening	midday	midnight	morning
count_category					
normal number	1675	2318	389	455	4606
abnormal number	605	418	67	0	353

In [54]:

```
def describe_cont_feature(feature):
    print('\n*** Results for {} ***'.format(feature))
    print(train.groupby('count_category')[feature].describe())

for col in train.columns:
    #if (train[col].dtype == 'int64' or train[col].dtype == 'float64 '):
    describe_cont_feature(col)
```

```
*** Results for datetime ***
                count unique                  top freq                first  \
count_category
normal number    9443   9443 2011-01-01 00:00:00    1 2011-01-01 00:00:00
abnormal number  1443   1443 2011-04-11 17:00:00    1 2011-04-11 17:00:00

                               last
count_category
normal number   2012-12-19 23:00:00
abnormal number 2012-12-19 18:00:00

*** Results for season ***
                count unique    top  freq
count_category
normal number    9443      4  spring  2566
abnormal number  1443      4    fall   508

*** Results for holiday ***
                 count  unique  top  freq
count_category
normal number     9443       2    0  9178
abnormal number   1443       2    0  1397

*** Results for workingday ***
                 count  unique  top  freq
count_category
normal number     9443       2    1  6464
abnormal number   1443       2    1   948

*** Results for weather ***
```

```
                count unique    top  freq
count_category
normal number    9443      4  clear  6094
abnormal number  1443      3  clear  1098

*** Results for temp ***
                  count       mean       std   min    25%    50%    75%    max
count_category
normal number    9443.0  19.463603  7.688941  0.82  13.12  18.86  25.42  41.00
abnormal number  1443.0  25.251795  6.486419  4.10  20.50  26.24  30.34  37.72

*** Results for atemp ***
                  count       mean       std   min    25%     50%     75%  \
count_category
normal number    9443.0  22.829653  8.400155  0.76  15.91  22.725  30.305
abnormal number  1443.0  29.056712  6.814310  6.06  24.24  31.060  34.090

                    max
count_category
normal number    45.455
abnormal number  42.425

*** Results for humidity ***
                  count       mean        std   min   25%   50%   75%    max
count_category
normal number    9443.0  63.297045  19.143434   0.0  49.0  64.0  79.0  100.0
abnormal number  1443.0  52.655579  17.258856  16.0  39.0  51.0  65.0  100.0

*** Results for windspeed ***
                  count       mean       std  min     25%      50%      75%  \
count_category
normal number    9443.0  12.632834  8.213963  0.0  7.0015  11.0014  16.9979
abnormal number  1443.0  13.889374  7.748206  0.0  8.9981  12.9980  19.0012

                     max
count_category
normal number    56.9969
abnormal number  43.9989

*** Results for casual ***
                  count        mean        std  min   25%   50%    75%    max
count_category
normal number    9443.0   25.363550  32.171598  0.0   3.0  12.0   36.0  240.0
abnormal number  1443.0  105.770617  80.325962  1.0  39.0  84.0  164.0  367.0

*** Results for registered ***
                  count        mean         std    min    25%    50%    75%  \
count_category
normal number    9443.0  111.189453   91.292051    0.0   27.0   96.0  175.0
abnormal number  1443.0  445.862093  142.848973  173.0  342.0  417.0  532.0

                   max
count_category
normal number    391.0
abnormal number  886.0

*** Results for count ***
                  count        mean         std    min    25%    50%    75%  \
count_category
normal number    9443.0  136.553002  112.433425    1.0   32.0  116.0  219.0
```

```
abnormal number  1443.0  551.632710  123.947170  401.0  453.5  516.0  623.0

                   max
count_category
normal number    400.0
abnormal number  977.0

*** Results for dayofweek ***
                count unique top  freq
count_category
normal number    9443      7   6  1378
abnormal number  1443      7   5   248

*** Results for hour ***
                count unique top freq
count_category
normal number    9443     24  23  456
abnormal number  1443     16  17  272

*** Results for day ***
                  count       mean       std  min  25%   50%   75%   max
count_category
normal number    9443.0   9.985492  5.484366  1.0  5.0  10.0  15.0  19.0
abnormal number  1443.0  10.038808  5.427228  1.0  5.0  10.0  15.0  19.0

*** Results for month ***
                 count  unique  top  freq
count_category
normal number     9443      12    2   868
abnormal number   1443      12    6   185

*** Results for year ***
                 count  unique   top  freq
count_category
normal number     9443       2  2011  5098
abnormal number   1443       2  2012  1119

*** Results for times of the day ***
                count unique       top  freq
count_category
normal number    9443      5    morning  4606
abnormal number  1443      4  afternoon   605

*** Results for temp of the day ***
                count unique   top  freq
count_category
normal number    9443      3  cold  4992
abnormal number  1443      3  mild   739

*** Results for windspeed of the day ***
                count unique   top  freq
count_category
normal number    9443      3  calm  8156
abnormal number  1443      3  calm  1235

*** Results for count_category ***
                count unique              top  freq
count_category
normal number    9443      1    normal number  9443
abnormal number  1443      1  abnormal number  1443
```

In [55]:

```
train[train["count_category"] == 'abnormal number'].head()
```

Out[55]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
1579	2011-04-11 17:00:00	summer	0	1	clear	30.34	33.335	48	35.0008	100
1747	2011-04-18 17:00:00	summer	0	1	clear	23.78	27.275	49	19.0012	66
1771	2011-04-19 17:00:00	summer	0	1	clear	22.96	26.515	60	7.0015	39
1772	2011-04-19 18:00:00	summer	0	1	few clouds	22.14	25.760	64	8.9981	44
1819	2011-05-02 17:00:00	summer	0	1	clear	27.06	31.060	65	12.9980	65

In [56]:

```
train[train["count_category"] == 'normal number'].head()
```

Out[56]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regist
0	2011-01-01 00:00:00	spring	0	0	clear	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	spring	0	0	clear	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	spring	0	0	clear	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	spring	0	0	clear	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	spring	0	0	clear	9.84	14.395	75	0.0	0	

In [57]:

```
train[train["count_category"] == 'normal number'].describe().style.background_gradient(
```

Out[57]:

	temp	atemp	humidity	windspeed	casual	registered	count	
count	9443.000000	9443.000000	9443.000000	9443.000000	9443.000000	9443.000000	9443.000000	944
mean	19.463603	22.829653	63.297045	12.632834	25.363550	111.189453	136.553002	9
std	7.688941	8.400155	19.143434	8.213963	32.171598	91.292051	112.433425	5
min	0.820000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000	1
25%	13.120000	15.910000	49.000000	7.001500	3.000000	27.000000	32.000000	5
50%	18.860000	22.725000	64.000000	11.001400	12.000000	96.000000	116.000000	10
75%	25.420000	30.305000	79.000000	16.997900	36.000000	175.000000	219.000000	15
max	41.000000	45.455000	100.000000	56.996900	240.000000	391.000000	400.000000	19

In [58]:

```
train[train["count_category"] == 'abnormal number'].describe().style.background_gradien
```

Out[58]:

	temp	atemp	humidity	windspeed	casual	registered	count	
count	1443.000000	1443.000000	1443.000000	1443.000000	1443.000000	1443.000000	1443.000000	144
mean	25.251795	29.056712	52.655579	13.889374	105.770617	445.862093	551.632710	10
std	6.486419	6.814310	17.258856	7.748206	80.325962	142.848973	123.947170	5
min	4.100000	6.060000	16.000000	0.000000	1.000000	173.000000	401.000000	1
25%	20.500000	24.240000	39.000000	8.998100	39.000000	342.000000	453.500000	5
50%	26.240000	31.060000	51.000000	12.998000	84.000000	417.000000	516.000000	10
75%	30.340000	34.090000	65.000000	19.001200	164.000000	532.000000	623.000000	15
max	37.720000	42.425000	100.000000	43.998900	367.000000	886.000000	977.000000	19

What is the best weather condition in which users rent bicycles ?

In [59]:

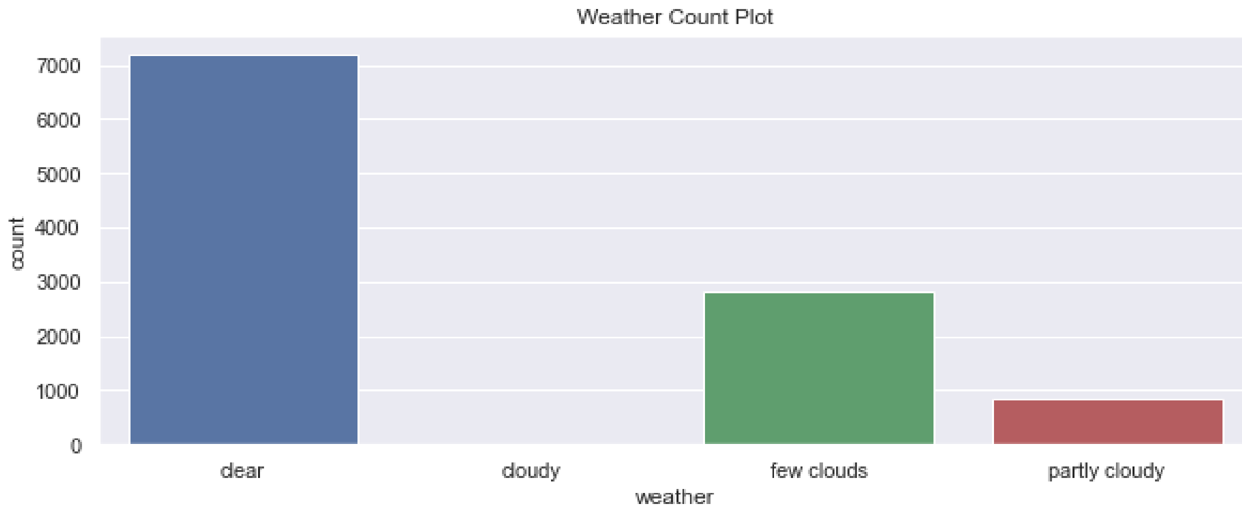
```
train['weather'].value_counts()
```

Out[59]:

```
clear            7192
few clouds       2834
partly cloudy     859
cloudy              1
Name: weather, dtype: int64
```

In [60]:

```
plt.figure(figsize= (11, 4))
sns.countplot(data = train, x = 'weather')
plt.title('Weather Count Plot');
plt.show();
```



the best weather condition in which users rent bicycles when the weather is clear .

In [ ]:

```

```

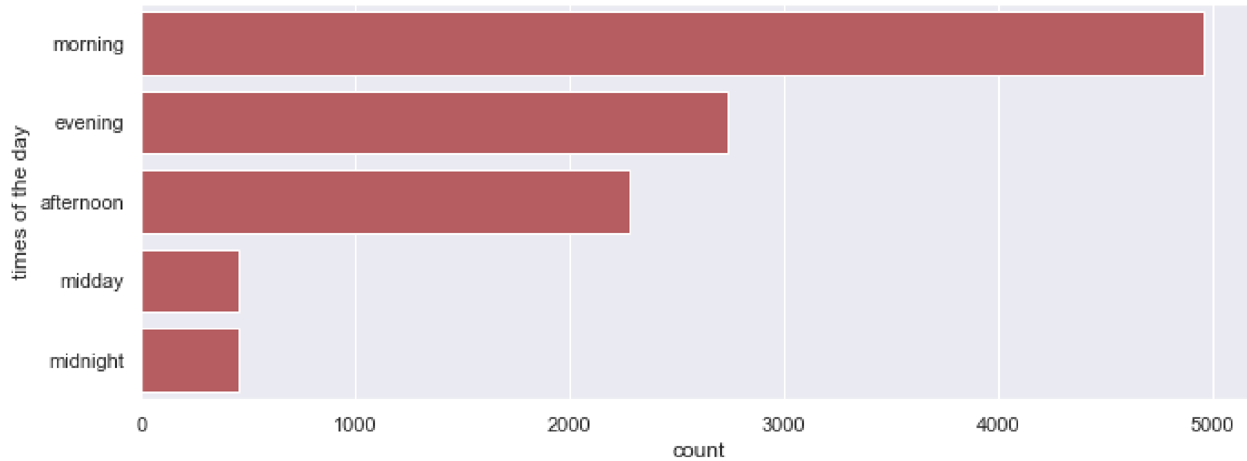
What is the best time of the day in which users rent bicycles ?

