OLA BIKE RIDE REQUEST FORECAST

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

In [2]:
```
train_df = pd.read_csv(r'C:\Users\lenovo\Downloads\train.csv')
```

In [3]:
```
train_df
```

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
0	01-01-2011 00:00	1	0	0	1	9.84	14.395	81	0.0000	3	
1	01-01-2011 01:00	1	0	0	1	9.02	13.635	80	0.0000	8	
2	01-01-2011 02:00	1	0	0	1	9.02	13.635	80	0.0000	5	
3	01-01-2011 03:00	1	0	0	1	9.84	14.395	75	0.0000	3	
4	01-01-2011 04:00	1	0	0	1	9.84	14.395	75	0.0000	0	
...	...	...	...	...	...	...	...	...	...	...	...
10881	19-12-2012 19:00	4	0	1	1	15.58	19.695	50	26.0027	7	
10882	19-12-2012 20:00	4	0	1	1	14.76	17.425	57	15.0013	10	
10883	19-12-2012 21:00	4	0	1	1	13.94	15.910	61	15.0013	4	
10884	19-12-2012 22:00	4	0	1	1	13.94	17.425	61	6.0032	12	
10885	19-12-2012	4	0	1	1	13.12	16.665	66	8.9981	4	

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
	23:00										

10886 rows × 12 columns

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

Out[4]:
```
(10886, 12)
```

In [5]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   datetime    10886 non-null  object 
 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  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

Dealing with null values

In [6]:
```
train_df.isnull().sum()
```

Out[6]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

In [7]:
```
train_df.columns
```

Out[7]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [8]:
```
train_df.dtypes
```

Out[8]:
```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

Changing into Datetime format

In [9]:
```
def change_into_datetime(col):
    train_df[col] = pd.to_datetime(train_df[col])
```

In [10]:
```
for i in ['datetime']:
    change_into_datetime(i)
```

In [11]:
```
train_df['year'] = train_df['datetime'].dt.year
train_df['month'] = train_df['datetime'].dt.month
train_df['date'] = train_df['datetime'].dt.date
train_df['hour'] = train_df['datetime'].dt.hour
train_df['day of the week'] = train_df['datetime'].dt.dayofweek
```

In [12]:
```
train_df['year'].unique()
```

Out[12]:
```
array([2011, 2012], dtype=int64)
```

Mapping key values to Dictionary

In [13]:
```
dict = {2011 :'1', 2012 :'2'}
```

In [14]:
```
train_df['year'] = train_df['year'].map(dict)
train_df['year'].unique()
```

Out[14]:
```
array(['1', '2'], dtype=object)
```

In [15]:
```
train_df.drop('datetime',axis=1,inplace=True)
```

In [16]:
```
train_df.head()
```

Out[16]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	coun
0	1	0	0	1	9.84	14.395	81	0.0	3	13	1
1	1	0	0	1	9.02	13.635	80	0.0	8	32	4
2	1	0	0	1	9.02	13.635	80	0.0	5	27	3
3	1	0	0	1	9.84	14.395	75	0.0	3	10	1
4	1	0	0	1	9.84	14.395	75	0.0	0	1	

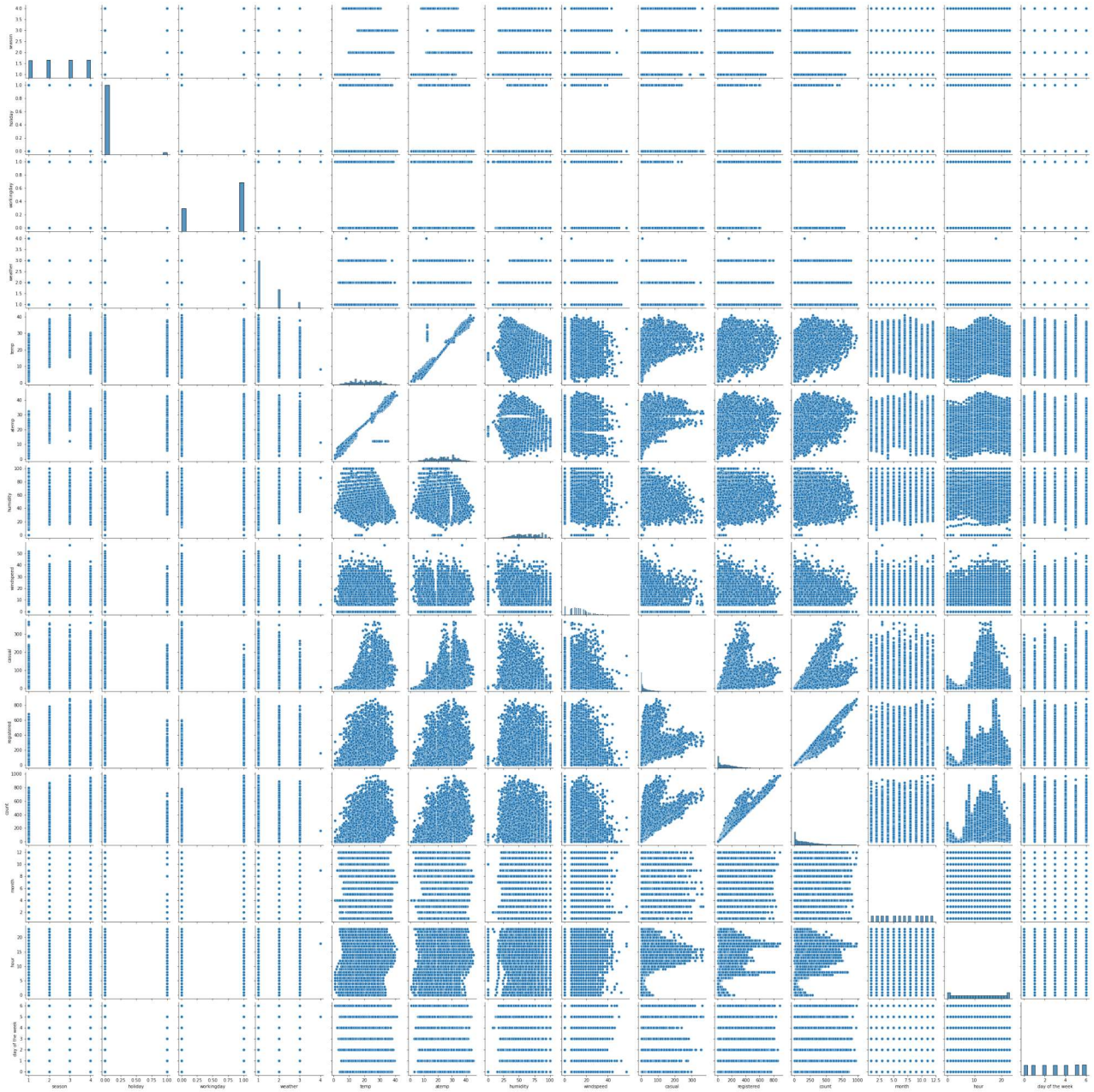
Performing Exploratory Data Analysis

In [17]:
```
train_df.drop('count',axis=1).describe()
```

Out[17]:

	season	holiday	workingday	weather	temp	atemp	humidity
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000

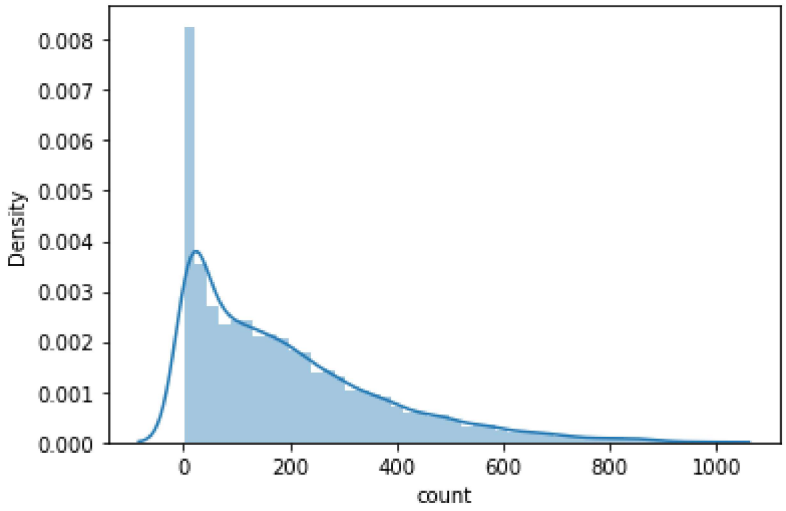
In [18]:
```
sns.pairplot(train_df);
```



Distribution of target variable 'count'

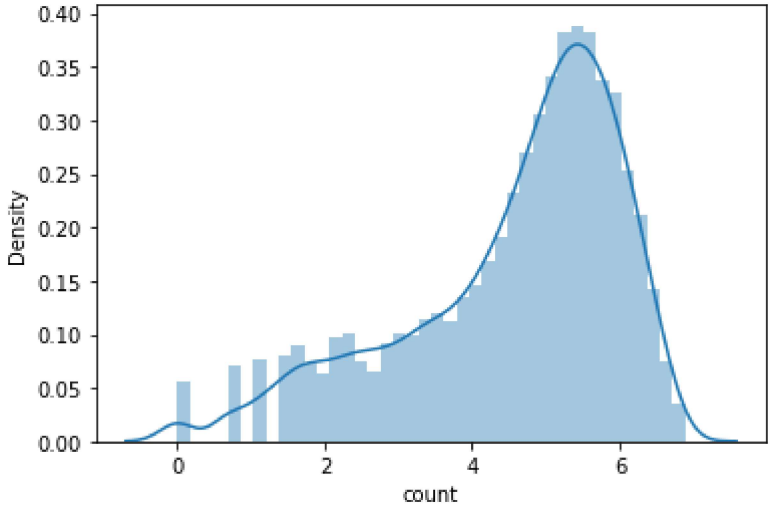
In [19]:
```
sns.distplot(train_df['count']);
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [20]:
```
sns.distplot(np.log(train_df['count']));
```

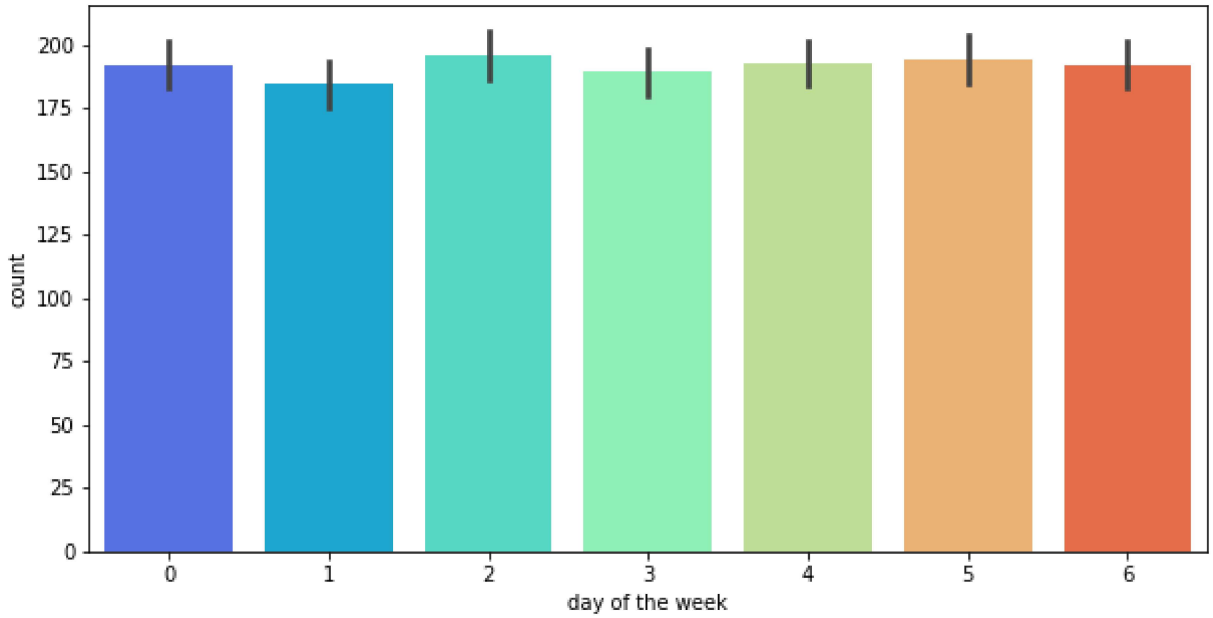
```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Visualizing the Demand using multiple variables

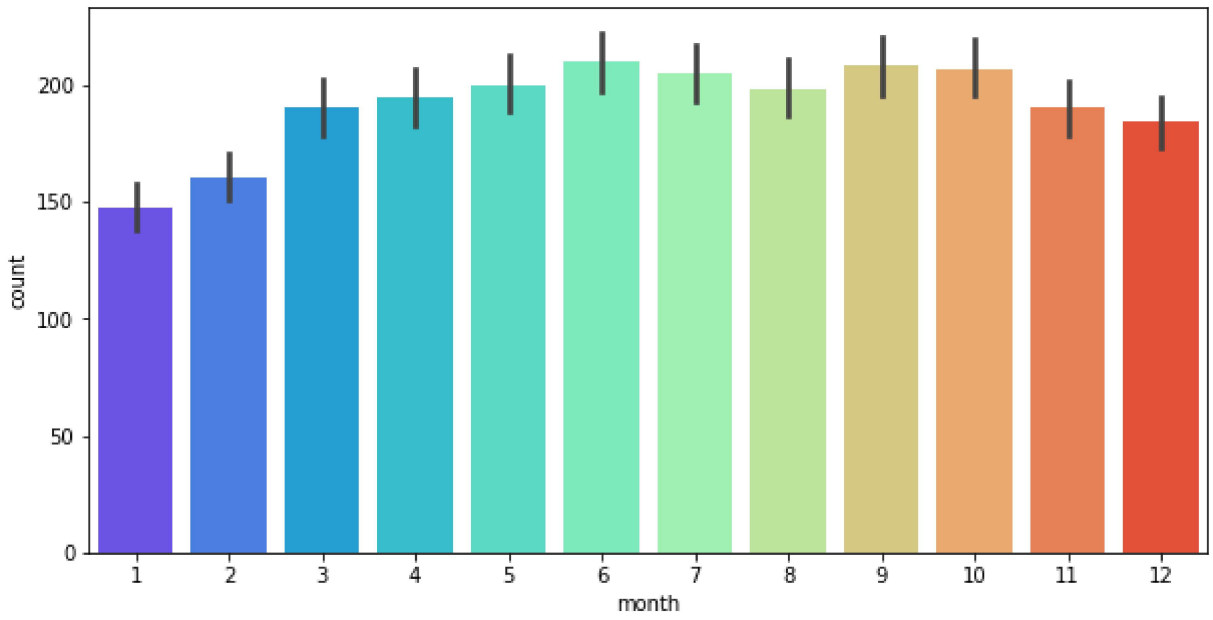
Demand per day of the week

In [21]:
```
plt.figure(figsize=(10,5))
sns.barplot(data=train_df, x = 'day of the week', y = 'count', palette='rainbow');
```



Inference - demand of bike rentals were almost same for each day of the week. So this feature will not be useful in predicting the demand therefore we will have to drop this feature.

In [22]:
```
plt.figure(figsize=(10,5))
sns.barplot(data=train_df, x='month', y = 'count', palette = 'rainbow');
```



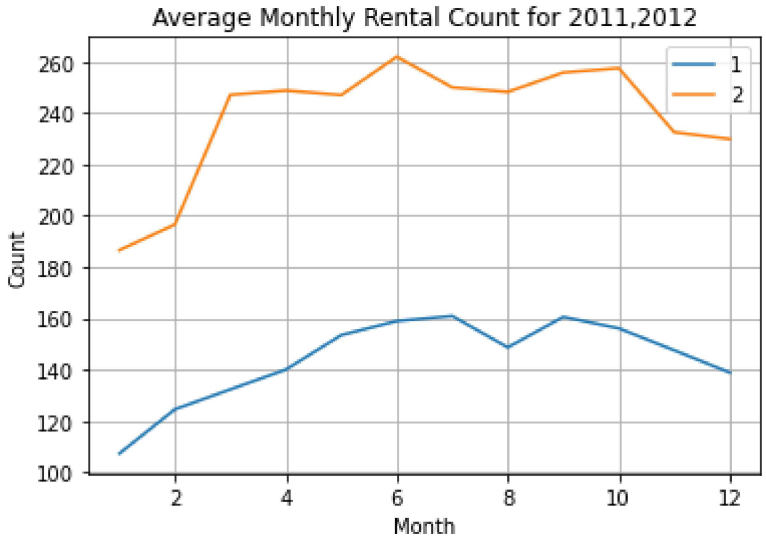
Inference - Demand for bike rentals was high during the months of summer and the demand drops during the months of winter.

Year Month Demand

In [23]:
```
group_year_month = train_df.groupby(['year','month'])
average_year_month = group_year_month['count'].mean()
```

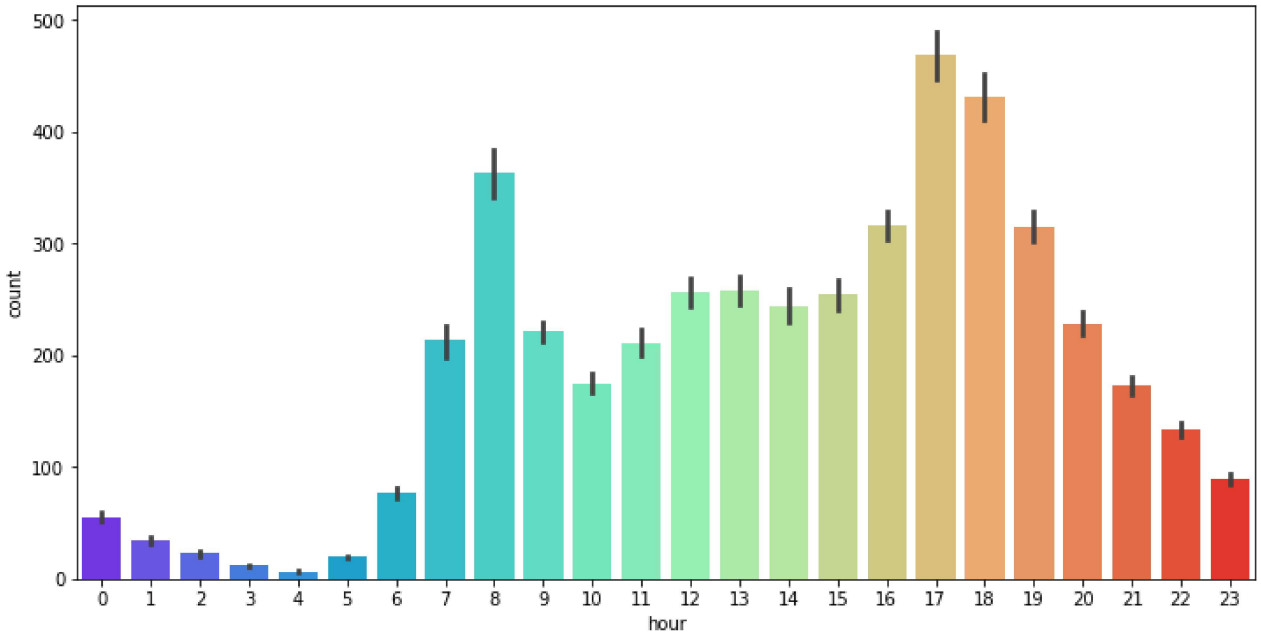
In [24]:
```
for year in average_year_month.index.levels[0]:
    plt.plot(average_year_month[year].index,average_year_month[year],label=year)