In [61]:

```
plt.figure(figsize= (11, 4))
# The `color_palette()` returns the the current / default palette as a list of RGB tupl
# Each tuple consists of three digits specifying the red, green, and blue channel value
# Choose the first tuple of RGB colors
base_color = sns.color_palette()[3]
# Dynamic-ordering the bars
# The order of the display of the bars can be computed with the following logic.
# Count the frequency of each unique value in the 'times of the day' column, and sort i
# Returns a Series
freq = train['times of the day'].value_counts()

# Get the indexes of the Series
gen_order = freq.index

# Plot the bar chart in the decreasing order of the frequency of the `times of the day`
sns.countplot(data=train, y='times of the day', color=base_color, order=gen_order);
```



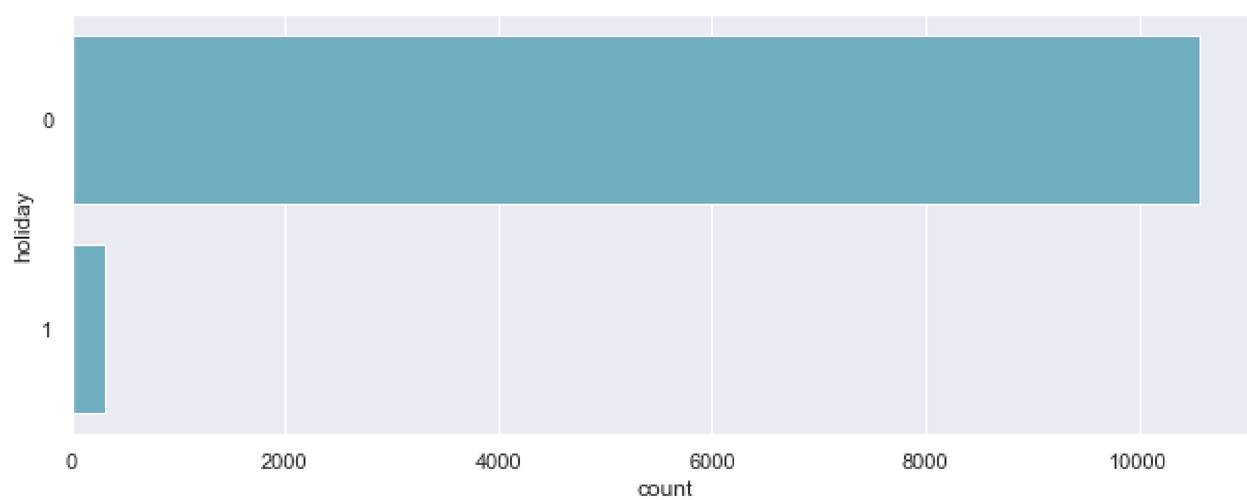
the best time of the day in which users rent bicycles is morning .

what is the most day type (Holiday or Not) in which users rent bicycles ?

In [62]:
```
plt.figure(figsize= (11, 4))
# The `color_palette()` returns the the current / default palette as a list of RGB tupl
# Each tuple consists of three digits specifying the red, green, and blue channel value
# Choose the first tuple of RGB colors
base_color = sns.color_palette()[9]
# Dynamic-ordering the bars
# The order of the display of the bars can be computed with the following logic.
# Count the frequency of each unique value in the 'times of the day' column, and sort i
# Returns a Series
freq = train['holiday'].value_counts()

# Get the indexes of the Series
gen_order = freq.index

# Plot the bar chart in the decreasing order of the frequency of the `times of the day`
sns.countplot(data=train, y='holiday', color=base_color, order=gen_order);
```



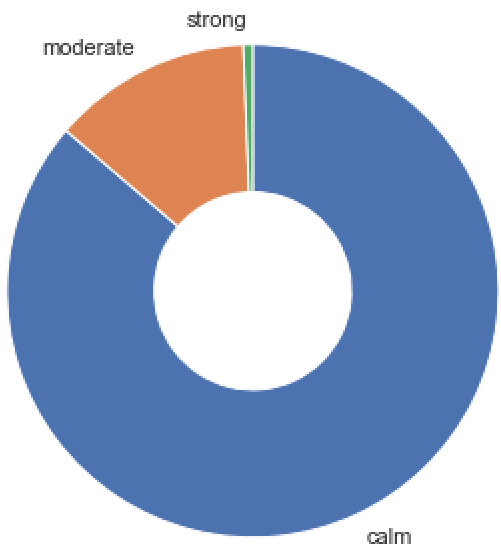
the most day type (Holiday or Working Day) in which users rent

bicycles is Not Holiday(Working Day) .

What is the best windspeed state of the day in which users rent bicycles ?

In [63]:
```
plt.figure(figsize= (11, 5))
sorted_counts = train['windspeed of the day'].value_counts()

plt.pie(sorted_counts, labels = sorted_counts.index, startangle = 90,
        counterclock = False, wedgeprops = {'width' : 0.6});
plt.axis('square')
plt.show()
```

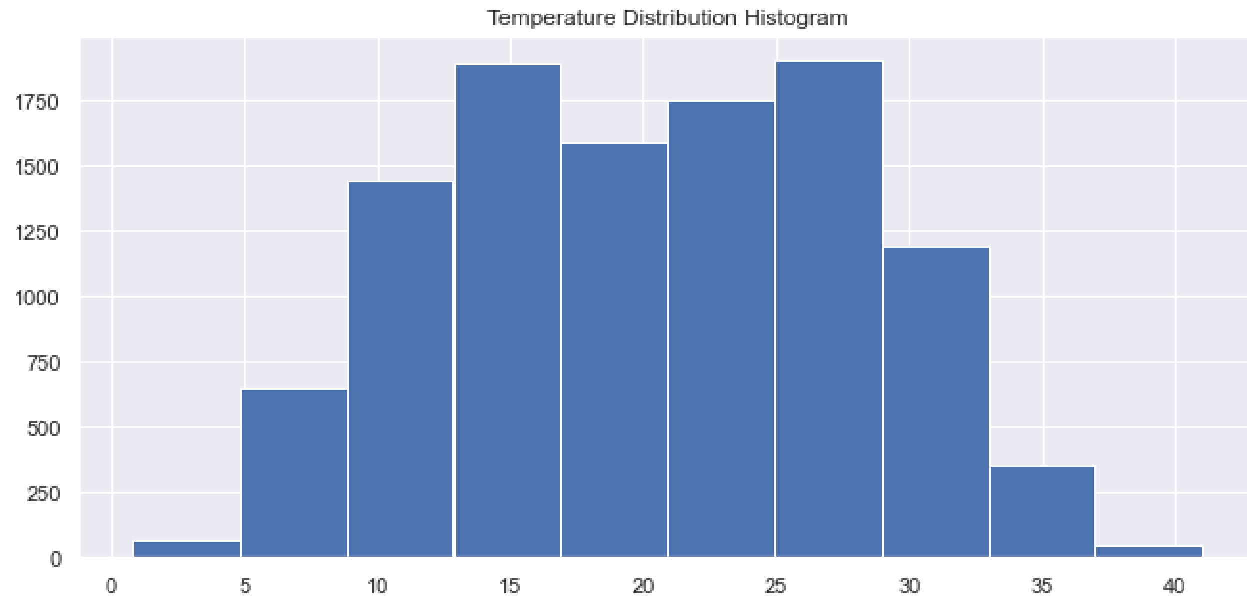


the best windspeed of the day in which users rent bicycles when it is Calm.

In [ ]:
```
```

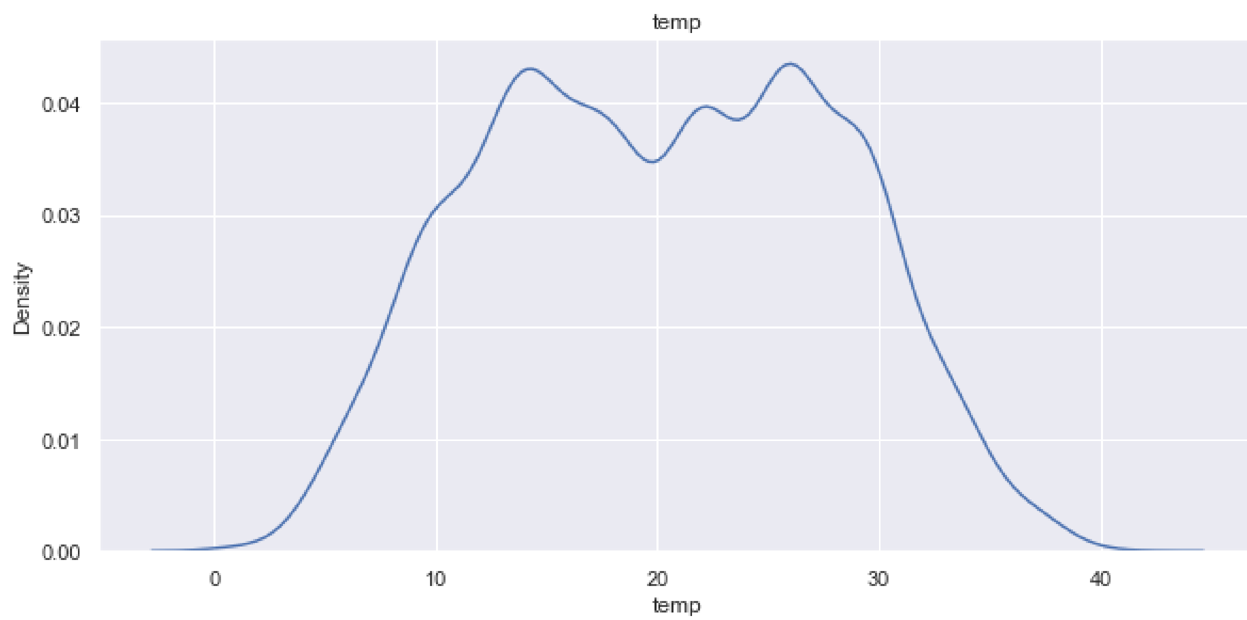
What is the best weather degree in which users rent bicycles ?

In [64]:
```
plt.figure(figsize= (11, 5))
train['temp'].hist();
plt.title('Temperature Distribution Histogram')
plt.show()
```



In [65]:

```
plt.figure(figsize= (11, 5))
sns.distplot(train['temp'], bins = bins)
#train['windspeed'].hist(figsize= (11, 5));
plt.title('temp')
plt.show()
```



from the distribution plot the best weather degree are (15 and 27.5)

In [ ]:

In [66]:

```
train.columns
```

Out[66]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
```

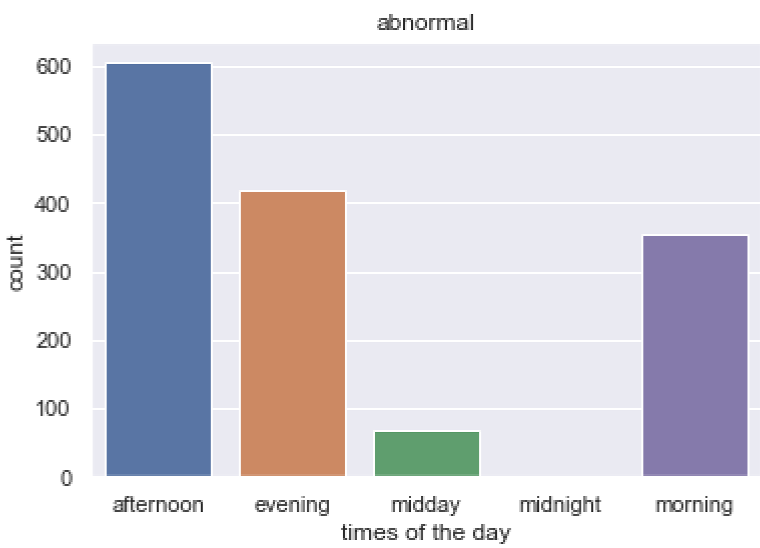
```
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'dayofweek', 'hour', 'day', 'month', 'year', 'times of the day',
       'temp of the day', 'windspeed of the day', 'count_category'],
      dtype='object')
```

Analyze ('times of the day', 'temp of the day', 'windspeed of the day') When the count of Renting Bicycles is Normal or Abnormal.

- Times of the day when count of Renting Bicycles is normal and abnormal

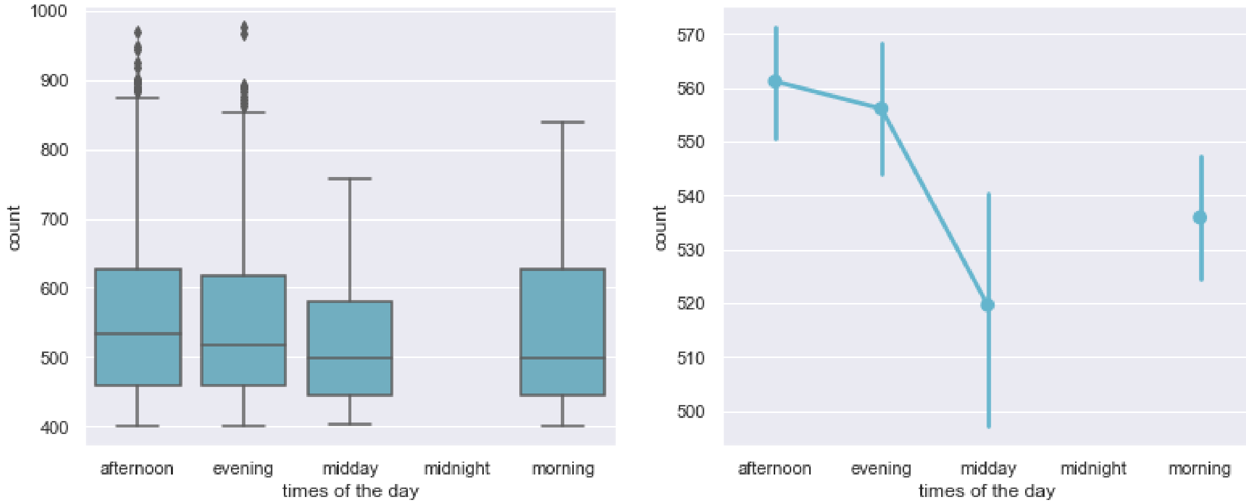
In [215…