plt.legend()
plt.xlabel('Month')
plt.ylabel('Count')
plt.grid(True)
plt.title('Average Monthly Rental Count for 2011,2012')
plt.show()
```



Demand per hour

In [25]:
```
plt.figure(figsize=(12,6))
sns.barplot(data = train_df, x = 'hour', y = 'count', palette = 'rainbow');
```

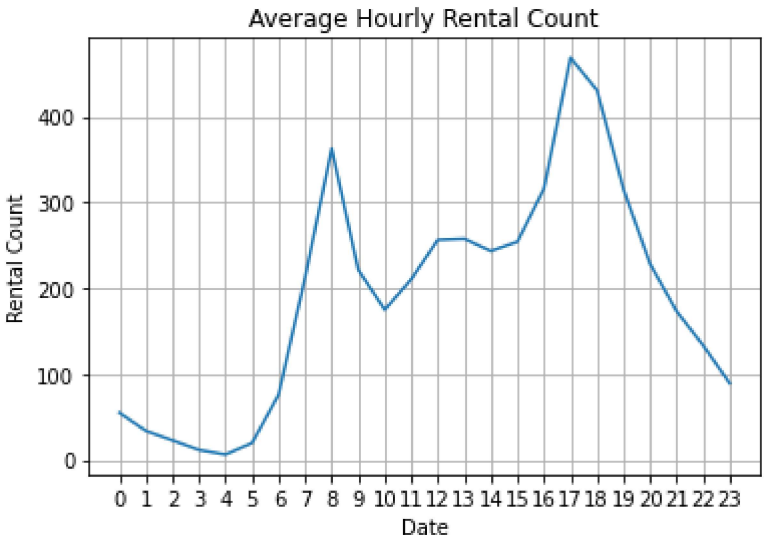


Inference - Demand for bike rentals were high during the office hours and it was low during the non-working hours as it might be possible that most of the people use the bike rental services to reach their office premises or leave their premises during these hours.

Average Hourly Count

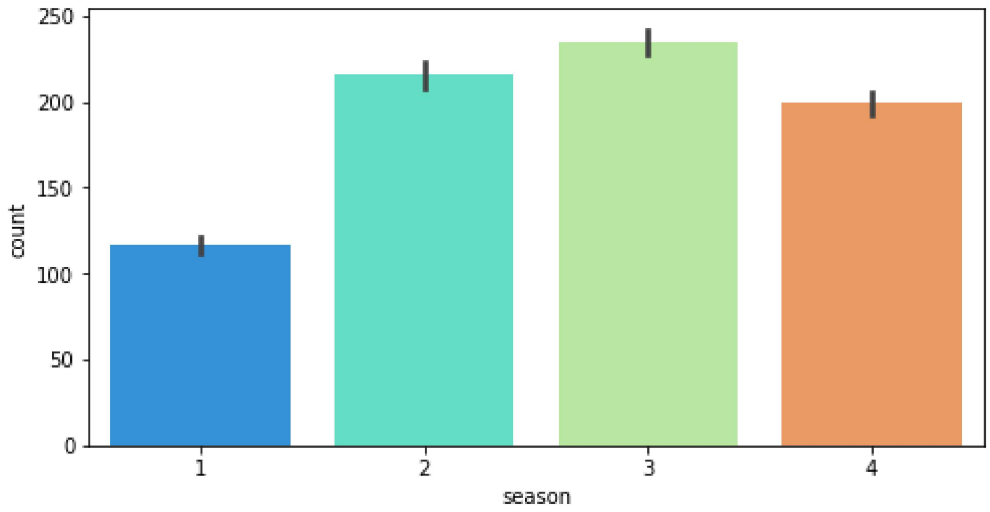
In [27]:
```
group_hour = train_df.groupby(['hour'])
average_by_hour = group_hour['count'].mean()
```

In [28]:
```
plt.plot(average_by_hour.index,average_by_hour)
plt.xlabel('Date')
plt.ylabel('Rental Count')
plt.xticks(np.arange(24))
plt.grid(True)
plt.title('Average Hourly Rental Count')
plt.show()
```



Demand per season

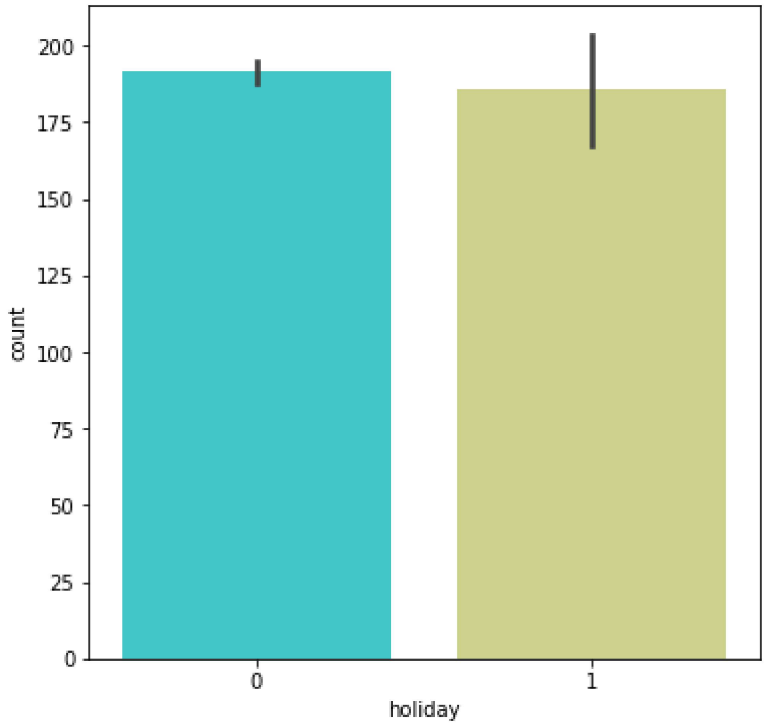
In [29]:
```
plt.figure(figsize = (8,4))
sns.barplot(data = train_df, x = 'season', y = 'count', palette = 'rainbow');
```



Inference - Demand was high during the summer and fall seasons while it drops during winter and spring season as the weather during these seasons might not be suitable for bike rentals.

Demand as per holidays

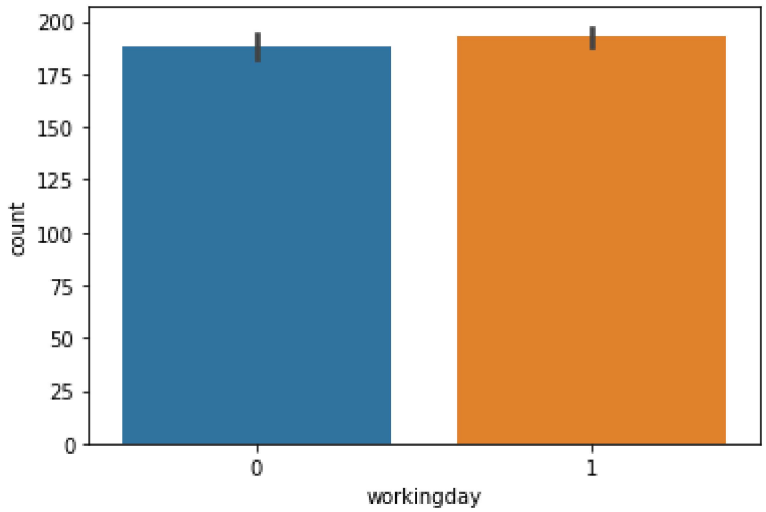
In [30]:
```
plt.figure(figsize=(6,6))
sns.barplot(data = train_df, x = 'holiday', y = 'count', palette = 'rainbow');
```



Inference - Demand was high during working days and low during holidays.

Demand on working days

In [31]:
```
plt.figure(figsize=(6,4))
sns.barplot(data = train_df, x = 'workingday', y = 'count');
```



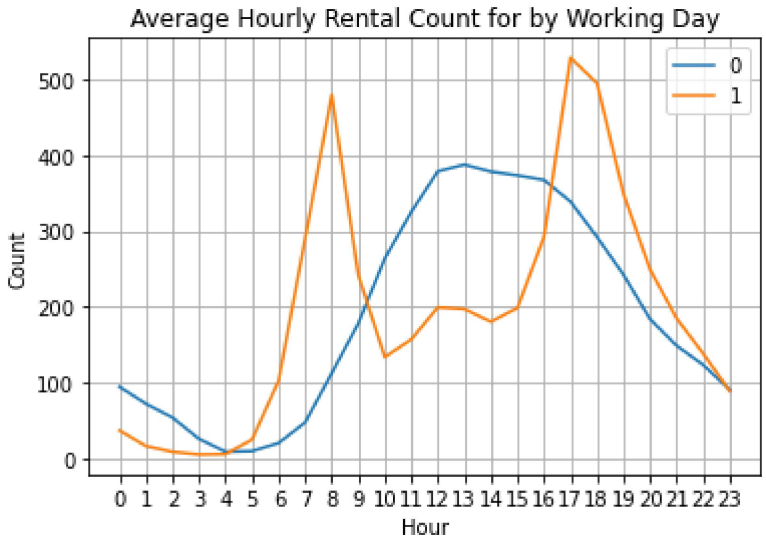
Inference - It doesn't affect the demand much as it remains almost the same both the time, therefore we will have to drop this column.

Working day Hour Demand

In [32]:
```
group_workingday_hour = train_df.groupby(['workingday','hour'])
average_workingday_hour = group_workingday_hour['count'].mean()
```

In [33]:
```
for workingday in average_workingday_hour.index.levels[0]:
    plt.plot(average_workingday_hour[workingday].index,average_workingday_hour[workingd

plt.legend()
plt.xlabel('Hour')
plt.ylabel('Count')
plt.xticks(np.arange(24))
plt.grid(True)
plt.title('Average Hourly Rental Count for by Working Day')
plt.show()
```

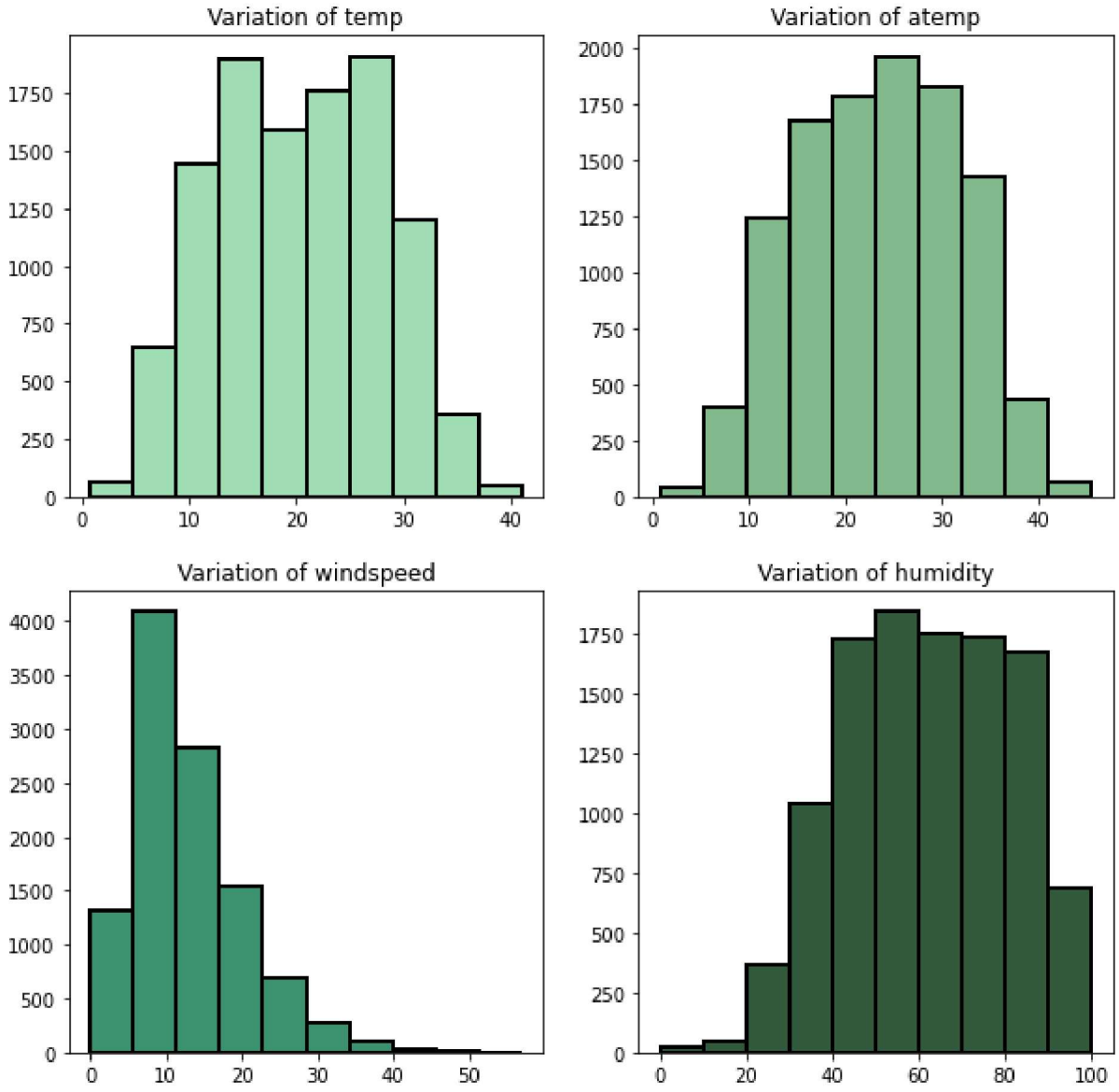


Visualization of the continuous varibles using Histogram

Continuous features - temp, atemp, humidity, windspeed, count

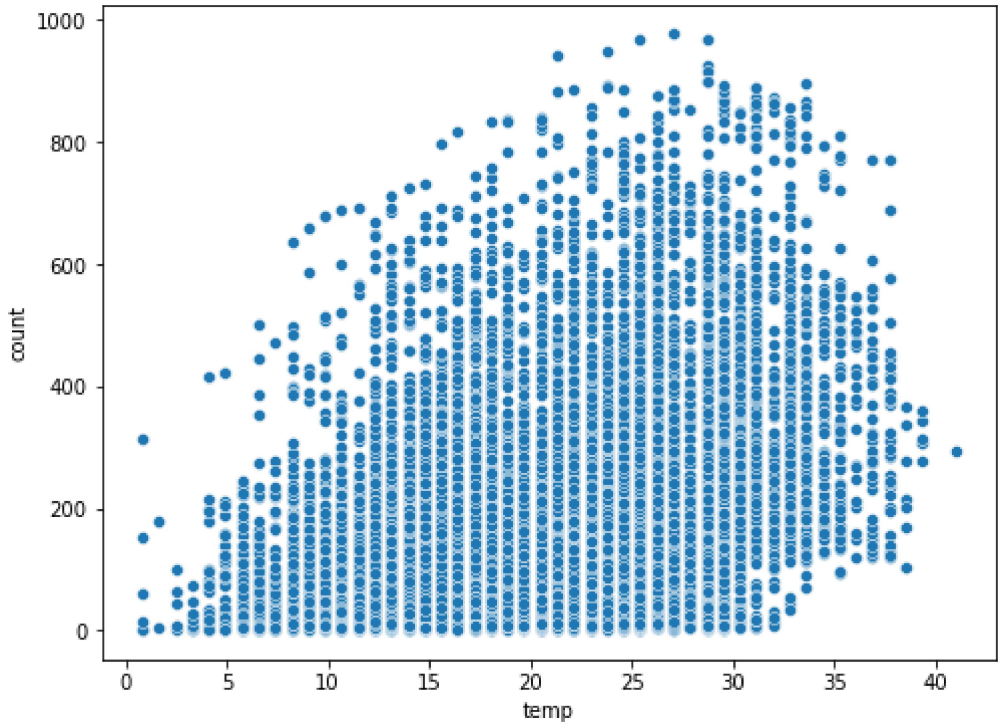
In [34]:

```
train_df.temp.unique()
fig,axes=plt.subplots(2,2)
axes[0,0].hist(x="temp",data=train_df,edgecolor="black",linewidth=2,color='#a0e0b6')
axes[0,0].set_title("Variation of temp")
axes[0,1].hist(x="atemp",data=train_df,edgecolor="black",linewidth=2,color='#85b98d')
axes[0,1].set_title("Variation of atemp")
axes[1,0].hist(x="windspeed",data=train_df,edgecolor="black",linewidth=2,color='#3e936e
axes[1,0].set_title("Variation of windspeed")
axes[1,1].hist(x="humidity",data=train_df,edgecolor="black",linewidth=2,color='#335d40'
axes[1,1].set_title("Variation of humidity")
fig.set_size_inches(10,10)
```



Temperature vs Demand

In [35]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data = train_df, x = 'temp', y = 'count');
```



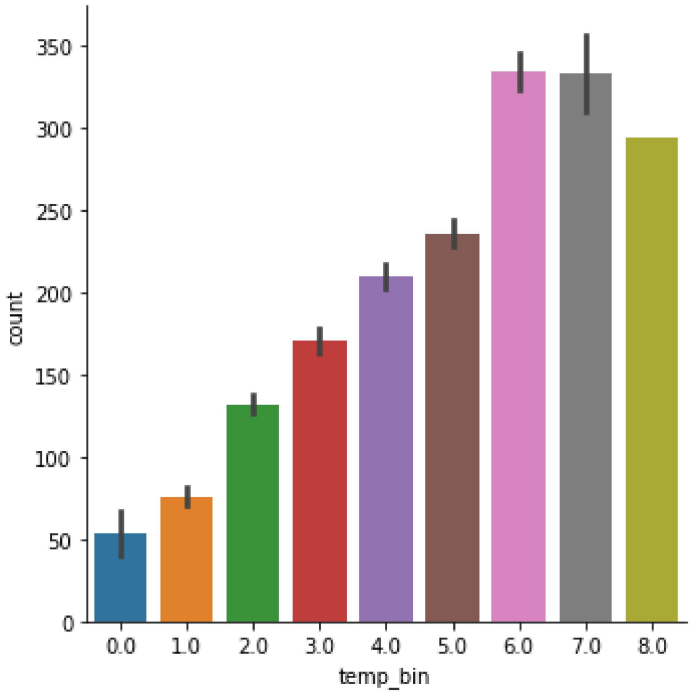
Since this is hard to visualze. a better way is to convert the 'temp' variable into intervals or so called bins and then treat it like a discrete variable.

In [36]:
```
viz_df=train_df.copy()
viz_df.temp.describe()
viz_df['temp_bin']=np.floor(viz_df['temp'])//5
viz_df['temp_bin'].unique()

# now we can visualize with temp as discrete values
sns.factorplot(x="temp_bin",y="count",data=viz_df,kind='bar')
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: Th
e `factorplot` function has been renamed to `catplot`. The original name will be removed
in a future release. Please update your code. Note that the default `kind` in `factorplo
t` (``'point'``) has changed ``'strip'`` in `catplot`.
  warnings.warn(msg)
```

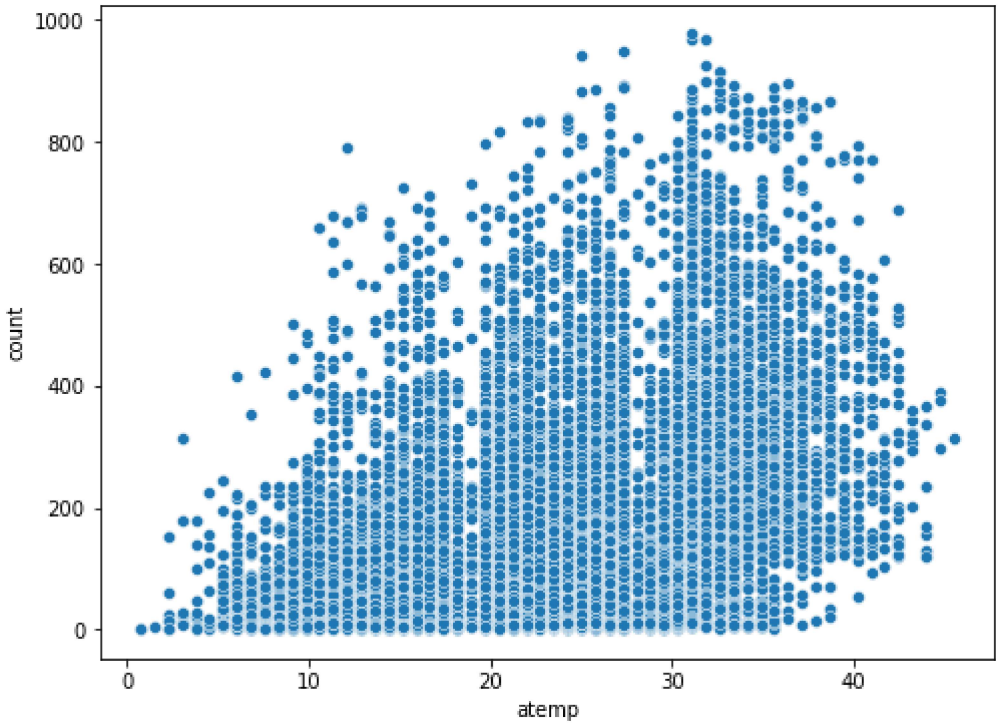
Out[36]:
```
<seaborn.axisgrid.FacetGrid at 0x18f09b9d940>
```



Inference - The demand is highest for bins 6 and 7 which is about tempearure 30-35(bin 6) and 35-40 (bin 7).