```
sns.countplot(train[train["count_category"] == 'abnormal number']['times of the day'])
plt.title("abnormal")
plt.show()
```



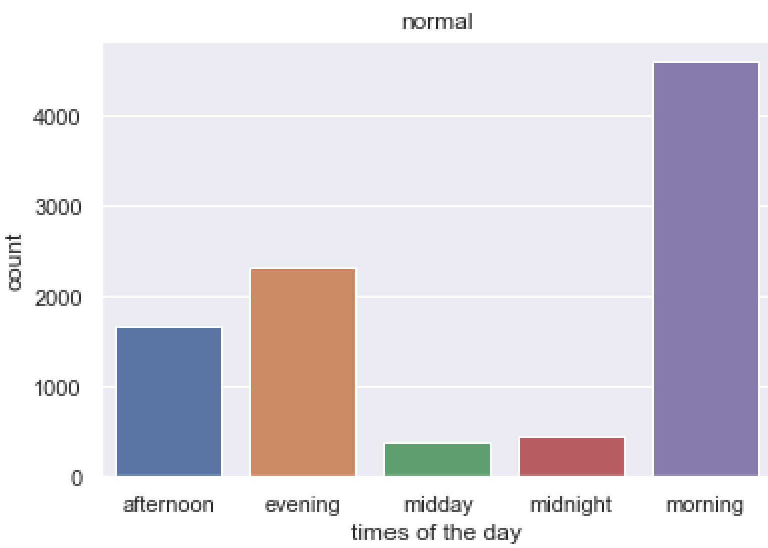
In [68]:

```
base_color = sns.color_palette()[9]
plt.figure(figsize= (13, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=train[train["count_category"] == 'abnormal number'], x='times of the d
plt.subplot(1, 2, 2)
sns.pointplot(data=train[train["count_category"] == 'abnormal number'], x='times of the
plt.show()
```



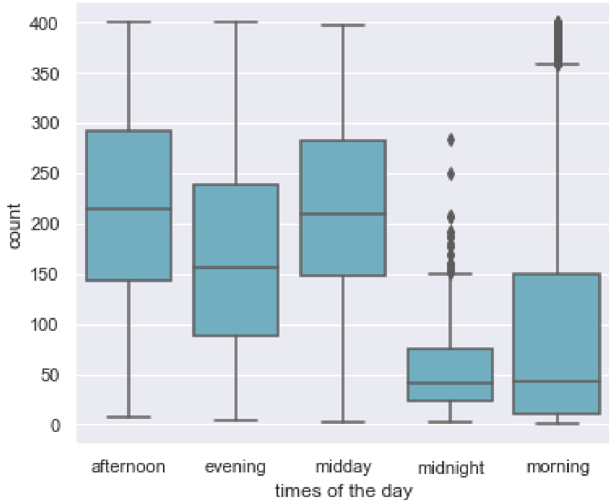
In [216…

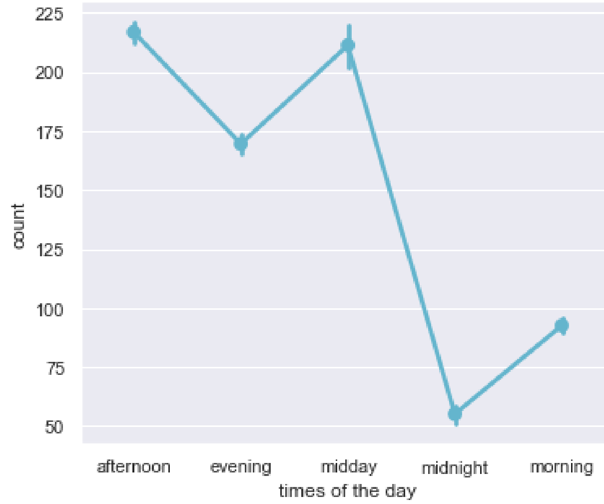
```
sns.countplot(train[train["count_category"] == 'normal number']['times of the day'])
plt.title("normal")
plt.show()
```



In [70]:

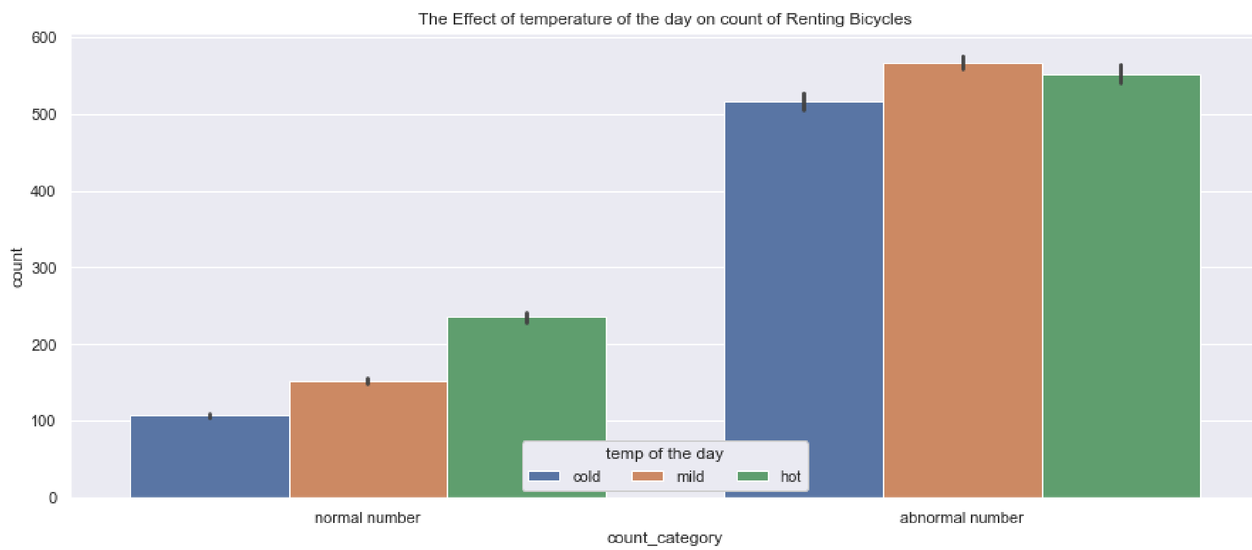
```
base_color = sns.color_palette()[9]
plt.figure(figsize= (13, 5))
plt.subplot(1, 2, 1)
sns.boxplot(data=train[train["count_category"] == 'normal number'], x='times of the day
plt.subplot(1, 2, 2)
sns.pointplot(data=train[train["count_category"] == 'normal number'], x='times of the d
plt.show()
```





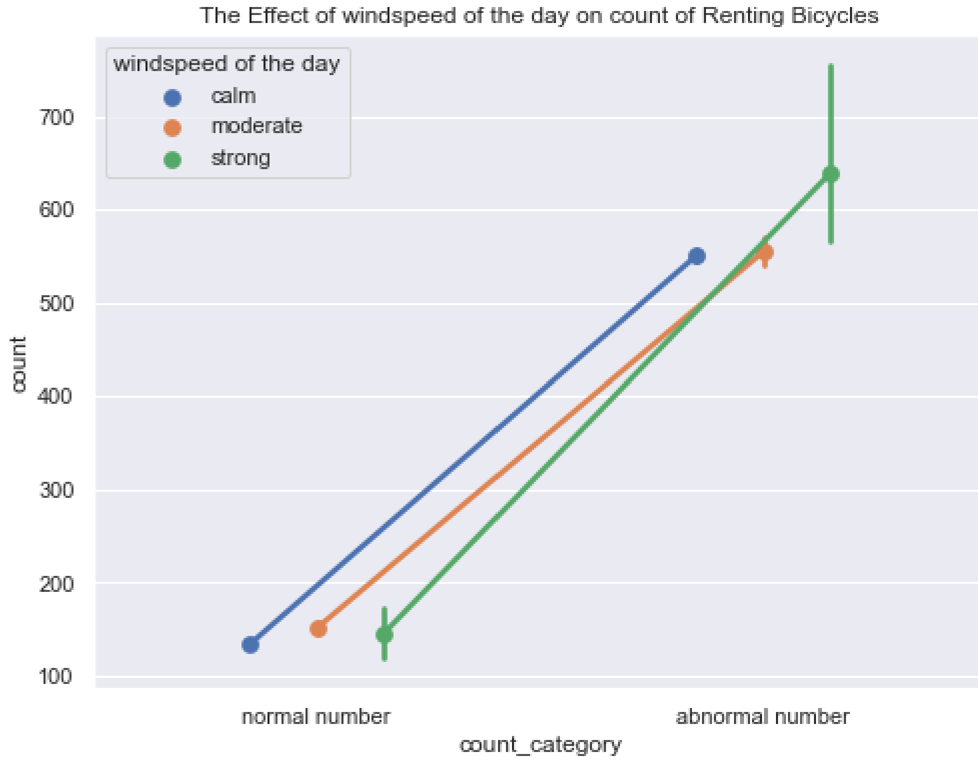
In [71]:

```
plt.figure(figsize= (15, 6))
ax = sns.barplot(data = train, x = 'count_category', y = 'count', hue = 'temp of the da
ax.legend(loc = 8, ncol = 3, framealpha = 1, title = 'temp of the day')
plt.title("The Effect of temperature of the day on count of Renting Bicycles")
plt.show()
```



In [72]:

```
plt.figure(figsize= (8, 6))
ax = sns.pointplot(data = train, x = 'count_category', y = 'count', hue = 'windspeed of
                   dodge = 0.3, linestyles = "-")
#plt.xticks(rotation= 45);
plt.title("The Effect of windspeed of the day on count of Renting Bicycles")
plt.show()
```



Drop Columns Based on the above findings, the below columns can be dropped due to the following reasons:

No longer relevant: 'datetime'

Very high correlation with other features / label:

- 'season'
- 'casual'
- 'registered'
- 'atemp'

Has no correlation with 'count':

- 'day'

In [73]:
```
train_trimmed = train.drop(["datetime", "season", "casual", "registered", "day", 'atemp
test_trimmed = test.drop(["datetime", "season", "day", 'atemp'], axis=1)
display(train_trimmed.head())
```

	holiday	workingday	weather	temp	humidity	windspeed	count	dayofweek	hour	month	year
0	0	0	clear	9.84	81	0.0	16	5	0	1	2011
1	0	0	clear	9.02	80	0.0	40	5	1	1	2011
2	0	0	clear	9.02	80	0.0	32	5	2	1	2011
3	0	0	clear	9.84	75	0.0	13	5	3	1	2011

	holiday	workingday	weather	temp	humidity	windspeed	count	dayofweek	hour	month	year
4	0	0	clear	9.84	75	0.0	1	5	4	1	2011

In [74]:
```
display(test_trimmed.head())
```

	holiday	workingday	weather	temp	humidity	windspeed	dayofweek	hour	month	year	times c the da
0	0	1	clear	10.66	56	26.0027	3	0	1	2011	midnigl
1	0	1	clear	10.66	56	0.0000	3	1	1	2011	mornir
2	0	1	clear	10.66	56	0.0000	3	2	1	2011	mornir
3	0	1	clear	10.66	56	11.0014	3	3	1	2011	mornir
4	0	1	clear	10.66	56	11.0014	3	4	1	2011	mornir

In [75]:
```
def detect_outlier(feature):
    outliers = []
    data = train_trimmed[feature]
    mean = np.mean(data)
    std =np.std(data)


    for y in data:
        z_score= (y - mean)/std
        if np.abs(z_score) > 3:
            outliers.append(y)
    print('\nOutlier caps for {}:'.format(feature))
    print('  --95p: {:.1f} / {} values exceed that'.format(data.quantile(.95),
                                                            len([i for i in data
                                                                 if i > data.quantile(
    print('  --3sd: {:.1f} / {} values exceed that'.format(mean + 3*(std), len(outliers
    print('  --99p: {:.1f} / {} values exceed that'.format(data.quantile(.99),
                                                           len([i for i in data
                                                                if i > data.quantile(.9
```

In [76]:
```
# Determine what the upperbound should be for continuous features
for feat in ['temp', 'humidity', 'windspeed']:
    detect_outlier(feat)
```

```
Outlier caps for temp:
  --95p: 32.8 / 403 values exceed that
  --3sd: 43.6 / 0 values exceed that
  --99p: 36.1 / 94 values exceed that