Demand vs aTemp

In [37]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data = train_df, x = 'atemp', y = 'count');
```



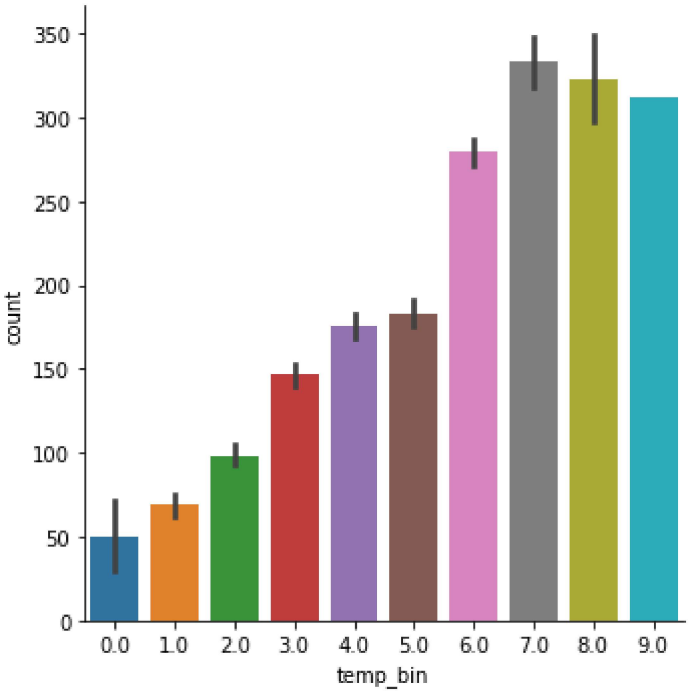
In [38]:
```
viz_df=train_df.copy()
viz_df.temp.describe()
viz_df['temp_bin']=np.floor(viz_df['atemp'])//5
```

```
viz_df['temp_bin'].unique()

# now we can visualize with temp as discrete values
sns.factorplot(x="temp_bin",y="count",data=viz_df,kind='bar')
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: Th
e `factorplot` function has been renamed to `catplot`. The original name will be removed
in a future release. Please update your code. Note that the default `kind` in `factorplo
t` (``'point'``) has changed ``'strip'`` in `catplot`.
  warnings.warn(msg)
```

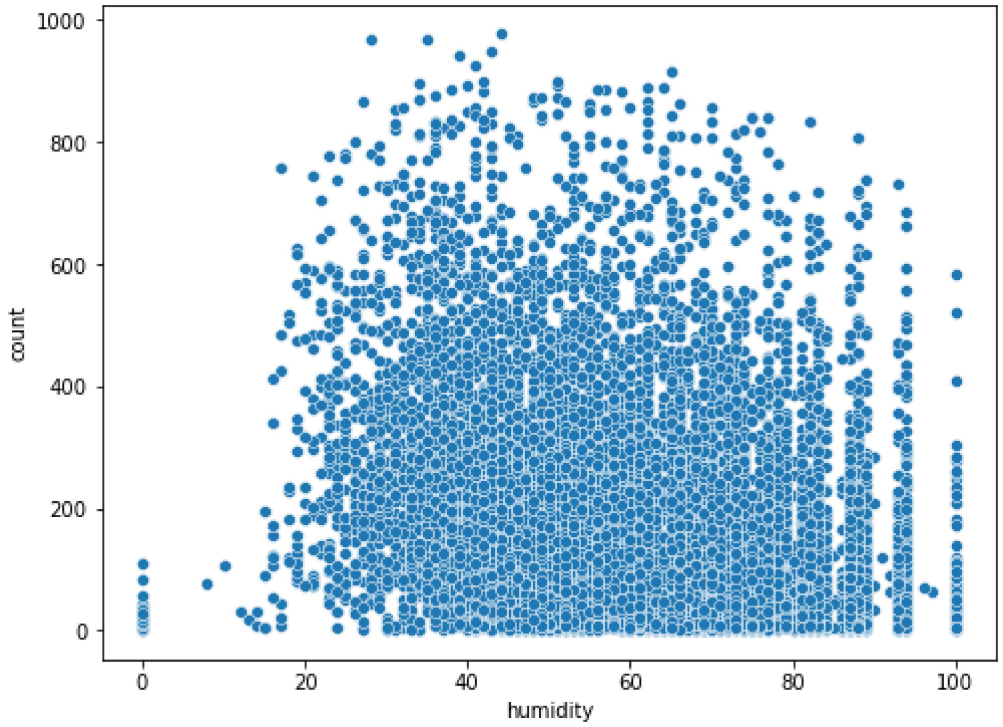
Out[38]: <seaborn.axisgrid.FacetGrid at 0x18f0ed39940>



Inference - We can see that this plot is almost similar to the previous plot of temperature and demand which means there is a high correlation between temp and atemp features.

Humidity vs Demand

In [39]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data = train_df, x = 'humidity', y='count');
```



In [41]:

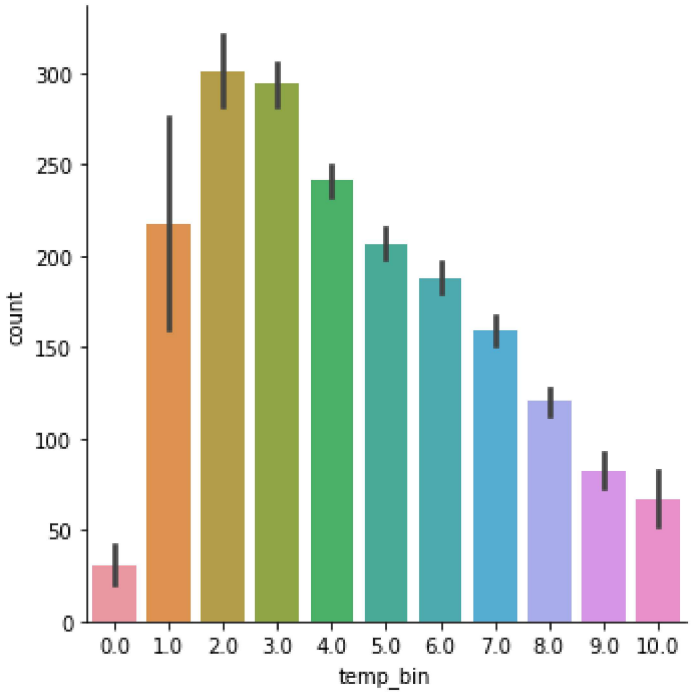
```
viz_df=train_df.copy()
viz_df.temp.describe()
viz_df['temp_bin']=np.floor(viz_df['humidity'])//10
viz_df['temp_bin'].unique()

# now we can visualize with temp as discrete values
sns.factorplot(x="temp_bin",y="count",data=viz_df,kind='bar')
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: Th
e `factorplot` function has been renamed to `catplot`. The original name will be removed
in a future release. Please update your code. Note that the default `kind` in `factorplo
t` (``'point'``) has changed ``'strip'`` in `catplot`.
  warnings.warn(msg)
```

Out[41]:

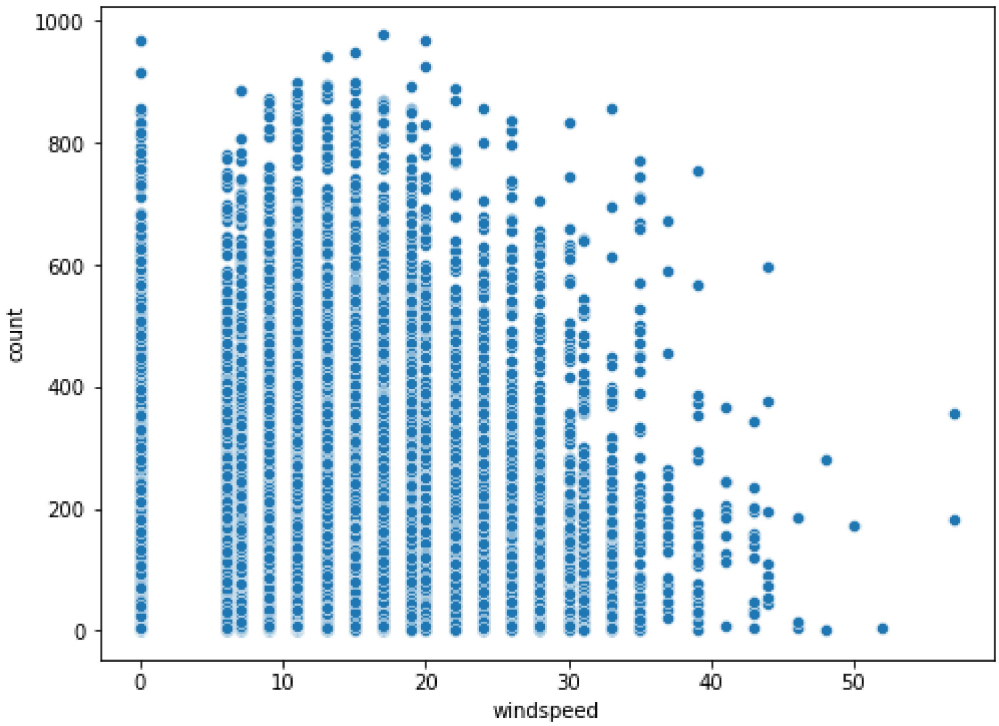
```
<seaborn.axisgrid.FacetGrid at 0x18f1b504040>
```