Outlier caps for humidity:
```

```
  --95p: 93.0 / 474 values exceed that
  --3sd: 119.6 / 22 values exceed that
  --99p: 100.0 / 0 values exceed that

Outlier caps for windspeed:
  --95p: 28.0 / 427 values exceed that
  --3sd: 37.3 / 67 values exceed that
  --99p: 35.0 / 89 values exceed that
```

In [77]:

```
train_trimmed.describe().T.style.background_gradient(cmap='PuBuGn')
```

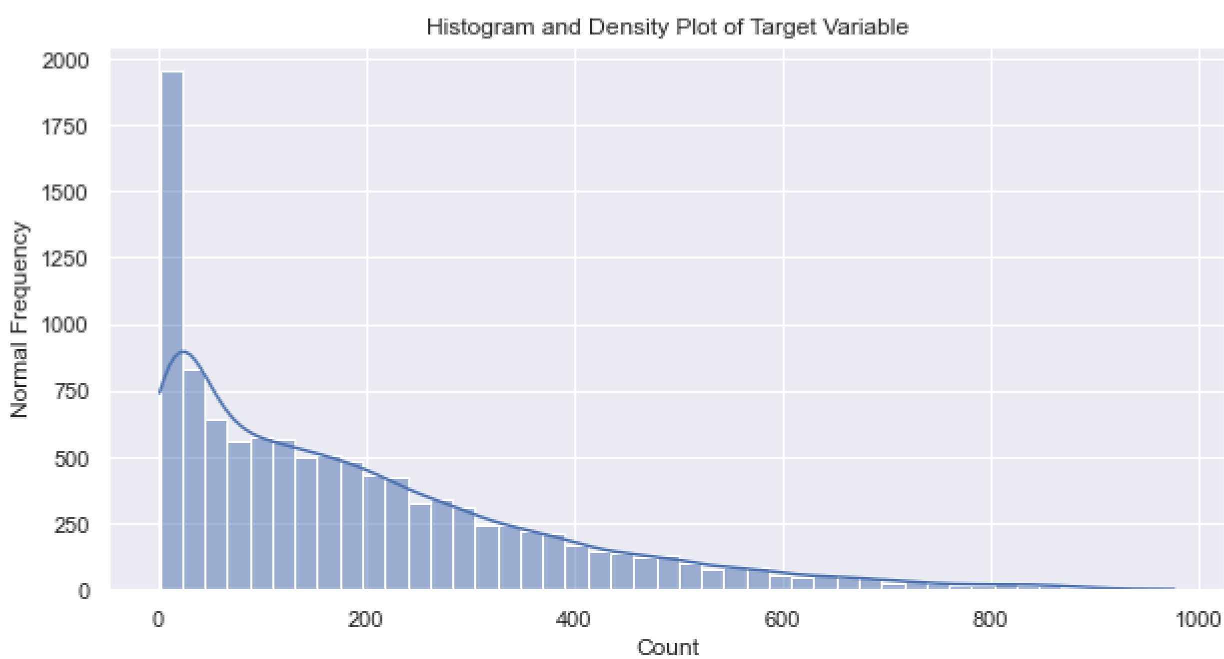
Out[77]:

	count	mean	std	min	25%	50%	75%	ma
temp	10886.000000	20.230860	7.791590	0.820000	13.940000	20.500000	26.240000	41.0000(
humidity	10886.000000	61.886460	19.245033	0.000000	47.000000	62.000000	77.000000	100.0000(
windspeed	10886.000000	12.799395	8.164537	0.000000	7.001500	12.998000	16.997900	56.9969(
count	10886.000000	191.574132	181.144454	1.000000	42.000000	145.000000	284.000000	977.0000(

In [78]:

```
fig, ax = plt.subplots(1,1, figsize=(10, 5))
ax.set_ylabel("Normal Frequency")
ax.set_xlabel("Count")
ax.set_title("Histogram and Density Plot of Target Variable")

sns.histplot(train_trimmed["count"], kde=True, ax=ax)
plt.show()
```



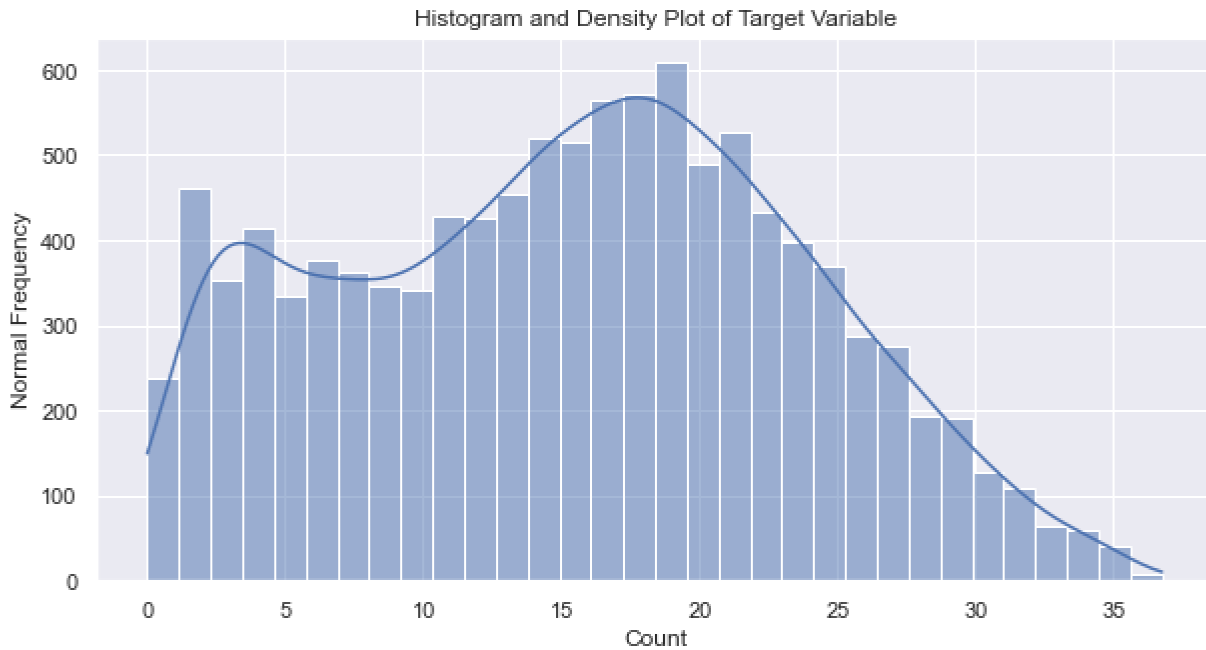
In [79]:

```
# Transforming target variable using boxcox transformation
train_trimmed['count'] = boxcox(train_trimmed['count'], 0.4)

fig, ax = plt.subplots(1,1, figsize=(10, 5))
ax.set_ylabel("Normal Frequency")
ax.set_xlabel("Count")
```

```
ax.set_title("Histogram and Density Plot of Target Variable")

sns.histplot(train_trimmed['count'], kde=True, ax=ax)
plt.show()# Transforming target variable using boxcox transformation
```



In [80]:

```
train_trimmed.head()
```

Out[80]:

	holiday	workingday	weather	temp	humidity	windspeed	count	dayofweek	hour	month	yea
0	0	0	clear	9.84	81	0.0	5.078583	5	0	1	201
1	0	0	clear	9.02	80	0.0	8.433621	5	1	1	201
2	0	0	clear	9.02	80	0.0	7.500000	5	2	1	201
3	0	0	clear	9.84	75	0.0	4.474569	5	3	1	201
4	0	0	clear	9.84	75	0.0	0.000000	5	4	1	201

In [81]:

```
# we need to scall continuous Features
train_trimmed.describe().T.style.background_gradient(cmap='PuBuGn')
```

Out[81]:

	count	mean	std	min	25%	50%	75%	max
temp	10886.000000	20.230860	7.791590	0.820000	13.940000	20.500000	26.240000	41.000000
humidity	10886.000000	61.886460	19.245033	0.000000	47.000000	62.000000	77.000000	100.000000
windspeed	10886.000000	12.799395	8.164537	0.000000	7.001500	12.998000	16.997900	56.996900
count	10886.000000	15.371114	8.307670	0.000000	8.649098	15.801522	21.447895	36.755258

In [82]:
```
# we need to scall continuous Features
train_trimmed.describe().T.style.background_gradient(cmap='PuBuGn')
```

Out[82]:

	count	mean	std	min	25%	50%	75%	max
temp	10886.000000	20.230860	7.791590	0.820000	13.940000	20.500000	26.240000	41.000000
humidity	10886.000000	61.886460	19.245033	0.000000	47.000000	62.000000	77.000000	100.000000
windspeed	10886.000000	12.799395	8.164537	0.000000	7.001500	12.998000	16.997900	56.996900
count	10886.000000	15.371114	8.307670	0.000000	8.649098	15.801522	21.447895	36.755258

In [83]:
```
test_trimmed.describe().T.style.background_gradient(cmap='PuBuGn')
```

Out[83]:

	count	mean	std	min	25%	50%	75%	max
temp	6493.000000	20.620607	8.059583	0.820000	13.940000	21.320000	27.060000	40.180000
humidity	6493.000000	64.125212	19.293391	16.000000	49.000000	65.000000	81.000000	100.000000
windspeed	6493.000000	12.631157	8.250151	0.000000	7.001500	11.001400	16.997900	55.998600

In [84]:
```
train_trimmed.var()
```

Out[84]:
```
temp          60.708872
humidity     370.371306
windspeed     66.659670
count         69.017380
dtype: float64
```

In [85]:
```
test_trimmed.var()
```

Out[85]:
```
temp          64.956879
humidity     372.234936
windspeed     68.064994
dtype: float64
```

In [86]:
```
train_trimmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype   
---  ------            --------------  -----   
 0   holiday           10886 non-null  category
 1   workingday        10886 non-null  category
 2   weather           10886 non-null  category
 3   temp              10886 non-null  float64 
 4   humidity          10886 non-null  int64   
 5   windspeed         10886 non-null  float64 
 6   count             10886 non-null  float64 
 7   dayofweek         10886 non-null  object  
 8   hour              10886 non-null  object  
 9   month             10886 non-null  category
 10  year              10886 non-null  category
 11  times of the day  10886 non-null  category
```

```
 12  temp of the day       10886 non-null  category
 13  windspeed of the day  10886 non-null  category
dtypes: category(8), float64(3), int64(1), object(2)
memory usage: 596.9+ KB
```

In [ ]:

In [87]:

```python
# Get the dummies columns for categorical columns.

cat_cols = train_trimmed.select_dtypes(include=['category']).columns.tolist()
print(cat_cols)
```

```
['holiday', 'workingday', 'weather', 'month', 'year', 'times of the day', 'temp of the d
ay', 'windspeed of the day']
```

In [88]:

```python
train_trimmed_dummies = pd.get_dummies(train_trimmed, drop_first= True,columns = cat_co
test_trimmed_dummies = pd.get_dummies(test_trimmed, drop_first= True,columns = cat_cols
train_trimmed_dummies.columns
```

Out[88]:

```
Index(['temp', 'humidity', 'windspeed', 'count', 'dayofweek', 'hour',
       'holiday_1', 'workingday_1', 'weather_cloudy', 'weather_few clouds',
       'weather_partly cloudy', 'month_2', 'month_3', 'month_4', 'month_5',
       'month_6', 'month_7', 'month_8', 'month_9', 'month_10', 'month_11',
       'month_12', 'year_2012', 'times of the day_evening',
       'times of the day_midday', 'times of the day_midnight',
       'times of the day_morning', 'temp of the day_mild',
       'temp of the day_hot', 'windspeed of the day_moderate',
       'windspeed of the day_strong'],
      dtype='object')
```

In [89]:

```python
train_trimmed_dummies.head()
```

Out[89]:

	temp	humidity	windspeed	count	dayofweek	hour	holiday_1	workingday_1	weather_cloudy
0	9.84	81	0.0	5.078583	5	0	0	0	0
1	9.02	80	0.0	8.433621	5	1	0	0	0
2	9.02	80	0.0	7.500000	5	2	0	0	0
3	9.84	75	0.0	4.474569	5	3	0	0	0
4	9.84	75	0.0	0.000000	5	4	0	0	0

In [90]:

```python
test_trimmed_dummies.head()
```

Out[90]:

	temp	humidity	windspeed	dayofweek	hour	holiday_1	workingday_1	weather_cloudy	weather_fe clou
0	10.66	56	26.0027	3	0	0	1	0	
1	10.66	56	0.0000	3	1	0	1	0	

	temp	humidity	windspeed	dayofweek	hour	holiday_1	workingday_1	weather_cloudy	weather_fe clou
2	10.66	56	0.0000	3	2	0	1	0	
3	10.66	56	11.0014	3	3	0	1	0	
4	10.66	56	11.0014	3	4	0	1	0	

In [91]:

```
test_trimmed_dummies.sample(20)
```

Out[91]:

	temp	humidity	windspeed	dayofweek	hour	holiday_1	workingday_1	weather_cloudy	weathe c
5687	20.50	72	8.9981	6	19	0	0	0	
497	22.14	64	15.0013	0	14	0	1	0	
2945	14.76	62	0.0000	1	7	0	1	0	
2042	31.98	43	23.9994	6	15	0	0	0	
1175	28.70	65	12.9980	3	23	0	1	0	
1188	31.16	55	19.9995	4	12	0	1	0	
165	8.20	75	8.9981	4	1	0	1	0	
1012	18.04	94	6.0032	4	4	0	1	0	
207	7.38	74	6.0032	5	21	0	0	0	
3801	19.68	100	6.0032	3	5	0	1	0	
1298	26.24	89	6.0032	0	2	0	1	0	
4466	28.70	74	16.9979	5	22	0	0	0	
5133	31.98	52	15.0013	1	17	0	1	0	
6265	10.66	48	22.0028	5	10	0	0	0	
3958	24.60	60	15.0013	2	18	0	1	0	
3341	13.12	81	0.0000	1	22	0	1	0	
3064	9.02	80	7.0015	6	7	0	0	0	
5293	27.88	79	16.9979	6	9	0	0	0	
335	13.12	29	7.0015	2	14	0	1	0	
873	24.60	83	8.9981	0	9	0	1	0	

Modelling and Model Evaluation

In [150…

```
X = train_trimmed_dummies.drop(['count'], axis=1) # drop target variable column
y = pd.Series(train_trimmed_dummies['count'])
# Do a train-test split
X_train, X_check, y_train, y_check = train_test_split(X, y, test_size=0.25, random_stat
```

In [151…

```
y.describe()
```

Out[151…

```
count    10886.000000
mean        15.371114
std          8.307670
min          0.000000
25%          8.649098
50%         15.801522
75%         21.447895
max         36.755258
Name: count, dtype: float64
```

In [152…

```
# Check if the indices match

print((X_train.index == y_train.index).all())
print((X_check.index == y_check.index).all())
```

```
True
True
```

In [153…

```
# Standard scale the input variables (features)
scl = StandardScaler()
scl.fit(X_train)

X_train_scaled = scl.transform(X_train)
X_check_scaled = scl.transform(X_check)
```

In [154…

```
print(f"Number of Rows, Features in Training Dataset: {X_train_scaled.shape}")
print(f"Number of Rows, Features in Check Dataset: {X_check_scaled.shape}")
print(f"Number of Rows in Training Target: {y_train.shape}")
print(f"Number of Rows in Check Target: {y_check.shape}")
```

```
Number of Rows, Features in Training Dataset: (8164, 30)
Number of Rows, Features in Check Dataset: (2722, 30)
Number of Rows in Training Target: (8164,)
Number of Rows in Check Target: (2722,)
```

Regression models

1. Linear regression
2. Ridge regression
3. Lasso regression
4. Random Forest Regressor
5. Gradient Boosting Regressor

In [175…

```
R2=[]
MAPE=[]
```

```
MAE=[]
RMSE=[]
```

In [176…

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
lr = LinearRegression()
lr.fit(X_train_scaled,y_train)
```

Out[176…

```
LinearRegression()
```

In [177…

```
## test
predicted = lr.predict(X_check_scaled)
```

In [178…

```
from sklearn import metrics

print("Mean Absolute Error:", "{:,.0f}".format(metrics.mean_absolute_error(y_check, pre
print('Mean Squared Error:', metrics.mean_squared_error(y_check, predicted))
print("Root Mean Squared Error :", "{:,.0f}".format(np.sqrt(metrics.mean_squared_error(
print("R2 (explained variance):", round(metrics.r2_score(y_check, predicted), 2))
R2.append(metrics.r2_score(y_check, predicted))
MAPE.append(np.mean(np.abs((y_check-predicted)/predicted)))
MAE.append(metrics.mean_absolute_error(y_check, predicted))
RMSE.append(np.sqrt(metrics.mean_squared_error(y_check, predicted)))
```

```
Mean Absolute Error: 4
Mean Squared Error: 29.511417568164507
Root Mean Squared Error : 5
R2 (explained variance): 0.57
```

Ridge regression

In [179…

```
# Required Libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
from sklearn import model_selection
from sklearn.linear_model import RidgeCV
```

In [180…

```
ridge_model = Ridge(alpha=0.1).fit(X_train_scaled,y_train)
ridge_model
```

Out[180…

```
Ridge(alpha=0.1)
```

In [181…

```
## test
predicted = ridge_model.predict(X_check_scaled)
```

In [182…

```
from sklearn import metrics
```

```
print("Mean Absolute Error:", "{:,.0f}".format(metrics.mean_absolute_error(y_check, pre
print('Mean Squared Error:', metrics.mean_squared_error(y_check, predicted))
print("Root Mean Squared Error :", "{:,.0f}".format(np.sqrt(metrics.mean_squared_error(
print("R2 (explained variance):", round(metrics.r2_score(y_check, predicted), 2))
R2.append(metrics.r2_score(y_check, predicted))
MAPE.append(np.mean(np.abs((y_check-predicted)/predicted)))
MAE.append(metrics.mean_absolute_error(y_check, predicted))
RMSE.append(np.sqrt(metrics.mean_squared_error(y_check, predicted)))
```

```
Mean Absolute Error: 4
Mean Squared Error: 29.511478351752817
Root Mean Squared Error : 5
R2 (explained variance): 0.57
```

Lasso regression

In [183…

```
# Required Libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import Ridge,Lasso
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import model_selection
from sklearn.linear_model import RidgeCV, LassoCV
```

In [184…

```
lasso_cv_model = LassoCV(cv=10,max_iter=100000).fit(X_train_scaled,y_train)
lasso_cv_model
```

Out[184…

```
LassoCV(cv=10, max_iter=100000)
```

In [185…

```
lasso_tuned = Lasso().set_params(alpha= lasso_cv_model.alpha_).fit(X_train_scaled,y_tra
predicted = lasso_tuned.predict(X_check_scaled)
```

In [186…

```
from sklearn import metrics

print("Mean Absolute Error:", "{:,.0f}".format(metrics.mean_absolute_error(y_check, pre
print('Mean Squared Error:', metrics.mean_squared_error(y_check, predicted))
print("Root Mean Squared Error :", "{:,.0f}".format(np.sqrt(metrics.mean_squared_error(
print("R2 (explained variance):", round(metrics.r2_score(y_check, predicted), 2))
R2.append(metrics.r2_score(y_check, predicted))
MAPE.append(np.mean(np.abs((y_check-predicted)/predicted)))
MAE.append(metrics.mean_absolute_error(y_check, predicted))
RMSE.append(np.sqrt(metrics.mean_squared_error(y_check, predicted)))
```

```
Mean Absolute Error: 4
Mean Squared Error: 29.521596720493392
Root Mean Squared Error : 5
R2 (explained variance): 0.57
```

Random Forest Regressor

In [187…
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train_scaled,y_train)
```

Out[187…
```
RandomForestRegressor()
```

In [188…
```
predicted = rfr.predict(X_check_scaled)
```

In [189…
```
from sklearn import metrics

print("Mean Absolute Error:", "{:,.0f}".format(metrics.mean_absolute_error(y_check, pre
print('Mean Squared Error:', metrics.mean_squared_error(y_check, predicted))
print("Root Mean Squared Error :", "{:,.0f}".format(np.sqrt(metrics.mean_squared_error(
print("R2 (explained variance):", round(metrics.r2_score(y_check, predicted), 2))
R2.append(metrics.r2_score(y_check, predicted))
MAPE.append(np.mean(np.abs((y_check-predicted)/predicted)))
MAE.append(metrics.mean_absolute_error(y_check, predicted))
RMSE.append(np.sqrt(metrics.mean_squared_error(y_check, predicted)))
```

```
Mean Absolute Error: 1
Mean Squared Error: 3.0820790815685597
Root Mean Squared Error : 2
R2 (explained variance): 0.95
```

Gradient Boosting Regressor

In [190…
```
from sklearn.ensemble import GradientBoostingRegressor

GBRModel = GradientBoostingRegressor(n_estimators=100,max_depth=2,learning_rate = 1.5 ,
GBRModel.fit(X_train_scaled,y_train)
```

Out[190…
```
GradientBoostingRegressor(learning_rate=1.5, max_depth=2, random_state=33)
```

In [191…
```
predicted = GBRModel.predict(X_check_scaled)
```

In [192…
```
from sklearn import metrics

print("Mean Absolute Error:", "{:,.0f}".format(metrics.mean_absolute_error(y_check, pre
print('Mean Squared Error:', metrics.mean_squared_error(y_check, predicted))
print("Root Mean Squared Error :", "{:,.0f}".format(np.sqrt(metrics.mean_squared_error(
print("R2 (explained variance):", round(metrics.r2_score(y_check, predicted), 2))
R2.append(metrics.r2_score(y_check, predicted))
MAPE.append(np.mean(np.abs((y_check-predicted)/predicted)))
MAE.append(metrics.mean_absolute_error(y_check, predicted))
RMSE.append(np.sqrt(metrics.mean_squared_error(y_check, predicted)))
```

```
Mean Absolute Error: 2
Mean Squared Error: 5.439094519647775
Root Mean Squared Error : 2
R2 (explained variance): 0.92
```

Evaluation for regression models

In [195…

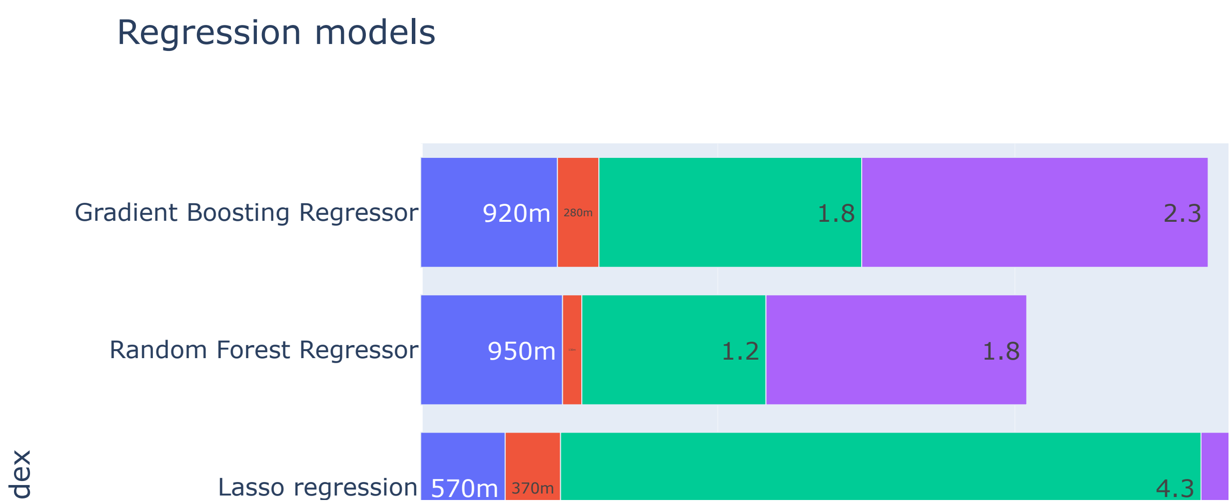
```
# import pandas as pd
import pandas as pd

# List1
lst = [R2,MAPE,MAE,RMSE]

d = pd.DataFrame({"R2":R2,"MAPE":MAPE,"MAE":MAE,"RMSE":RMSE},index=["Linear regression"
```

In [198…

```
import plotly.express as px
fig = px.bar(data_frame=d, y=d.index, x=["R2","MAPE","MAE","RMSE"], text_auto='0.2s',
            title="Regression models")
fig.update_traces(textfont_size=12, textangle=0,textposition="auto", cliponaxis=False)
fig.show()
```



Make Predictions on Test Dataset

In [200…

```
X_test = test_trimmed_dummies
X_test
```

Out[200…

	temp	humidity	windspeed	dayofweek	hour	holiday_1	workingday_1	weather_cloudy	weathe c
0	10.66	56	26.0027	3	0	0	1	0	
1	10.66	56	0.0000	3	1	0	1	0	
2	10.66	56	0.0000	3	2	0	1	0	
3	10.66	56	11.0014	3	3	0	1	0	
4	10.66	56	11.0014	3	4	0	1	0	
...	...	...	...	...	...	...	...	...	
6488	10.66	60	11.0014	0	19	0	1	0	
6489	10.66	60	11.0014	0	20	0	1	0	
6490	10.66	60	11.0014	0	21	0	1	0	
6491	10.66	56	8.9981	0	22	0	1	0	
6492	10.66	65	8.9981	0	23	0	1	0	

6493 rows × 30 columns

In [201…

```
X_test_scaled = scl.transform(X_test)
```

In [207…

```
rf_reg2 = RandomForestRegressor(n_estimators=500)
rf_model_refit = rf_reg2.fit(X_train_scaled, y_train)
y_test_pred2 = rf_model_refit.predict(X_test_scaled)
y_test_pred2
```

Out[207…