In [ ]:
```
Inference - Humidity is not much correlated to Demand
```

Windspeed vs Demand

In [42]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data = train_df, x = 'windspeed', y='count');
```



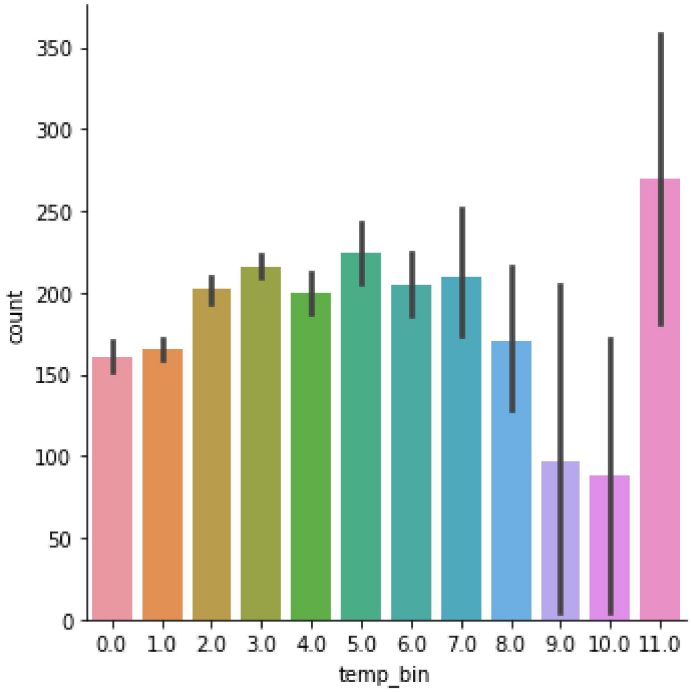
In [43]:
```
viz_df=train_df.copy()
viz_df.temp.describe()
```

```
viz_df['temp_bin']=np.floor(viz_df['windspeed'])//5
viz_df['temp_bin'].unique()

# now we can visualize with temp as discrete values
sns.factorplot(x="temp_bin",y="count",data=viz_df,kind='bar')
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning: Th
e `factorplot` function has been renamed to `catplot`. The original name will be removed
in a future release. Please update your code. Note that the default `kind` in `factorplo
t` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

Out[43]: <seaborn.axisgrid.FacetGrid at 0x18f1d20ef10>

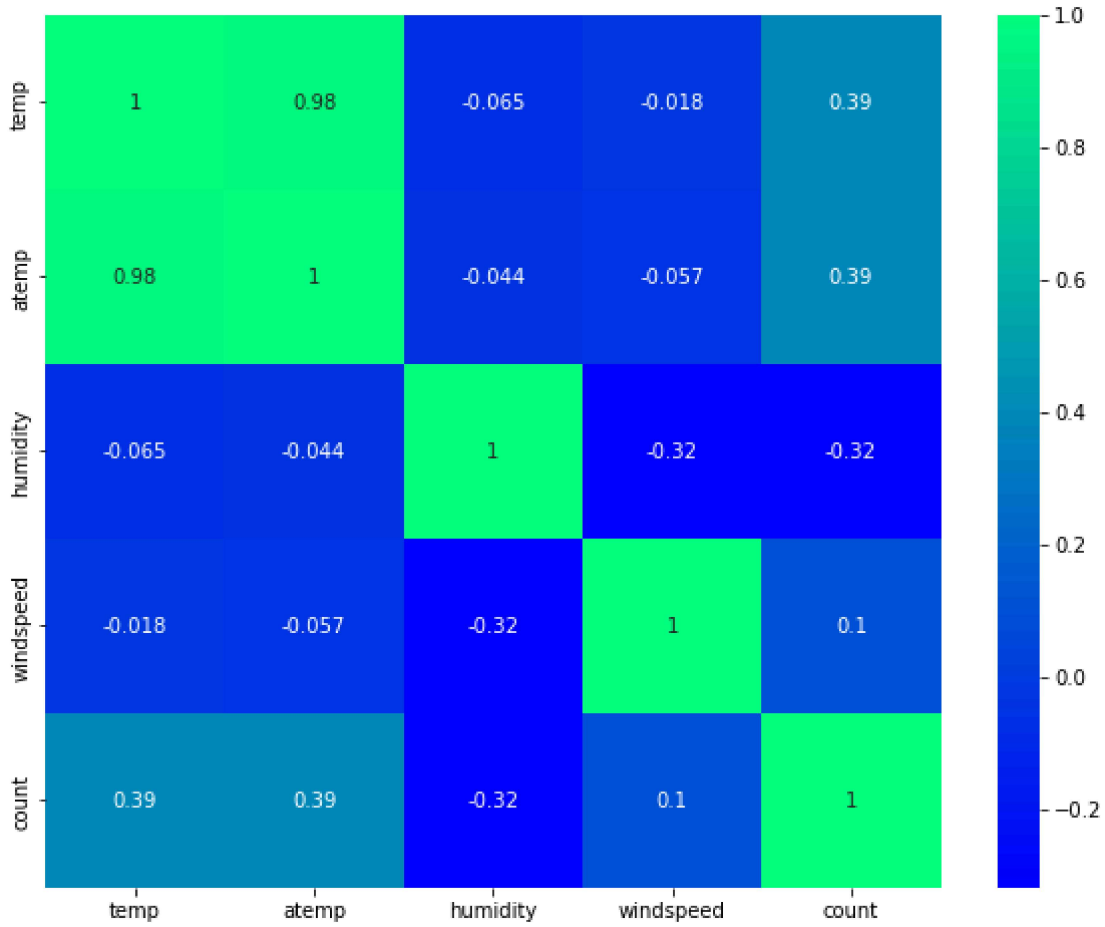


Inference - Since windspeed has a very low correlation with the demand so it does not affect the demand much.

Visualization of the Correlation Matrix of continuous features and understanding how it will affect the Target variable

In [44]:

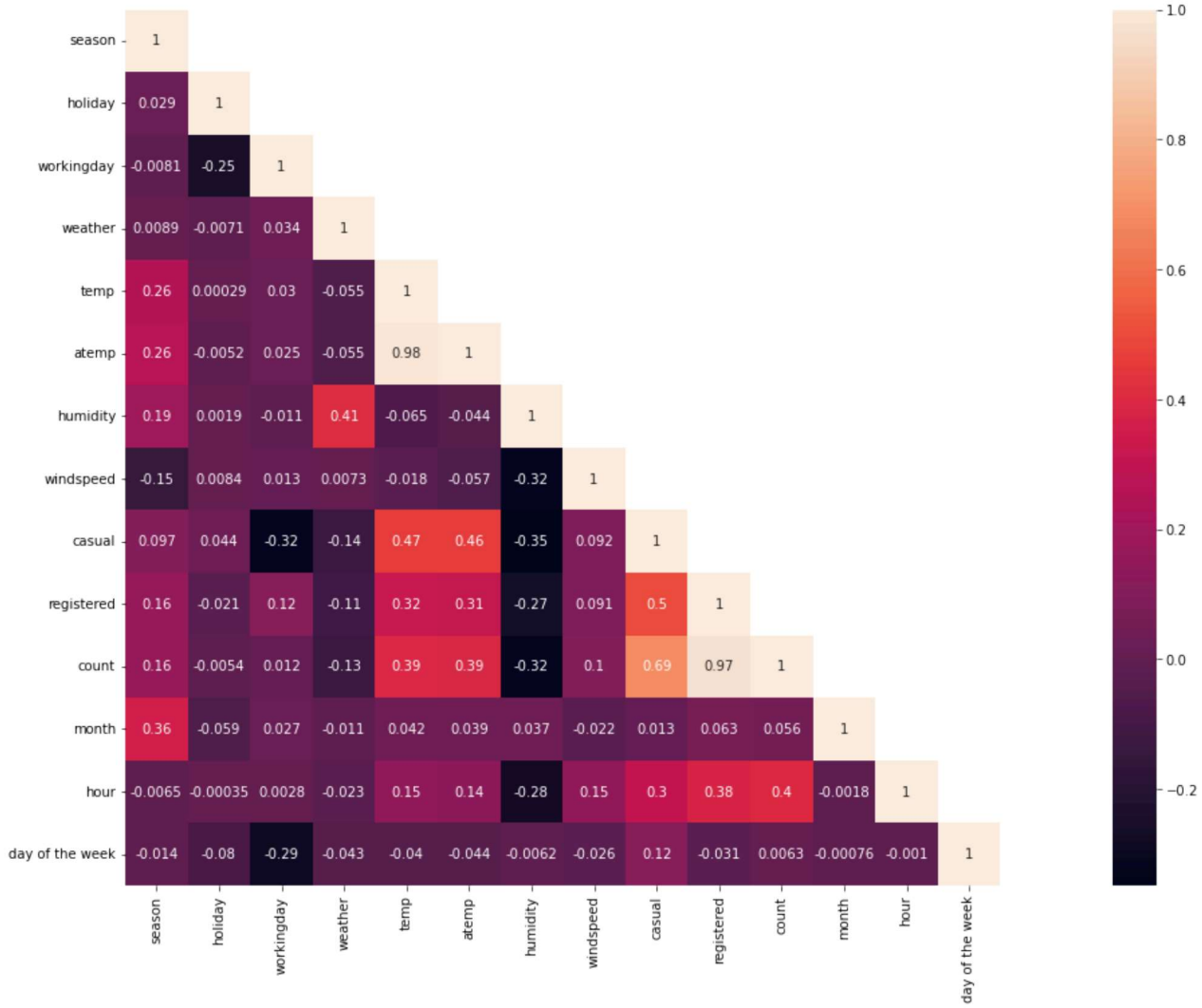
```
corrdata = train_df[["temp","atemp","humidity","windspeed","count"]]
corrmat = corrdata.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corrmat, annot = True, cmap= 'winter');
```



Visualization of the Correlation Matrix of all the independent features

In [45]:
```
cor_mat= train_df[:].corr()
mask = np.array(cor_mat)
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(30,12)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)
```

Out[45]: <AxesSubplot:>



Data Preprocessing

In [46]:
```
train_df
```

Out[46]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
0	1	0	0	1	9.84	14.395	81	0.0000	3	13
1	1	0	0	1	9.02	13.635	80	0.0000	8	32
2	1	0	0	1	9.02	13.635	80	0.0000	5	27
3	1	0	0	1	9.84	14.395	75	0.0000	3	10
4	1	0	0	1	9.84	14.395	75	0.0000	0	1
...	...	...	...	...	...	...	...	...	...	...

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
10881	4	0	1	1	15.58	19.695	50	26.0027	7	329
10882	4	0	1	1	14.76	17.425	57	15.0013	10	231
10883	4	0	1	1	13.94	15.910	61	15.0013	4	164
10884	4	0	1	1	13.94	17.425	61	6.0032	12	117
10885	4	0	1	1	13.12	16.665	66	8.9981	4	84

10886 rows × 16 columns

In [47]:
```
train_df["year"] = pd.to_numeric(train_df["year"])
```

Performing One Hot Encoding on categorical features

In [48]:
```
season = pd.get_dummies(train_df['season'],prefix='season',drop_first=True)
weather = pd.get_dummies(train_df['weather'],prefix='weather',drop_first=True)
holiday = pd.get_dummies(train_df['holiday'],prefix='holiday',drop_first=True)
month = pd.get_dummies(train_df['month'],prefix='month',drop_first=True)
hour = pd.get_dummies(train_df['hour'],prefix='hour',drop_first=True)
train_df = pd.concat([train_df,season,weather,holiday,month,hour],axis=1)
train_df.drop(['season','weather','holiday','month','hour'], axis=1,inplace=True)
```

In [49]:
```
pd.set_option('display.max_columns',52)
train_df.head()
```

Out[49]:

	workingday	temp	atemp	humidity	windspeed	casual	registered	count	year	date	day of the week	seas
0	0	9.84	14.395	81	0.0	3	13	16	1	2011-01-01	5	
1	0	9.02	13.635	80	0.0	8	32	40	1	2011-01-01	5	
2	0	9.02	13.635	80	0.0	5	27	32	1	2011-01-01	5	
3	0	9.84	14.395	75	0.0	3	10	13	1	2011-01-01	5	
4	0	9.84	14.395	75	0.0	0	1	1	1	2011-	5	

	workingday	temp	atemp	humidity	windspeed	casual	registered	count	year	date	day of the week	seas
										01-01		

Dropping the features with low correlation

In [50]:
```
train_df.drop(['date','day of the week','year','windspeed','workingday'], axis=1, inpla
```

In [51]:
```
train_df.head()
```

Out[51]:

	temp	atemp	humidity	casual	registered	count	season_2	season_3	season_4	weather_2	weathe
0	9.84	14.395	81	3	13	16	0	0	0	0	
1	9.02	13.635	80	8	32	40	0	0	0	0	
2	9.02	13.635	80	5	27	32	0	0	0	0	
3	9.84	14.395	75	3	10	13	0	0	0	0	
4	9.84	14.395	75	0	1	1	0	0	0	0	

In [52]:
```
train_df.columns.to_series().groupby(train_df.dtypes).groups
```

Out[52]:
```
{uint8: ['season_2', 'season_3', 'season_4', 'weather_2', 'weather_3', 'weather_4', 'hol
iday_1', 'month_2', 'month_3', 'month_4', 'month_5', 'month_6', 'month_7', 'month_8', 'm
onth_9', 'month_10', 'month_11', 'month_12', 'hour_1', 'hour_2', 'hour_3', 'hour_4', 'ho
ur_5', 'hour_6', 'hour_7', 'hour_8', 'hour_9', 'hour_10', 'hour_11', 'hour_12', 'hour_1
3', 'hour_14', 'hour_15', 'hour_16', 'hour_17', 'hour_18', 'hour_19', 'hour_20', 'hour_2
1', 'hour_22', 'hour_23'], int64: ['humidity', 'casual', 'registered', 'count'], float6
4: ['temp', 'atemp']}
```

In [53]:
```
train_df.dtypes
```

Out[53]:
```
temp          float64
atemp         float64
humidity        int64
casual          int64
registered      int64
count           int64
season_2        uint8
season_3        uint8
season_4        uint8
weather_2       uint8
weather_3       uint8
weather_4       uint8
holiday_1       uint8
month_2         uint8
month_3         uint8
month_4         uint8
```

```
month_5         uint8
month_6         uint8
month_7         uint8
month_8         uint8
month_9         uint8
month_10        uint8
month_11        uint8
month_12        uint8
hour_1          uint8
hour_2          uint8
hour_3          uint8
hour_4          uint8
hour_5          uint8
hour_6          uint8
hour_7          uint8
hour_8          uint8
hour_9          uint8
hour_10         uint8
hour_11         uint8
hour_12         uint8
hour_13         uint8
hour_14         uint8
hour_15         uint8
hour_16         uint8
hour_17         uint8
hour_18         uint8
hour_19         uint8
hour_20         uint8
hour_21         uint8
hour_22         uint8
hour_23         uint8
dtype: object
```

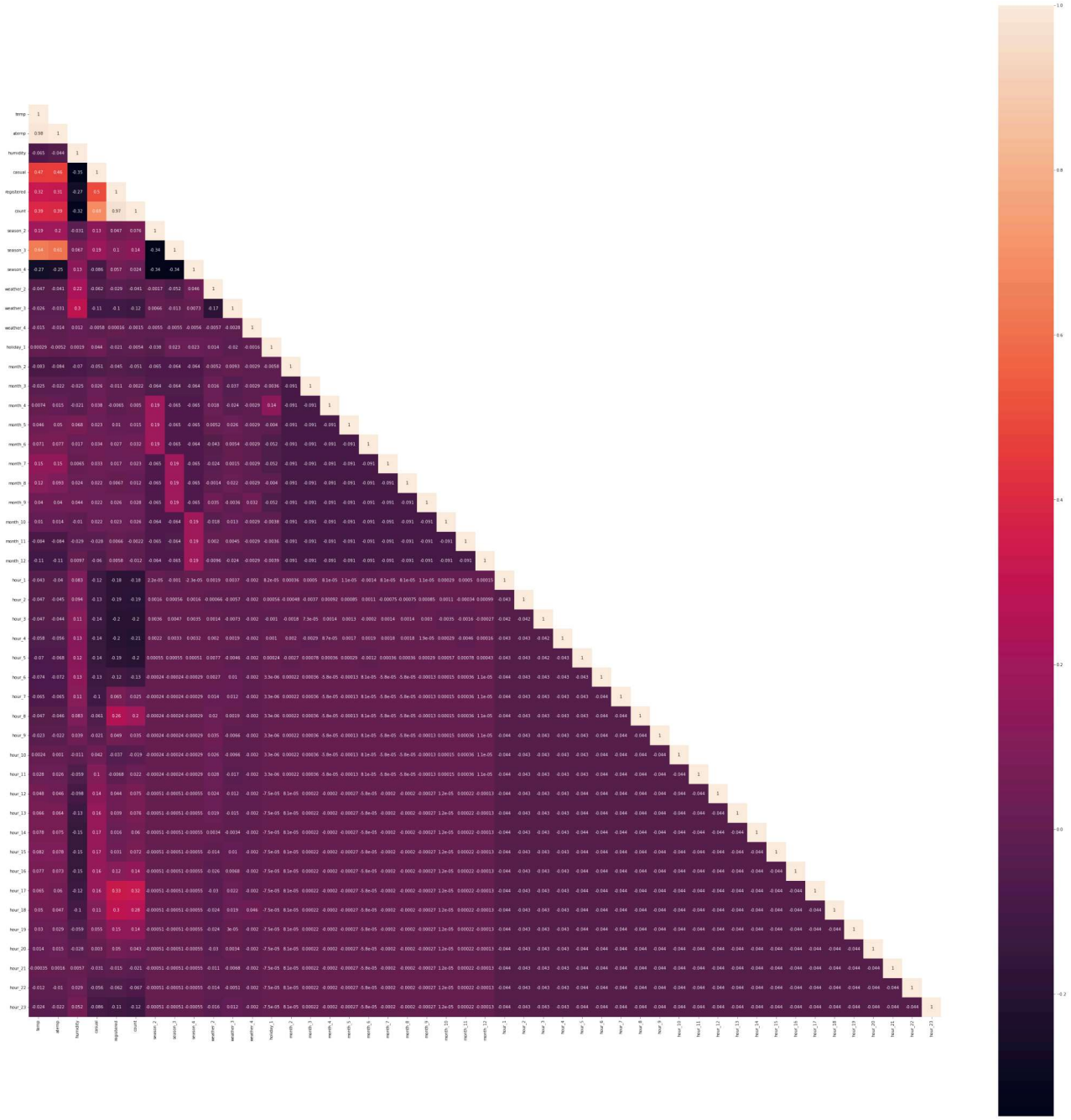
Visualization of the Correlation Matrix of the preprocessed data

In [54]:

```
cor_mat= train_df[:].corr()
mask = np.array(cor_mat)
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(50,50)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)
```