```
array([ 4.70573568,  2.06632037,  1.54829143, ..., 15.3554761 ,
       14.51678471, 11.47513259])
```

In [2]:

```
!pip install -U notebook-as-pdf
```

```
Requirement already satisfied: notebook-as-pdf in c:\users\lenovo\anaconda3\lib\site-pac
kages (0.5.0)
Requirement already satisfied: nbconvert in c:\users\lenovo\anaconda3\lib\site-packages
(from notebook-as-pdf) (6.1.0)
Requirement already satisfied: PyPDF2 in c:\users\lenovo\anaconda3\lib\site-packages (fr
om notebook-as-pdf) (2.11.1)
Requirement already satisfied: pyppeteer in c:\users\lenovo\anaconda3\lib\site-packages
(from notebook-as-pdf) (1.0.2)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\lenovo\appdata\roaming\pyt
hon\python39\site-packages (from nbconvert->notebook-as-pdf) (0.4)
Requirement already satisfied: jupyter-core in c:\users\lenovo\appdata\roaming\python\py
thon39\site-packages (from nbconvert->notebook-as-pdf) (4.9.2)
Requirement already satisfied: defusedxml in c:\users\lenovo\anaconda3\lib\site-packages
(from nbconvert->notebook-as-pdf) (0.7.1)
Requirement already satisfied: jinja2>=2.4 in c:\users\lenovo\anaconda3\lib\site-package
s (from nbconvert->notebook-as-pdf) (2.11.3)
Requirement already satisfied: traitlets>=5.0 in c:\users\lenovo\appdata\roaming\python
```

```
\python39\site-packages (from nbconvert->notebook-as-pdf) (5.1.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\lenovo\appdata\roaming\python
\python39\site-packages (from nbconvert->notebook-as-pdf) (2.11.2)
Requirement already satisfied: jupyterlab-pygments in c:\users\lenovo\anaconda3\lib\site
-packages (from nbconvert->notebook-as-pdf) (0.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\lenovo\anaconda3\lib\sit
e-packages (from nbconvert->notebook-as-pdf) (1.4.3)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\lenovo\anaconda3\lib\site-p
ackages (from nbconvert->notebook-as-pdf) (0.8.4)
Requirement already satisfied: bleach in c:\users\lenovo\anaconda3\lib\site-packages (fr
om nbconvert->notebook-as-pdf) (4.0.0)
WARNING: Ignoring invalid distribution -etuptools (c:\users\lenovo\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -etuptools (c:\users\lenovo\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -etuptools (c:\users\lenovo\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -etuptools (c:\users\lenovo\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -etuptools (c:\users\lenovo\anaconda3\lib\site-pa
ckages)
WARNING: Ignoring invalid distribution -etuptools (c:\users\lenovo\anaconda3\lib\site-pa
ckages)

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: testpath in c:\users\lenovo\anaconda3\lib\site-packages
(from nbconvert->notebook-as-pdf) (0.5.0)
Requirement already satisfied: nbformat>=4.4 in c:\users\lenovo\anaconda3\lib\site-packa
ges (from nbconvert->notebook-as-pdf) (5.1.3)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from nbconvert->notebook-as-pdf) (0.5.3)
Requirement already satisfied: typing-extensions>=3.10.0.0 in c:\users\lenovo\appdata\ro
aming\python\python39\site-packages (from PyPDF2->notebook-as-pdf) (4.1.1)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in c:\users\lenovo\anaconda3\lib\site-
packages (from pyppeteer->notebook-as-pdf) (8.2.2)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from pyppeteer->notebook-as-pdf) (1.26.12)
Requirement already satisfied: websockets<11.0,>=10.0 in c:\users\lenovo\anaconda3\lib\s
ite-packages (from pyppeteer->notebook-as-pdf) (10.4)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\lenovo\anaconda3\lib\si
te-packages (from pyppeteer->notebook-as-pdf) (1.4.4)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\lenovo\anaconda3\lib
\site-packages (from pyppeteer->notebook-as-pdf) (4.8.1)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\lenovo\anaconda3\lib\site
-packages (from pyppeteer->notebook-as-pdf) (4.62.3)
Requirement already satisfied: certifi>=2021 in c:\users\lenovo\appdata\roaming\python\p
ython39\site-packages (from pyppeteer->notebook-as-pdf) (2021.10.8)
Requirement already satisfied: zipp>=0.5 in c:\users\lenovo\anaconda3\lib\site-packages
(from importlib-metadata>=1.4->pyppeteer->notebook-as-pdf) (3.6.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from jinja2>=2.4->nbconvert->notebook-as-pdf) (1.1.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\lenovo\appdata\roaming
\python\python39\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook-as-pdf)
(7.1.2)
Requirement already satisfied: async-generator in c:\users\lenovo\anaconda3\lib\site-pac
kages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook-as-pdf) (1.10)
Requirement already satisfied: nest-asyncio in c:\users\lenovo\appdata\roaming\python\py
thon39\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert->notebook-as-pdf) (1.5.4)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\lenovo\anaconda3\lib
```

```
\site-packages (from nbformat>=4.4->nbconvert->notebook-as-pdf) (3.2.0)
Requirement already satisfied: ipython-genutils in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from nbformat>=4.4->nbconvert->notebook-as-pdf) (0.2.0)
Requirement already satisfied: colorama in c:\users\lenovo\appdata\roaming\python\python
39\site-packages (from tqdm<5.0.0,>=4.42.1->pyppeteer->notebook-as-pdf) (0.4.4)
Requirement already satisfied: packaging in c:\users\lenovo\anaconda3\lib\site-packages
(from bleach->nbconvert->notebook-as-pdf) (21.0)
Requirement already satisfied: six>=1.9.0 in c:\users\lenovo\appdata\roaming\python\pyth
on39\site-packages (from bleach->nbconvert->notebook-as-pdf) (1.16.0)
Requirement already satisfied: webencodings in c:\users\lenovo\anaconda3\lib\site-packag
es (from bleach->nbconvert->notebook-as-pdf) (0.5.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\lenovo\appdata\roaming\python\py
thon39\site-packages (from jupyter-core->nbconvert->notebook-as-pdf) (303)
Requirement already satisfied: attrs>=17.4.0 in c:\users\lenovo\anaconda3\lib\site-packa
ges (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert->notebook-as-pdf) (21.2.0)
Requirement already satisfied: setuptools in c:\users\lenovo\anaconda3\lib\site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert->notebook-as-pdf) (59.8.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\lenovo\anaconda3\lib\site-
packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert->notebook-as-pdf) (0.1
8.0)
Requirement already satisfied: tornado>=4.1 in c:\users\lenovo\appdata\roaming\python\py
thon39\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert->not
ebook-as-pdf) (6.1)
Requirement already satisfied: pyzmq>=13 in c:\users\lenovo\appdata\roaming\python\pytho
n39\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert->notebo
ok-as-pdf) (22.3.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\lenovo\appdata\roaming\p
ython\python39\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconv
ert->notebook-as-pdf) (2.8.2)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from packaging->bleach->nbconvert->notebook-as-pdf) (3.0.4)
```

In [3]:

```
!pyppeteer-install
```

```
[INFO] Starting Chromium download.

  0%|          | 0.00/137M [00:00<?, ?b/s]
  0%|          | 81.9k/137M [00:00<02:47, 817kb/s]
  0%|          | 297k/137M [00:00<01:33, 1.46Mb/s]
  1%|          | 686k/137M [00:00<00:57, 2.38Mb/s]
  1%|          | 1.09M/137M [00:00<00:46, 2.94Mb/s]
  1%|1         | 1.47M/137M [00:00<00:41, 3.27Mb/s]
  1%|1         | 1.94M/137M [00:00<00:38, 3.53Mb/s]
  2%|1         | 2.33M/137M [00:00<00:36, 3.67Mb/s]
  2%|2         | 2.78M/137M [00:00<00:36, 3.71Mb/s]
  2%|2         | 3.24M/137M [00:00<00:35, 3.80Mb/s]
  3%|2         | 3.62M/137M [00:01<00:34, 3.82Mb/s]
  3%|2         | 4.07M/137M [00:01<00:34, 3.84Mb/s]
  3%|3         | 4.47M/137M [00:01<00:34, 3.82Mb/s]
  4%|3         | 4.87M/137M [00:01<00:34, 3.87Mb/s]
  4%|3         | 5.26M/137M [00:01<00:34, 3.87Mb/s]
  4%|4         | 5.65M/137M [00:01<00:33, 3.87Mb/s]
  4%|4         | 6.04M/137M [00:01<00:33, 3.88Mb/s]
  5%|4         | 6.43M/137M [00:01<00:34, 3.81Mb/s]
  5%|4         | 6.82M/137M [00:01<00:34, 3.80Mb/s]
  5%|5         | 7.23M/137M [00:02<00:33, 3.82Mb/s]
  6%|5         | 7.62M/137M [00:02<00:33, 3.82Mb/s]
  6%|5         | 8.02M/137M [00:02<00:33, 3.85Mb/s]
  6%|6         | 8.41M/137M [00:02<00:33, 3.78Mb/s]
```

```
  6%|6         | 8.79M/137M [00:02<00:33, 3.78Mb/s]
  7%|6         | 9.16M/137M [00:02<00:33, 3.78Mb/s]
  7%|6         | 9.54M/137M [00:02<00:33, 3.76Mb/s]
  7%|7         | 9.92M/137M [00:02<00:33, 3.74Mb/s]
  8%|7         | 10.3M/137M [00:02<00:33, 3.74Mb/s]
  8%|7         | 10.7M/137M [00:02<00:34, 3.66Mb/s]
  8%|8         | 11.1M/137M [00:03<00:33, 3.79Mb/s]
  8%|8         | 11.5M/137M [00:03<00:33, 3.74Mb/s]
  9%|8         | 11.9M/137M [00:03<00:33, 3.75Mb/s]
  9%|8         | 12.2M/137M [00:03<00:33, 3.76Mb/s]
  9%|9         | 12.6M/137M [00:03<00:33, 3.71Mb/s]
 10%|9         | 13.0M/137M [00:03<00:33, 3.66Mb/s]
 10%|9         | 13.4M/137M [00:03<00:33, 3.69Mb/s]
 10%|#         | 13.8M/137M [00:03<00:33, 3.71Mb/s]
 10%|#         | 14.2M/137M [00:03<00:32, 3.73Mb/s]
 11%|#         | 14.5M/137M [00:03<00:33, 3.63Mb/s]
 11%|#         | 14.9M/137M [00:04<00:33, 3.62Mb/s]
 11%|#1        | 15.3M/137M [00:04<00:33, 3.59Mb/s]
 11%|#1        | 15.7M/137M [00:04<00:33, 3.64Mb/s]
 12%|#1        | 16.1M/137M [00:04<00:33, 3.65Mb/s]
 12%|#2        | 16.5M/137M [00:04<00:33, 3.61Mb/s]
 12%|#2        | 16.8M/137M [00:04<00:33, 3.63Mb/s]
 13%|#2        | 17.2M/137M [00:04<00:33, 3.61Mb/s]
 13%|#2        | 17.6M/137M [00:04<00:37, 3.20Mb/s]
 13%|#3        | 18.1M/137M [00:04<00:31, 3.74Mb/s]
 14%|#3        | 18.5M/137M [00:05<00:31, 3.74Mb/s]
 14%|#3        | 18.9M/137M [00:05<00:32, 3.69Mb/s]
 14%|#4        | 19.3M/137M [00:05<00:33, 3.52Mb/s]
 14%|#4        | 19.6M/137M [00:05<00:39, 2.95Mb/s]
 15%|#4        | 19.9M/137M [00:05<00:41, 2.81Mb/s]
 15%|#4        | 20.3M/137M [00:05<00:39, 2.97Mb/s]
 15%|#5        | 20.7M/137M [00:05<00:36, 3.16Mb/s]
 15%|#5        | 21.0M/137M [00:05<00:35, 3.29Mb/s]
 16%|#5        | 21.4M/137M [00:06<00:33, 3.42Mb/s]
 16%|#5        | 21.8M/137M [00:06<00:33, 3.47Mb/s]
 16%|#6        | 22.2M/137M [00:06<00:32, 3.53Mb/s]
 16%|#6        | 22.5M/137M [00:06<00:31, 3.59Mb/s]
 17%|#6        | 22.9M/137M [00:06<00:31, 3.62Mb/s]
 17%|#7        | 23.3M/137M [00:06<00:31, 3.64Mb/s]
 17%|#7        | 23.7M/137M [00:06<00:31, 3.64Mb/s]
 18%|#7        | 24.1M/137M [00:06<00:30, 3.65Mb/s]
 18%|#7        | 24.4M/137M [00:06<00:30, 3.67Mb/s]
 18%|#8        | 24.8M/137M [00:06<00:30, 3.68Mb/s]
 18%|#8        | 25.2M/137M [00:07<00:30, 3.68Mb/s]
 19%|#8        | 25.6M/137M [00:07<00:30, 3.64Mb/s]
 19%|#8        | 25.9M/137M [00:07<00:30, 3.64Mb/s]
 19%|#9        | 26.3M/137M [00:07<00:30, 3.64Mb/s]
 19%|#9        | 26.7M/137M [00:07<00:30, 3.64Mb/s]
 20%|#9        | 27.1M/137M [00:07<00:30, 3.65Mb/s]
 20%|##        | 27.4M/137M [00:07<00:30, 3.62Mb/s]
 20%|##        | 27.8M/137M [00:07<00:30, 3.63Mb/s]
 21%|##        | 28.2M/137M [00:07<00:30, 3.62Mb/s]
 21%|##        | 28.6M/137M [00:07<00:29, 3.68Mb/s]
 21%|##1       | 28.9M/137M [00:08<00:29, 3.64Mb/s]
 21%|##1       | 29.3M/137M [00:08<00:29, 3.64Mb/s]
 22%|##1       | 29.7M/137M [00:08<00:29, 3.64Mb/s]
 22%|##1       | 30.1M/137M [00:08<00:29, 3.62Mb/s]
 22%|##2       | 30.5M/137M [00:08<00:29, 3.62Mb/s]
 23%|##2       | 30.9M/137M [00:08<00:29, 3.58Mb/s]
 23%|##2       | 31.2M/137M [00:08<00:29, 3.58Mb/s]
```

```
23%|##3          | 31.6M/137M [00:08<00:29, 3.60Mb/s]
23%|##3          | 32.0M/137M [00:08<00:29, 3.61Mb/s]
24%|##3          | 32.4M/137M [00:09<00:28, 3.61Mb/s]
24%|##3          | 32.7M/137M [00:09<00:28, 3.63Mb/s]
24%|##4          | 33.1M/137M [00:09<00:28, 3.59Mb/s]
24%|##4          | 33.5M/137M [00:09<00:28, 3.61Mb/s]
25%|##4          | 34.0M/137M [00:09<00:28, 3.66Mb/s]
25%|##5          | 34.3M/137M [00:09<00:27, 3.67Mb/s]
25%|##5          | 34.7M/137M [00:09<00:27, 3.67Mb/s]
26%|##5          | 35.1M/137M [00:09<00:27, 3.67Mb/s]
26%|##5          | 35.4M/137M [00:09<00:28, 3.60Mb/s]
26%|##6          | 35.8M/137M [00:09<00:28, 3.56Mb/s]
26%|##6          | 36.2M/137M [00:10<00:28, 3.59Mb/s]
27%|##6          | 36.5M/137M [00:10<00:27, 3.62Mb/s]
27%|##6          | 36.9M/137M [00:10<00:27, 3.59Mb/s]
27%|##7          | 37.3M/137M [00:10<00:27, 3.58Mb/s]
28%|##7          | 37.7M/137M [00:10<00:27, 3.62Mb/s]
28%|##7          | 38.0M/137M [00:10<00:27, 3.61Mb/s]
28%|##8          | 38.4M/137M [00:10<00:36, 2.73Mb/s]
28%|##8          | 38.8M/137M [00:10<00:33, 2.93Mb/s]
29%|##8          | 39.2M/137M [00:11<00:31, 3.12Mb/s]
29%|##8          | 39.6M/137M [00:11<00:29, 3.25Mb/s]
29%|##9          | 39.9M/137M [00:11<00:29, 3.34Mb/s]
29%|##9          | 40.4M/137M [00:11<00:27, 3.45Mb/s]
30%|##9          | 40.7M/137M [00:11<00:27, 3.49Mb/s]
30%|###          | 41.1M/137M [00:11<00:27, 3.51Mb/s]
30%|###          | 41.5M/137M [00:11<00:27, 3.53Mb/s]
31%|###          | 41.8M/137M [00:11<00:26, 3.55Mb/s]
31%|###          | 42.2M/137M [00:11<00:26, 3.58Mb/s]
31%|###1         | 42.6M/137M [00:11<00:26, 3.60Mb/s]
31%|###1         | 43.0M/137M [00:12<00:25, 3.62Mb/s]
32%|###1         | 43.3M/137M [00:12<00:25, 3.60Mb/s]
32%|###1         | 43.7M/137M [00:12<00:25, 3.62Mb/s]
32%|###2         | 44.1M/137M [00:12<00:25, 3.64Mb/s]
32%|###2         | 44.5M/137M [00:12<00:25, 3.59Mb/s]
33%|###2         | 44.9M/137M [00:12<00:25, 3.63Mb/s]
33%|###3         | 45.3M/137M [00:12<00:25, 3.63Mb/s]
33%|###3         | 45.7M/137M [00:12<00:25, 3.64Mb/s]
34%|###3         | 46.0M/137M [00:12<00:24, 3.64Mb/s]
34%|###3         | 46.4M/137M [00:13<00:25, 3.59Mb/s]
34%|###4         | 46.8M/137M [00:13<00:24, 3.62Mb/s]
34%|###4         | 47.2M/137M [00:13<00:24, 3.63Mb/s]
35%|###4         | 47.6M/137M [00:13<00:24, 3.64Mb/s]
35%|###5         | 47.9M/137M [00:13<00:24, 3.65Mb/s]
35%|###5         | 48.3M/137M [00:13<00:24, 3.58Mb/s]
36%|###5         | 48.7M/137M [00:13<00:24, 3.59Mb/s]
36%|###5         | 49.1M/137M [00:13<00:24, 3.62Mb/s]
36%|###6         | 49.4M/137M [00:13<00:24, 3.64Mb/s]
36%|###6         | 49.8M/137M [00:13<00:25, 3.47Mb/s]
37%|###6         | 50.2M/137M [00:14<00:24, 3.54Mb/s]
37%|###6         | 50.6M/137M [00:14<00:24, 3.58Mb/s]
37%|###7         | 51.0M/137M [00:14<00:23, 3.60Mb/s]
37%|###7         | 51.3M/137M [00:14<00:23, 3.59Mb/s]
38%|###7         | 51.7M/137M [00:14<00:23, 3.61Mb/s]
38%|###8         | 52.1M/137M [00:14<00:23, 3.59Mb/s]
38%|###8         | 52.4M/137M [00:14<00:23, 3.61Mb/s]
39%|###8         | 52.8M/137M [00:14<00:23, 3.59Mb/s]
39%|###8         | 53.2M/137M [00:14<00:23, 3.60Mb/s]
39%|###9         | 53.6M/137M [00:15<00:22, 3.62Mb/s]
39%|###9         | 53.9M/137M [00:15<00:23, 3.59Mb/s]
```

```
 40%|###9      | 54.3M/137M [00:15<00:22, 3.63Mb/s]
 40%|###9      | 54.7M/137M [00:15<00:22, 3.63Mb/s]
 40%|####      | 55.1M/137M [00:15<00:22, 3.65Mb/s]
 40%|####      | 55.4M/137M [00:15<00:22, 3.66Mb/s]
 41%|####      | 55.8M/137M [00:15<00:22, 3.66Mb/s]
 41%|####1     | 56.2M/137M [00:15<00:22, 3.57Mb/s]
 41%|####1     | 56.6M/137M [00:15<00:22, 3.60Mb/s]
 42%|####1     | 56.9M/137M [00:16<00:27, 2.89Mb/s]
 42%|####1     | 57.3M/137M [00:16<00:29, 2.69Mb/s]
 42%|####2     | 57.6M/137M [00:16<00:27, 2.90Mb/s]
 42%|####2     | 58.0M/137M [00:16<00:25, 3.10Mb/s]
 43%|####2     | 58.3M/137M [00:16<00:24, 3.22Mb/s]
 43%|####2     | 58.7M/137M [00:16<00:23, 3.35Mb/s]
 43%|####3     | 59.1M/137M [00:16<00:22, 3.41Mb/s]
 43%|####3     | 59.5M/137M [00:16<00:22, 3.48Mb/s]
 44%|####3     | 59.8M/137M [00:16<00:21, 3.52Mb/s]
 44%|####3     | 60.2M/137M [00:16<00:21, 3.52Mb/s]
 44%|####4     | 60.6M/137M [00:17<00:21, 3.56Mb/s]
 45%|####4     | 61.0M/137M [00:17<00:20, 3.62Mb/s]
 45%|####4     | 61.4M/137M [00:17<00:20, 3.61Mb/s]
 45%|####5     | 61.8M/137M [00:17<00:20, 3.62Mb/s]
 45%|####5     | 62.1M/137M [00:17<00:20, 3.64Mb/s]
 46%|####5     | 62.5M/137M [00:17<00:21, 3.42Mb/s]
 46%|####5     | 62.9M/137M [00:17<00:20, 3.63Mb/s]
 46%|####6     | 63.3M/137M [00:17<00:20, 3.64Mb/s]
 47%|####6     | 63.7M/137M [00:17<00:20, 3.64Mb/s]
 47%|####6     | 64.0M/137M [00:18<00:20, 3.63Mb/s]
 47%|####7     | 64.4M/137M [00:18<00:19, 3.63Mb/s]
 47%|####7     | 64.8M/137M [00:18<00:20, 3.60Mb/s]
 48%|####7     | 65.3M/137M [00:18<00:19, 3.61Mb/s]
 48%|####7     | 65.6M/137M [00:18<00:19, 3.61Mb/s]
 48%|####8     | 66.0M/137M [00:18<00:19, 3.61Mb/s]
 48%|####8     | 66.4M/137M [00:18<00:19, 3.59Mb/s]
 49%|####8     | 66.7M/137M [00:18<00:19, 3.61Mb/s]
 49%|####9     | 67.1M/137M [00:18<00:19, 3.59Mb/s]
 49%|####9     | 67.5M/137M [00:18<00:19, 3.61Mb/s]
 50%|####9     | 67.9M/137M [00:19<00:19, 3.59Mb/s]
 50%|####9     | 68.2M/137M [00:19<00:19, 3.60Mb/s]
 50%|#####     | 68.6M/137M [00:19<00:18, 3.62Mb/s]
 50%|#####     | 69.0M/137M [00:19<00:18, 3.60Mb/s]
 51%|#####     | 69.3M/137M [00:19<00:18, 3.62Mb/s]
 51%|#####     | 69.7M/137M [00:19<00:18, 3.58Mb/s]
 51%|#####1    | 70.1M/137M [00:19<00:18, 3.61Mb/s]
 51%|#####1    | 70.4M/137M [00:19<00:18, 3.59Mb/s]
 52%|#####1    | 70.8M/137M [00:19<00:18, 3.59Mb/s]
 52%|#####1    | 71.2M/137M [00:20<00:18, 3.57Mb/s]
 52%|#####2    | 71.6M/137M [00:20<00:18, 3.60Mb/s]
 53%|#####2    | 71.9M/137M [00:20<00:18, 3.57Mb/s]
 53%|#####2    | 72.3M/137M [00:20<00:17, 3.60Mb/s]
 53%|#####3    | 72.7M/137M [00:20<00:17, 3.57Mb/s]
 53%|#####3    | 73.0M/137M [00:20<00:17, 3.60Mb/s]
 54%|#####3    | 73.4M/137M [00:20<00:17, 3.57Mb/s]
 54%|#####3    | 73.8M/137M [00:20<00:17, 3.59Mb/s]
 54%|#####4    | 74.1M/137M [00:20<00:17, 3.60Mb/s]
 54%|#####4    | 74.5M/137M [00:20<00:17, 3.57Mb/s]
 55%|#####4    | 74.9M/137M [00:21<00:17, 3.58Mb/s]
 55%|#####4    | 75.2M/137M [00:21<00:18, 3.39Mb/s]
 55%|#####5    | 75.6M/137M [00:21<00:22, 2.71Mb/s]
 55%|#####5    | 75.9M/137M [00:21<00:20, 2.91Mb/s]
 56%|#####5    | 76.3M/137M [00:21<00:19, 3.10Mb/s]
```

```
56%|#####6    | 76.7M/137M [00:21<00:18, 3.23Mb/s]
56%|#####6    | 77.0M/137M [00:21<00:17, 3.35Mb/s]
57%|#####6    | 77.4M/137M [00:21<00:17, 3.41Mb/s]
57%|#####6    | 77.8M/137M [00:21<00:17, 3.46Mb/s]
57%|#####7    | 78.1M/137M [00:22<00:16, 3.48Mb/s]
57%|#####7    | 78.5M/137M [00:22<00:16, 3.51Mb/s]
58%|#####7    | 78.8M/137M [00:22<00:16, 3.56Mb/s]
58%|#####7    | 79.2M/137M [00:22<00:16, 3.55Mb/s]
58%|#####8    | 79.6M/137M [00:22<00:16, 3.55Mb/s]
58%|#####8    | 79.9M/137M [00:22<00:16, 3.56Mb/s]
59%|#####8    | 80.3M/137M [00:22<00:15, 3.58Mb/s]
59%|#####8    | 80.7M/137M [00:22<00:15, 3.58Mb/s]
59%|#####9    | 81.0M/137M [00:22<00:15, 3.61Mb/s]
59%|#####9    | 81.4M/137M [00:22<00:15, 3.63Mb/s]
60%|#####9    | 81.8M/137M [00:23<00:15, 3.61Mb/s]
60%|######    | 82.2M/137M [00:23<00:15, 3.61Mb/s]
60%|######    | 82.5M/137M [00:23<00:15, 3.62Mb/s]
61%|######    | 82.9M/137M [00:23<00:14, 3.60Mb/s]
61%|######    | 83.3M/137M [00:23<00:14, 3.61Mb/s]
61%|######1   | 83.7M/137M [00:23<00:14, 3.63Mb/s]
61%|######1   | 84.0M/137M [00:23<00:14, 3.64Mb/s]
62%|######1   | 84.4M/137M [00:23<00:14, 3.61Mb/s]
62%|######1   | 84.8M/137M [00:23<00:14, 3.63Mb/s]
62%|######2   | 85.2M/137M [00:24<00:14, 3.63Mb/s]
62%|######2   | 85.5M/137M [00:24<00:14, 3.59Mb/s]
63%|######2   | 85.9M/137M [00:24<00:14, 3.61Mb/s]
63%|######3   | 86.3M/137M [00:24<00:13, 3.62Mb/s]
63%|######3   | 86.7M/137M [00:24<00:13, 3.63Mb/s]
64%|######3   | 87.0M/137M [00:24<00:13, 3.64Mb/s]
64%|######3   | 87.4M/137M [00:24<00:13, 3.60Mb/s]
64%|######4   | 87.8M/137M [00:24<00:13, 3.61Mb/s]
64%|######4   | 88.1M/137M [00:24<00:13, 3.58Mb/s]
65%|######4   | 88.5M/137M [00:24<00:13, 3.60Mb/s]
65%|######4   | 88.9M/137M [00:25<00:13, 3.62Mb/s]
65%|######5   | 89.3M/137M [00:25<00:13, 3.62Mb/s]
65%|######5   | 89.6M/137M [00:25<00:13, 3.60Mb/s]
66%|######5   | 90.0M/137M [00:25<00:12, 3.62Mb/s]
66%|######6   | 90.4M/137M [00:25<00:12, 3.59Mb/s]
66%|######6   | 90.7M/137M [00:25<00:12, 3.61Mb/s]
67%|######6   | 91.1M/137M [00:25<00:12, 3.59Mb/s]
67%|######6   | 91.5M/137M [00:25<00:12, 3.60Mb/s]
67%|######7   | 91.9M/137M [00:25<00:12, 3.61Mb/s]
67%|######7   | 92.2M/137M [00:25<00:12, 3.59Mb/s]
68%|######7   | 92.6M/137M [00:26<00:12, 3.62Mb/s]
68%|######7   | 93.0M/137M [00:26<00:12, 3.59Mb/s]
68%|######8   | 93.3M/137M [00:26<00:12, 3.62Mb/s]
68%|######8   | 93.7M/137M [00:26<00:12, 3.58Mb/s]
69%|######8   | 94.1M/137M [00:26<00:13, 3.17Mb/s]
69%|######8   | 94.4M/137M [00:26<00:15, 2.75Mb/s]
69%|######9   | 94.8M/137M [00:26<00:14, 2.97Mb/s]
69%|######9   | 95.2M/137M [00:26<00:13, 3.15Mb/s]
70%|######9   | 95.5M/137M [00:27<00:12, 3.26Mb/s]
70%|#######   | 95.9M/137M [00:27<00:12, 3.38Mb/s]
70%|#######   | 96.2M/137M [00:27<00:11, 3.44Mb/s]
71%|#######   | 96.6M/137M [00:27<00:11, 3.45Mb/s]
71%|#######   | 97.0M/137M [00:27<00:11, 3.52Mb/s]
71%|#######1  | 97.4M/137M [00:27<00:11, 3.55Mb/s]
71%|#######1  | 97.7M/137M [00:27<00:11, 3.55Mb/s]
72%|#######1  | 98.1M/137M [00:27<00:10, 3.58Mb/s]
72%|#######1  | 98.5M/137M [00:27<00:10, 3.61Mb/s]
```

```
72%|#######2  | 98.9M/137M [00:27<00:10, 3.62Mb/s]
72%|#######2  | 99.2M/137M [00:28<00:10, 3.63Mb/s]
73%|#######2  | 99.6M/137M [00:28<00:10, 3.61Mb/s]
73%|#######3  | 100M/137M [00:28<00:10, 3.63Mb/s]
73%|#######3  | 100M/137M [00:28<00:10, 3.63Mb/s]
74%|#######3  | 101M/137M [00:28<00:09, 3.63Mb/s]
74%|#######3  | 101M/137M [00:28<00:09, 3.66Mb/s]
74%|#######4  | 101M/137M [00:28<00:09, 3.62Mb/s]
74%|#######4  | 102M/137M [00:28<00:09, 3.64Mb/s]
75%|#######4  | 102M/137M [00:28<00:09, 3.61Mb/s]
75%|#######4  | 103M/137M [00:28<00:09, 3.63Mb/s]
75%|#######5  | 103M/137M [00:29<00:09, 3.60Mb/s]
75%|#######5  | 103M/137M [00:29<00:09, 3.61Mb/s]
76%|#######5  | 104M/137M [00:29<00:09, 3.63Mb/s]
76%|#######6  | 104M/137M [00:29<00:09, 3.60Mb/s]
76%|#######6  | 104M/137M [00:29<00:08, 3.62Mb/s]
77%|#######6  | 105M/137M [00:29<00:08, 3.60Mb/s]
77%|#######6  | 105M/137M [00:29<00:08, 3.61Mb/s]
77%|#######7  | 106M/137M [00:29<00:08, 3.60Mb/s]
77%|#######7  | 106M/137M [00:29<00:08, 3.63Mb/s]
78%|#######7  | 106M/137M [00:29<00:08, 3.60Mb/s]
78%|#######7  | 107M/137M [00:30<00:08, 3.62Mb/s]
78%|#######8  | 107M/137M [00:30<00:08, 3.59Mb/s]
78%|#######8  | 107M/137M [00:30<00:08, 3.60Mb/s]
79%|#######8  | 108M/137M [00:30<00:08, 3.62Mb/s]
79%|#######8  | 108M/137M [00:30<00:07, 3.60Mb/s]
79%|#######9  | 109M/137M [00:30<00:07, 3.61Mb/s]
80%|#######9  | 109M/137M [00:30<00:07, 3.61Mb/s]
80%|#######9  | 109M/137M [00:30<00:07, 3.59Mb/s]
80%|########  | 110M/137M [00:30<00:07, 3.61Mb/s]
80%|########  | 110M/137M [00:31<00:07, 3.61Mb/s]
81%|########  | 110M/137M [00:31<00:07, 3.63Mb/s]
81%|########  | 111M/137M [00:31<00:07, 3.61Mb/s]
81%|########1 | 111M/137M [00:31<00:07, 3.63Mb/s]
81%|########1 | 112M/137M [00:31<00:07, 3.60Mb/s]
82%|########1 | 112M/137M [00:31<00:06, 3.60Mb/s]
82%|########2 | 112M/137M [00:31<00:06, 3.62Mb/s]
82%|########2 | 113M/137M [00:31<00:08, 2.91Mb/s]
83%|########2 | 113M/137M [00:31<00:08, 2.75Mb/s]
83%|########2 | 113M/137M [00:32<00:07, 2.95Mb/s]
83%|########3 | 114M/137M [00:32<00:07, 3.14Mb/s]
83%|########3 | 114M/137M [00:32<00:07, 3.25Mb/s]
84%|########3 | 114M/137M [00:32<00:06, 3.36Mb/s]
84%|########3 | 115M/137M [00:32<00:06, 3.44Mb/s]
84%|########4 | 115M/137M [00:32<00:06, 3.47Mb/s]
84%|########4 | 116M/137M [00:32<00:06, 3.52Mb/s]
85%|########4 | 116M/137M [00:32<00:05, 3.53Mb/s]
85%|########4 | 116M/137M [00:32<00:05, 3.57Mb/s]
85%|########5 | 117M/137M [00:32<00:05, 3.60Mb/s]
85%|########5 | 117M/137M [00:33<00:05, 3.58Mb/s]
86%|########5 | 117M/137M [00:33<00:05, 3.60Mb/s]
86%|########6 | 118M/137M [00:33<00:05, 3.60Mb/s]
86%|########6 | 118M/137M [00:33<00:05, 3.63Mb/s]
87%|########6 | 119M/137M [00:33<00:05, 3.61Mb/s]
87%|########6 | 119M/137M [00:33<00:04, 3.63Mb/s]
87%|########7 | 119M/137M [00:33<00:04, 3.61Mb/s]
87%|########7 | 120M/137M [00:33<00:04, 3.64Mb/s]
88%|########7 | 120M/137M [00:33<00:04, 3.64Mb/s]
88%|########7 | 120M/137M [00:34<00:04, 3.65Mb/s]
88%|########8 | 121M/137M [00:34<00:04, 3.66Mb/s]
```

```
 88%|########8 | 121M/137M [00:34<00:04, 3.63Mb/s]
 89%|########8 | 122M/137M [00:34<00:04, 3.64Mb/s]
 89%|########9 | 122M/137M [00:34<00:04, 3.65Mb/s]
 89%|########9 | 122M/137M [00:34<00:04, 3.65Mb/s]
 90%|########9 | 123M/137M [00:34<00:03, 3.68Mb/s]
 90%|########9 | 123M/137M [00:34<00:03, 3.63Mb/s]
 90%|######### | 123M/137M [00:34<00:03, 3.64Mb/s]
 90%|######### | 124M/137M [00:34<00:03, 3.61Mb/s]
 91%|######### | 124M/137M [00:35<00:03, 3.63Mb/s]
 91%|######### | 125M/137M [00:35<00:03, 3.60Mb/s]
 91%|#########1| 125M/137M [00:35<00:03, 3.62Mb/s]
 91%|#########1| 125M/137M [00:35<00:03, 3.63Mb/s]
 92%|#########1| 126M/137M [00:35<00:03, 3.59Mb/s]
 92%|#########2| 126M/137M [00:35<00:03, 3.60Mb/s]
 92%|#########2| 126M/137M [00:35<00:02, 3.62Mb/s]
 93%|#########2| 127M/137M [00:35<00:03, 3.22Mb/s]
 93%|#########2| 127M/137M [00:35<00:02, 3.72Mb/s]
 93%|#########3| 128M/137M [00:36<00:02, 3.72Mb/s]
 94%|#########3| 128M/137M [00:36<00:02, 3.68Mb/s]
 94%|#########3| 128M/137M [00:36<00:02, 3.66Mb/s]
 94%|#########4| 129M/137M [00:36<00:02, 3.65Mb/s]
 94%|#########4| 129M/137M [00:36<00:02, 3.61Mb/s]
 95%|#########4| 130M/137M [00:36<00:02, 3.61Mb/s]
 95%|#########4| 130M/137M [00:36<00:01, 3.59Mb/s]
 95%|#########5| 130M/137M [00:36<00:01, 3.61Mb/s]
 95%|#########5| 131M/137M [00:36<00:01, 3.62Mb/s]
 96%|#########5| 131M/137M [00:36<00:01, 3.59Mb/s]
 96%|#########5| 131M/137M [00:37<00:01, 3.02Mb/s]
 96%|#########6| 132M/137M [00:37<00:01, 2.81Mb/s]
 96%|#########6| 132M/137M [00:37<00:01, 3.00Mb/s]
 97%|#########6| 132M/137M [00:37<00:01, 3.15Mb/s]
 97%|#########7| 133M/137M [00:37<00:01, 3.29Mb/s]
 97%|#########7| 133M/137M [00:37<00:01, 3.37Mb/s]
 98%|#########7| 134M/137M [00:37<00:00, 3.44Mb/s]
 98%|#########7| 134M/137M [00:37<00:00, 3.49Mb/s]
 98%|#########8| 134M/137M [00:37<00:00, 3.54Mb/s]
 98%|#########8| 135M/137M [00:38<00:00, 3.55Mb/s]
 99%|#########8| 135M/137M [00:38<00:00, 3.58Mb/s]
 99%|#########8| 135M/137M [00:38<00:00, 3.59Mb/s]
 99%|#########9| 136M/137M [00:38<00:00, 3.59Mb/s]
 99%|#########9| 136M/137M [00:38<00:00, 3.62Mb/s]
100%|#########9| 137M/137M [00:38<00:00, 3.63Mb/s]
100%|##########| 137M/137M [00:38<00:00, 3.54Mb/s]
[INFO] Beginning extraction
[INFO] Chromium extracted to: C:\Users\LENOVO\AppData\Local\pyppeteer\pyppeteer\local-ch
romium\588429
```

In [ ]:

```

```