Out[54]:

```
<AxesSubplot:>
```



Splitting the data

In [55]:
```
X = train_df.drop('count',axis=1)
```

In [56]:
```
X.shape
```

Out[56]:
```
(10886, 46)
```

In [57]:
```
y = np.log(train_df['count'])
```

In [58]:
```
y.shape
```

Out[58]:
```
(10886,)
```

In [59]:
```
from sklearn.model_selection import train_test_split
```

In [60]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.2)
```

Using Linear Regression

In [61]:
```
from sklearn.linear_model import LinearRegression
```

In [62]:
```
linear_rg = LinearRegression()
```

In [63]:
```
linear_rg.fit(X_train,y_train)
```

Out[63]:
```
LinearRegression()
```

In [64]:
```
preds1 = linear_rg.predict(X_test)
```

In [65]:
```
preds1
```

Out[65]:
```
array([3.70255125, 4.74222312, 2.05952471, ..., 5.7742364 , 4.50704272,
       6.25428832])
```

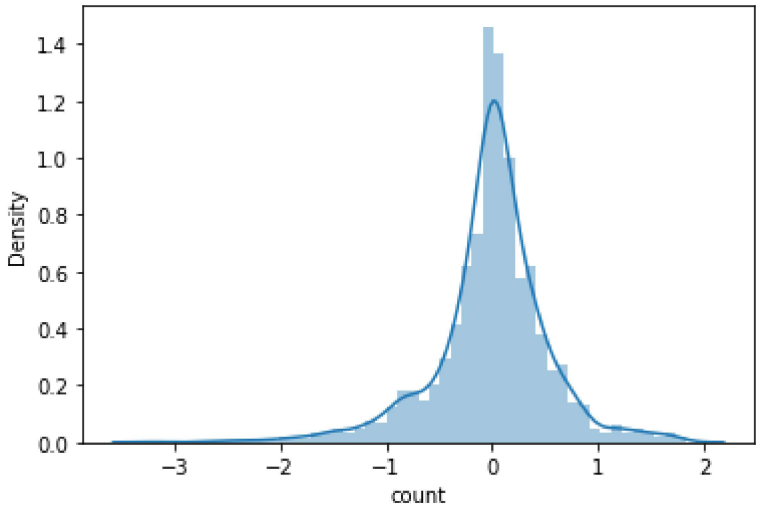
In [66]:
```
from sklearn import metrics
```

In [67]:
```
r2_score = metrics.r2_score(y_test,preds1)
print('r2_score is:{}'.format(r2_score))

print('MAE:',metrics.mean_absolute_error(y_test,preds1))
print('MSE:',metrics.mean_squared_error(y_test,preds1))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,preds1)))
sns.distplot(y_test-preds1)
```

```
r2_score is:0.8692571038643412
MAE: 0.3680634793445018
MSE: 0.29606787049405886
RMSE: 0.5441211909988977
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[67]:
```
<AxesSubplot:xlabel='count', ylabel='Density'>
```



Accuracy - 86.55%

Using Decision Tree

In [68]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [69]:
```
dt_rg = DecisionTreeRegressor(max_depth=5)
```

In [70]:
```
dt_rg.fit(X_train,y_train)
```

Out[70]:
```
DecisionTreeRegressor(max_depth=5)
```

In [71]:
```
preds2 = dt_rg.predict(X_test)
```

In [72]:
```
preds2
```

Out[72]:
```
array([3.48832432, 4.9477944 , 1.38159999, ..., 5.8104737 , 4.58533259,
       6.05308632])
```

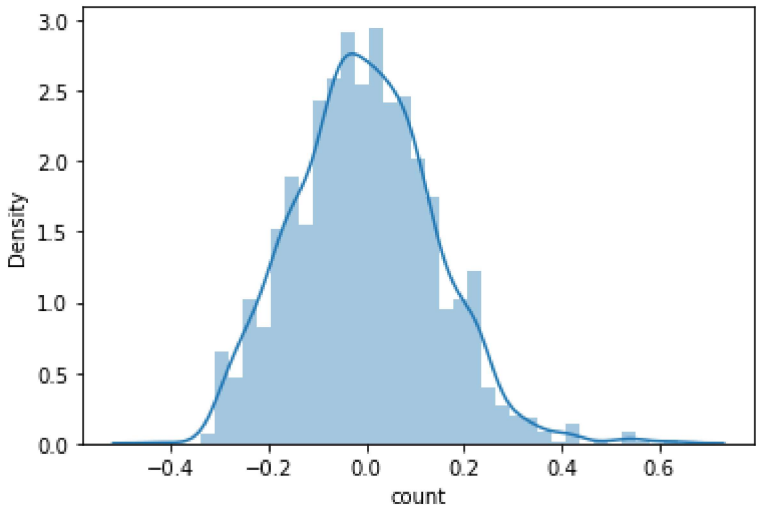
In [73]:
```
r2_score = metrics.r2_score(y_test,preds2)
print('r2_score is:{}'.format(r2_score))

print('MAE:',metrics.mean_absolute_error(y_test,preds2))
print('MSE:',metrics.mean_squared_error(y_test,preds2))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,preds2)))
sns.distplot(y_test-preds2)
```

```
r2_score is:0.9909591591485867
MAE: 0.11375195700924633
MSE: 0.020473024366664504
RMSE: 0.14308397662444422
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
```

```
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[73]:
```
<AxesSubplot:xlabel='count', ylabel='Density'>
```



Accuracy - 99.03%

Using Hypertuned KNN

In [74]:
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
```

In [75]:
```
n_neighbors=[]
for i in range (0,50,5):
    if(i!=0):
        n_neighbors.append(i)
params_dict={'n_neighbors':n_neighbors,'n_jobs':[-1]}
clf_knn = GridSearchCV(estimator=KNeighborsRegressor(),param_grid=params_dict,scoring='
clf_knn.fit(X_train,y_train)
preds3 = clf_knn.predict(X_test)
```

In [76]:
```
preds3
```

Out[76]:
```
array([3.50735595, 4.85927873, 1.60601682, ..., 5.88196062, 4.59478098,
       6.04212532])
```

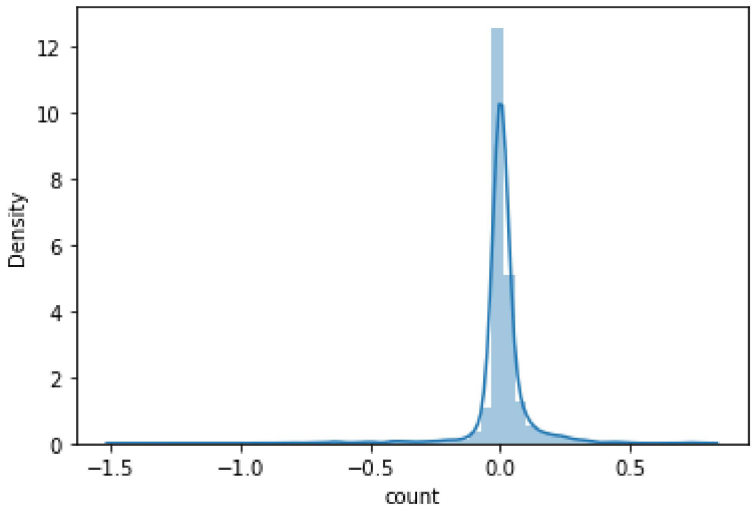
In [77]:
```
r2_score = metrics.r2_score(y_test,preds3)
print('r2_score is:{}'.format(r2_score))

print('MAE:',metrics.mean_absolute_error(y_test,preds3))
print('MSE:',metrics.mean_squared_error(y_test,preds3))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,preds3)))
sns.distplot(y_test-preds3)
```

```
r2_score is:0.9938745586264556
MAE: 0.04766722478677022
MSE: 0.013871089266829194
RMSE: 0.11777558858621422
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[77]:
```
<AxesSubplot:xlabel='count', ylabel='Density'>
```



Accuracy - 99.32%

Using Hypertuned Random Forest

In [78]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [79]:
```
no_of_test=[500]
params_dict={'n_estimators':no_of_test,'n_jobs':[-1],'max_features':["auto",'sqrt','log
clf_rf = GridSearchCV(estimator=RandomForestRegressor(),param_grid=params_dict,scoring=
clf_rf.fit(X_train,y_train)
preds4 = clf_rf.predict(X_test)
```

In [80]:
```
preds4
```

Out[80]:
```
array([3.62136171, 4.87608429, 1.60943791, ..., 5.89227954, 4.60414916,
       6.04248229])
```

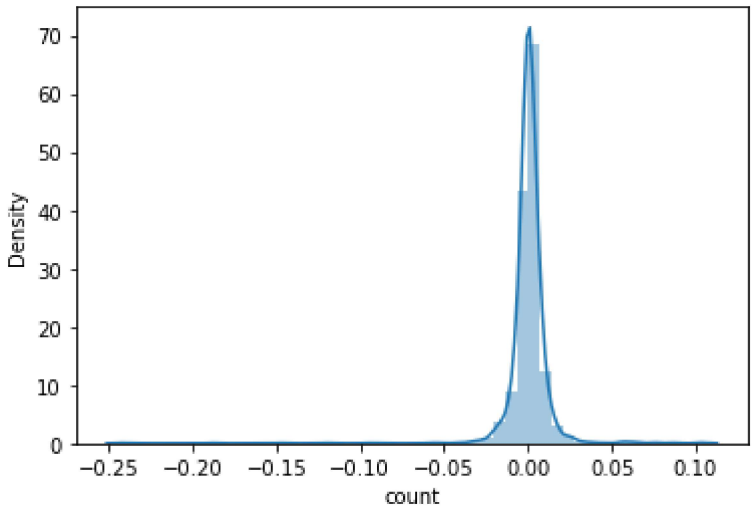
In [82]:
```
r2_score = metrics.r2_score(y_test,preds4)
print('r2_score is:{}'.format(r2_score))

print('MAE:',metrics.mean_absolute_error(y_test,preds4))
print('MSE:',metrics.mean_squared_error(y_test,preds4))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,preds4)))
sns.distplot(y_test-preds4)
```

```
r2_score is:0.9999190844143179
MAE: 0.0056852171691416835
MSE: 0.00018323370409219146
RMSE: 0.01353638445421049
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[82]: <AxesSubplot:xlabel='count', ylabel='Density'>



Accuracy - 99.995%

Using Hypertuned XGBoost

In [85]:

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_3564/2785909336.py in <module>
----> 1 import xgboost as xgb

ModuleNotFoundError: No module named 'xgboost'
```

In [ ]: