In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
summer_data = pd.read_csv('indian_summer.csv')
```

In [3]:

```
summer_data.head()
```

Out[3]:

winddir	sealevelpressure	cloudcover	visibility	sunrise	sunset	moonphase	conditions	desc
272.9	1002.8	0.0	3.1	01-04-2021 06:11	01-04-2021 18:39	0.60	Clear	con thro tl
275.0	1006.2	0.0	3.5	02-04-2021 06:10	02-04-2021 18:39	0.65	Clear	con thro tl
127.5	1008.8	1.4	3.5	03-04-2021 06:08	03-04-2021 18:40	0.70	Clear	con thro tl
157.6	1009.5	2.6	3.2	04-04-2021 06:07	04-04-2021 18:40	0.76	Clear	con thro tl
100.4	1007.8	38.4	3.1	05-04-2021 06:06	05-04-2021 18:41	0.81	Partially cloudy	thro tl

In [4]:

```
summer_data.tail()
```

Out[4]:

	City	Date	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew
13645	Hyderabad	26-06-2012	32.1	22.1	25.8	35.9	22.1	26.7	19.9
13646	Hyderabad	27-06-2012	31.8	21.1	25.5	33.3	21.1	26.1	19.0
13647	Hyderabad	28-06-2012	31.8	23.1	26.8	33.3	23.1	27.6	19.1
13648	Hyderabad	29-06-2012	32.8	23.1	26.7	35.1	23.1	27.5	19.5
13649	Hyderabad	30-06-2012	32.9	23.1	27.7	34.5	23.1	28.6	18.8

In [5]:

```
summer_data.shape
```

Out[5]:

```
(13650, 20)
```

In [6]:

```
summer_data.columns
```

Out[6]:

```
Index(['City', 'Date', 'tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'windspeed', 'windd
ir',
       'sealevelpressure', 'cloudcover', 'visibility', 'sunrise', 'sunse
t',
       'moonphase', 'conditions', 'description'],
      dtype='object')
```

In [7]:

```
summer_data.duplicated().sum()
```

Out[7]:

```
91
```

In [8]:

```
summer_data = summer_data.drop_duplicates()
```

In [9]:

```
summer_data.isnull().sum()
```

Out[9]:

```
City                  0
Date                  0
tempmax              35
tempmin              35
temp                 45
feelslikemax         36
feelslikemin         36
feelslike            46
dew                  45
humidity             45
windspeed            45
winddir              50
sealevelpressure    3019
cloudcover           45
visibility           45
sunrise               0
sunset                0
moonphase             0
conditions           45
description          45
dtype: int64
```

In [10]:

```
summer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13559 entries, 0 to 13649
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   City              13559 non-null  object 
 1   Date              13559 non-null  object 
 2   tempmax           13524 non-null  float64
 3   tempmin           13524 non-null  float64
 4   temp              13514 non-null  float64
 5   feelslikemax      13523 non-null  float64
 6   feelslikemin      13523 non-null  float64
 7   feelslike         13513 non-null  float64
 8   dew               13514 non-null  float64
 9   humidity          13514 non-null  float64
 10  windspeed         13514 non-null  float64
 11  winddir           13509 non-null  float64
 12  sealevelpressure  10540 non-null  float64
 13  cloudcover        13514 non-null  float64
 14  visibility        13514 non-null  float64
 15  sunrise           13559 non-null  object 
 16  sunset            13559 non-null  object 
 17  moonphase         13559 non-null  float64
 18  conditions        13514 non-null  object 
 19  description       13514 non-null  object 
dtypes: float64(14), object(6)
memory usage: 2.2+ MB
```

In [11]:

```
summer_data.describe()
```

Out[11]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike
count	13524.000000	13524.000000	13514.000000	13523.000000	13523.000000	13513.000000
mean	36.713931	25.821628	31.144332	40.201553	27.227834	33.702042
std	4.118441	3.218607	3.078221	5.400608	4.918202	4.677675
min	0.000000	0.000000	19.900000	0.000000	0.000000	19.900000
25%	34.000000	23.700000	29.200000	36.500000	23.700000	30.200000
50%	37.000000	26.000000	31.100000	40.000000	26.000000	33.500000
75%	39.800000	28.100000	33.200000	43.700000	31.100000	37.200000
max	50.000000	37.000000	40.500000	79.200000	43.300000	48.500000

In [12]:

```
summer_data.nunique()
```

Out[12]:

```
City                  15
Date                 910
tempmax              220
tempmin              194
temp                 188
feelslikemax         321
feelslikemin         253
feelslike            265
dew                  317
humidity            6100
windspeed            384
winddir             2490
sealevelpressure     260
cloudcover           977
visibility            80
sunrise            12214
sunset             12654
moonphase            101
conditions             6
description           31
dtype: int64
```

In [14]:

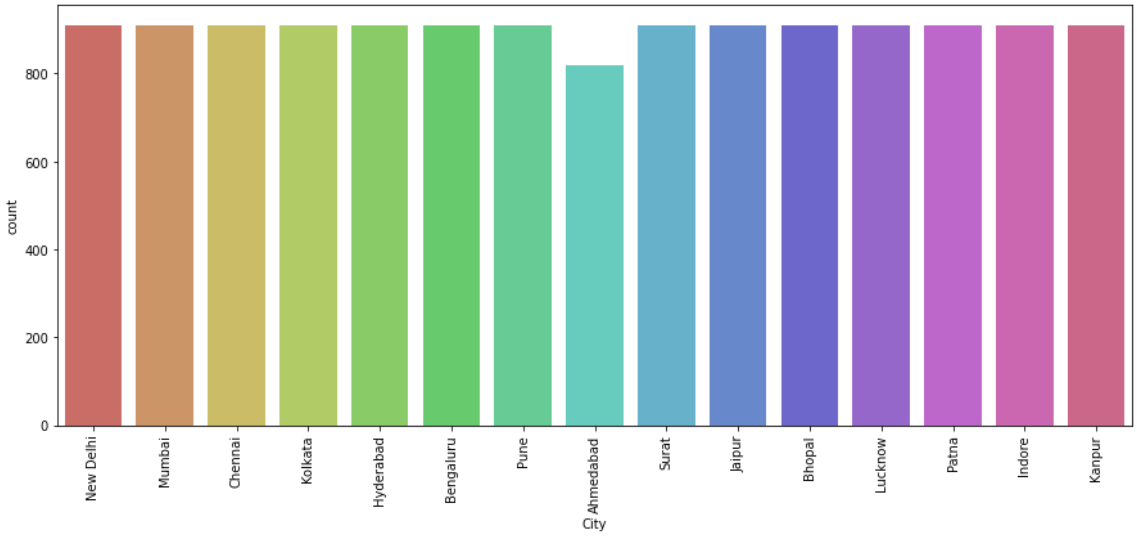
```
summer_data['City'].unique()
```

Out[14]:

```
array(['New Delhi', 'Mumbai', 'Chennai', 'Kolkata', 'Hyderabad',
       'Bengaluru', 'Pune', 'Ahmedabad', 'Surat', 'Jaipur', 'Bhopal',
       'Lucknow', 'Patna', 'Indore', 'Kanpur'], dtype=object)
```

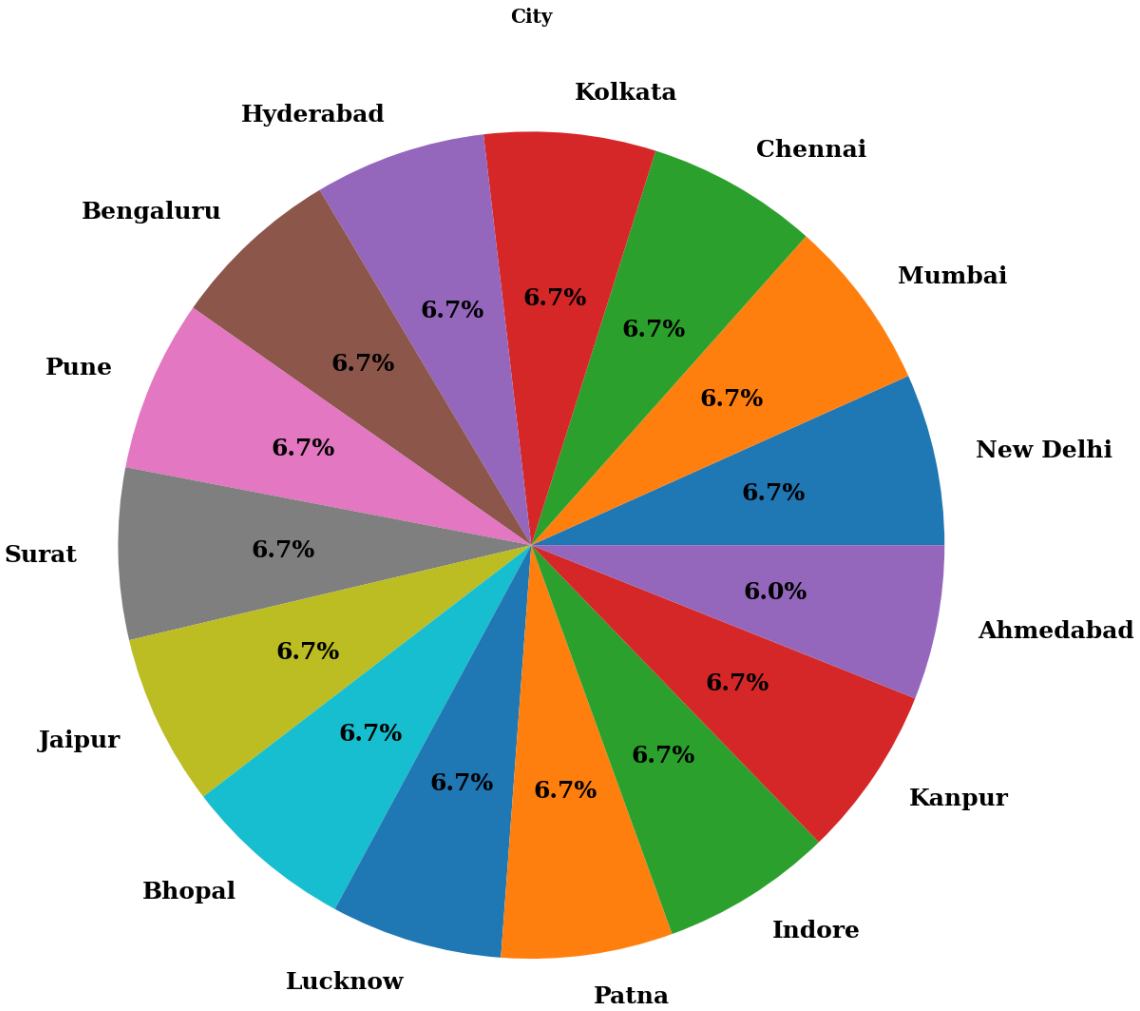
In [16]:

```
plt.figure(figsize=(15,6))
sns.countplot(summer_data['City'], data = summer_data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [17]:

```
plt.figure(figsize=(30,20))
plt.pie(summer_data['City'].value_counts(), labels=summer_data['City'].value_counts().in
                                            'color': 'black',
                                            'weight': 'bold',
                                            'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('City', size=20, **hfont)
plt.show()
```



In [18]:

```
summer_data['conditions'].unique()
```

Out[18]:

```
array(['Clear', 'Partially cloudy', 'Rain, Partially cloudy',
       'Rain, Overcast', 'Overcast', 'Rain', nan], dtype=object)
```

In [19]:

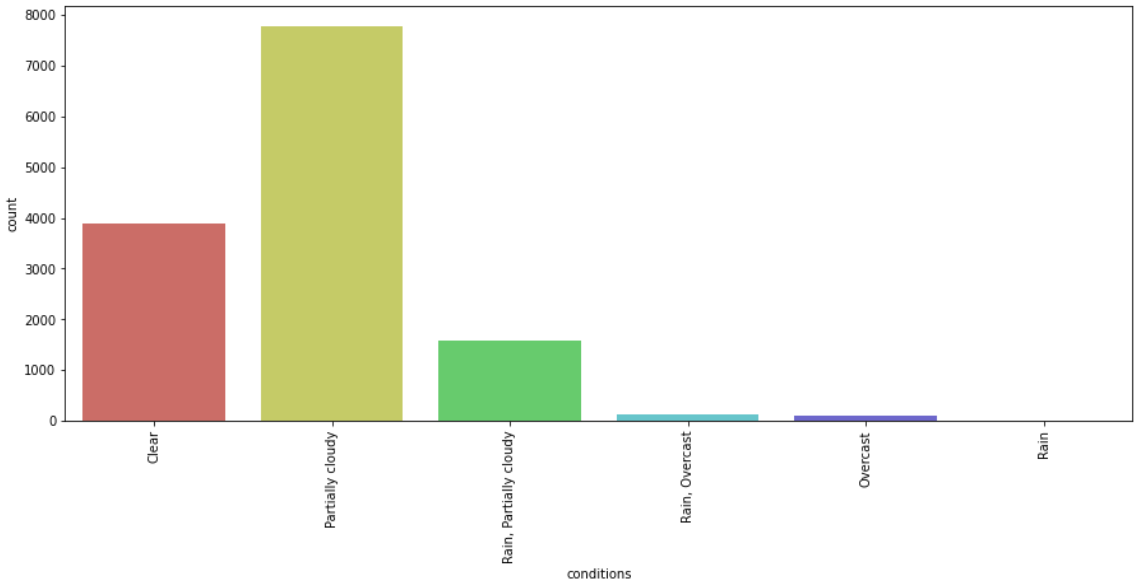
```
summer_data['conditions'].value_counts()
```

Out[19]:

```
Partially cloudy          7782
Clear                     3882
Rain, Partially cloudy    1592
Rain, Overcast             136
Overcast                   113
Rain                         9
Name: conditions, dtype: int64
```

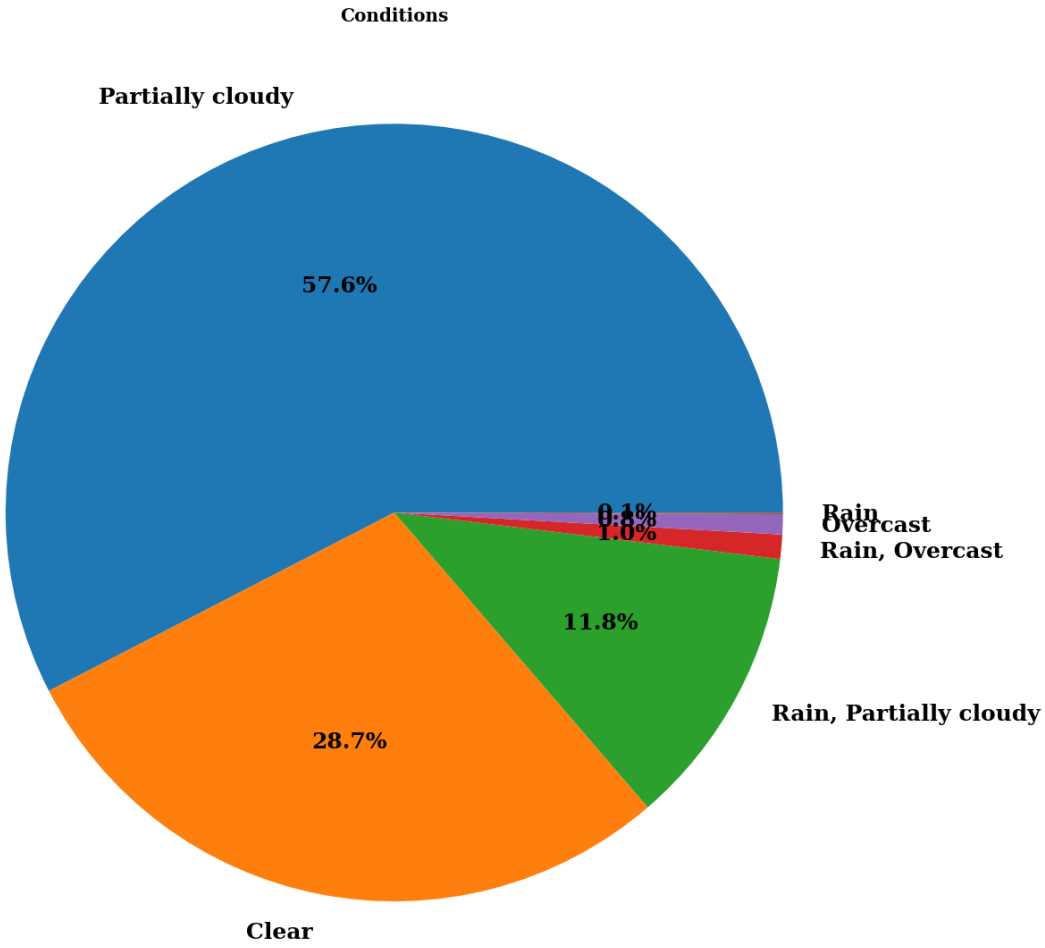
In [20]:

```
plt.figure(figsize=(15,6))
sns.countplot(summer_data['conditions'], data = summer_data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [21]:

```
plt.figure(figsize=(30,20))
plt.pie(summer_data['conditions'].value_counts(), labels=summer_data['conditions'].value
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Conditions', size=20, **hfont)
plt.show()
```



In [22]:

```
summer_data['description'].unique()
```

Out[22]:

```
array(['Clear conditions throughout the day.',
       'Partly cloudy throughout the day.',
       'Partly cloudy throughout the day with late afternoon rain.',
       'Partly cloudy throughout the day with morning rain.',
       'Becoming cloudy in the afternoon.',
       'Partly cloudy throughout the day with afternoon rain.',
       'Partly cloudy throughout the day with rain in the morning and afte
rnoon.',
       'Clearing in the afternoon.',
       'Cloudy skies throughout the day with afternoon rain.',
       'Cloudy skies throughout the day with rain.',
       'Partly cloudy throughout the day with early morning rain.',
       'Clearing in the afternoon with early morning rain.',
       'Partly cloudy throughout the day with rain.',
       'Cloudy skies throughout the day.',
       'Becoming cloudy in the afternoon with late afternoon rain.',
       'Partly cloudy throughout the day with rain clearing later.',
       'Cloudy skies throughout the day with a chance of rain throughout t
he day.',
       'Cloudy skies throughout the day with rain clearing later.',
       'Becoming cloudy in the afternoon with afternoon rain.',
       'Cloudy skies throughout the day with morning rain.',
       'Cloudy skies throughout the day with rain in the morning and after
noon.',
       'Cloudy skies throughout the day with late afternoon rain.',
       'Clearing in the afternoon with morning rain.',
       'Partly cloudy throughout the day with a chance of rain throughout
the day.',
       'Becoming cloudy in the afternoon with rain.',
       'Clear conditions throughout the day with early morning rain.',
       'Cloudy skies throughout the day with early morning rain.',
       'Clear conditions throughout the day with late afternoon rain.',
       'Clearing in the afternoon with rain clearing later.',
       'Becoming cloudy in the afternoon with rain clearing later.',
       'Clearing in the afternoon with late afternoon rain.', nan],
      dtype=object)
```

In [23]:

```
summer_data['sealevelpressure'].fillna(summer_data['sealevelpressure'].mean(), inplace=T
```

In [24]:

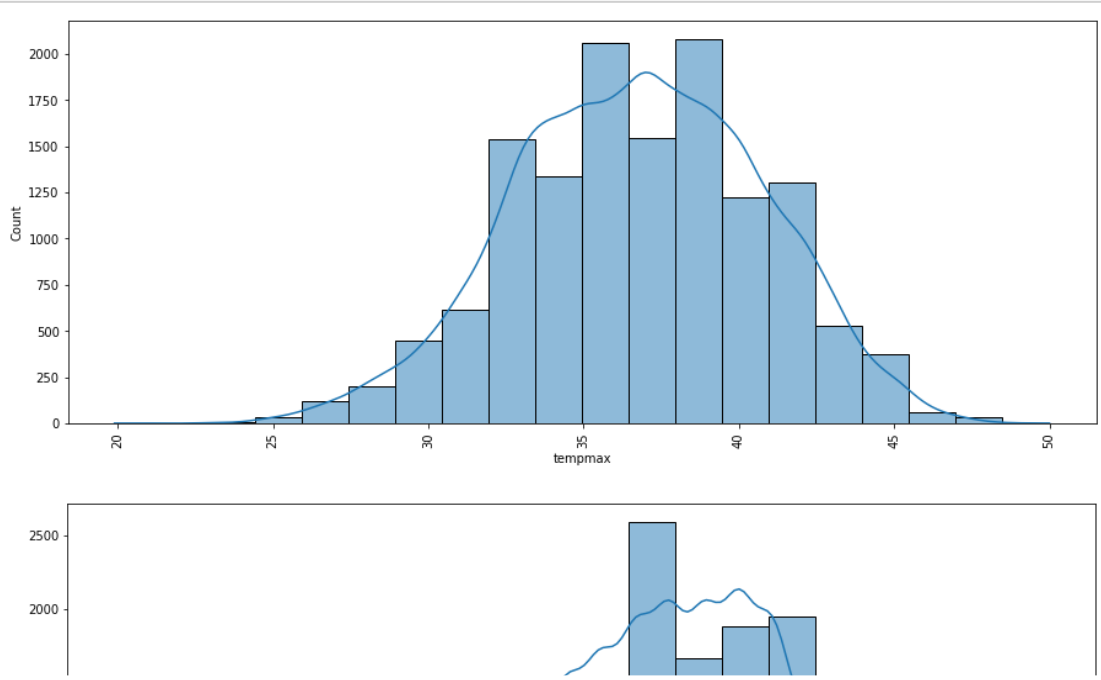
```
summer_data = summer_data.dropna()
```

In [25]:

```
summer_data_1 = summer_data[['tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'windspeed', 'winddir',
       'cloudcover', 'visibility', 'moonphase']]
```

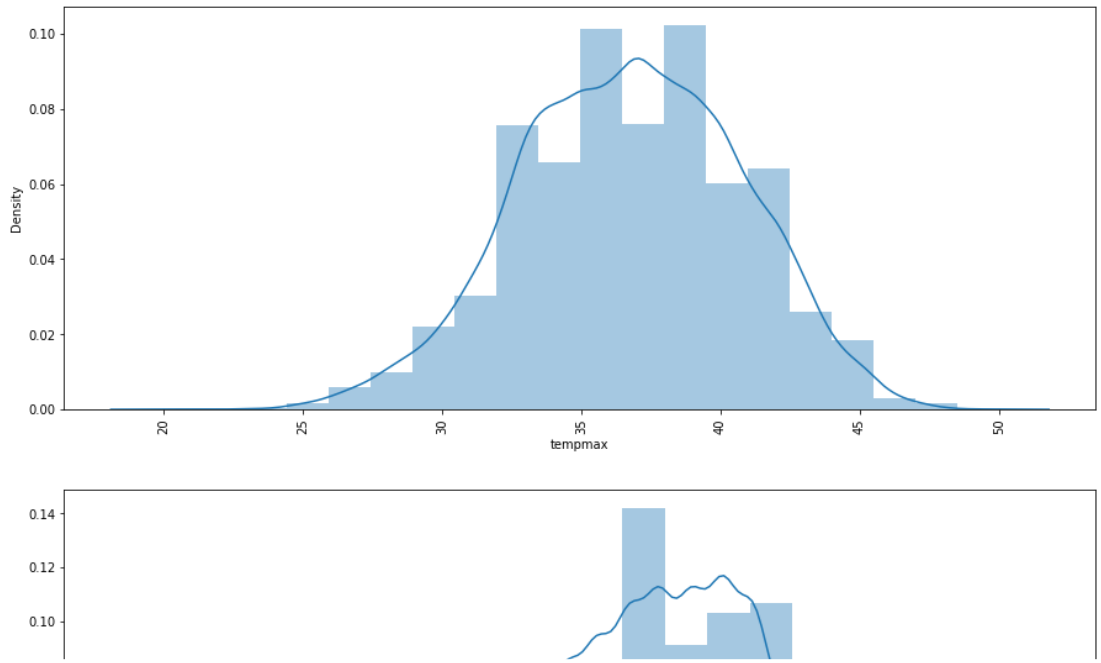
In [26]:

```
for i in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.histplot(summer_data[i], bins = 20, kde = True, palette='hls')
    plt.xticks(rotation = 90)
    plt.show()
```



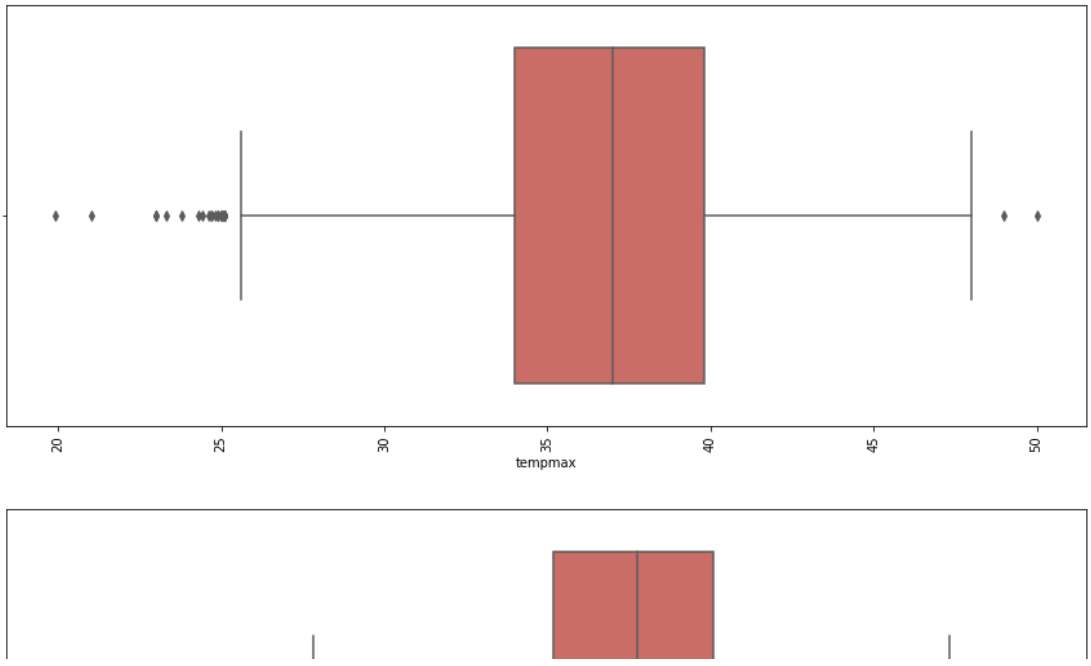
In [27]:

```
for i in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.distplot(summer_data[i], bins = 20, kde = True)
    plt.xticks(rotation = 90)
    plt.show()
```



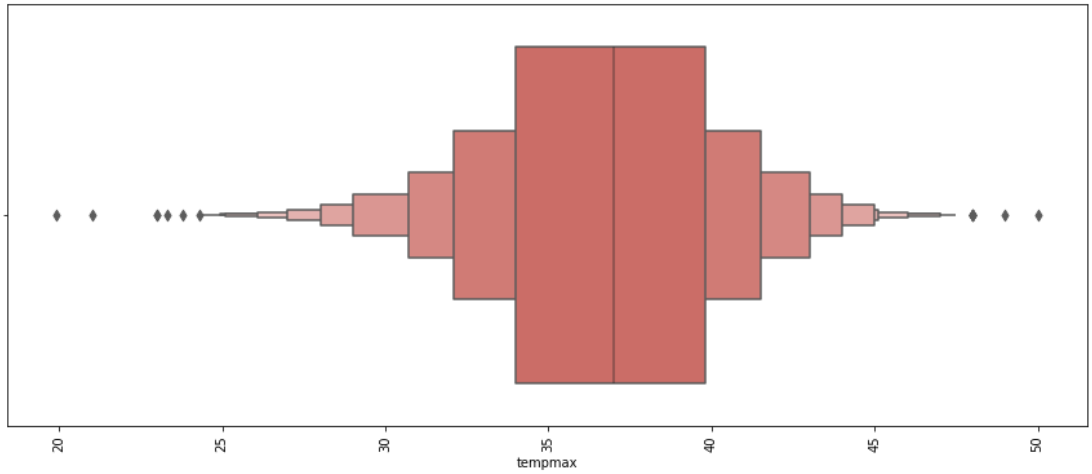
In [28]:

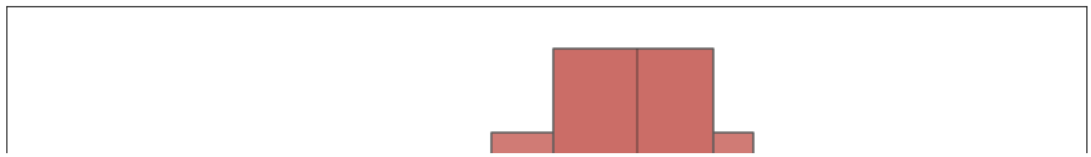
```
for i in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.boxplot(summer_data[i], palette='hls')
    plt.xticks(rotation = 90)
    plt.show()
```



In [29]:

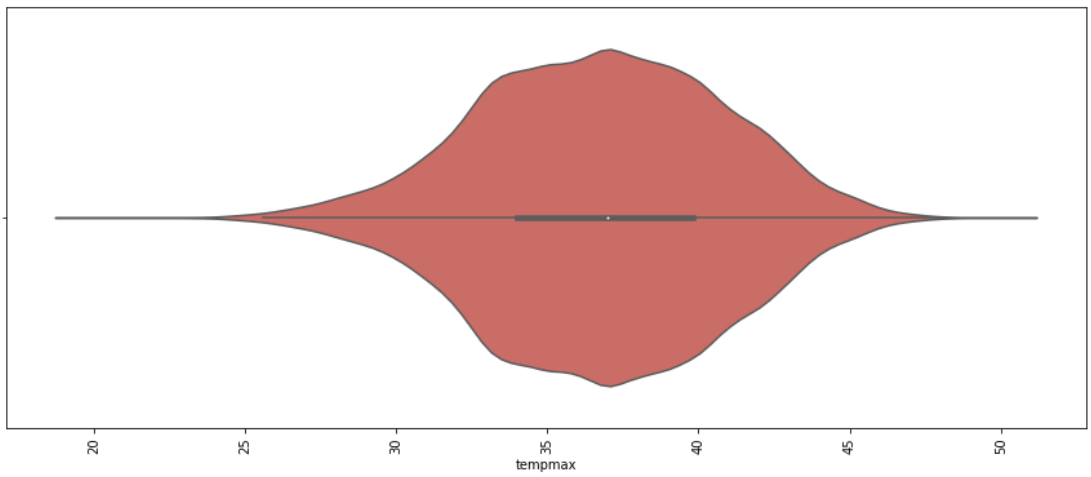
```
for i in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.boxenplot(summer_data[i], palette='hls')
    plt.xticks(rotation = 90)
    plt.show()
```

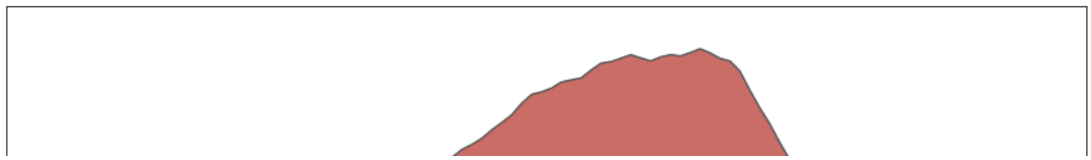




In [31]:

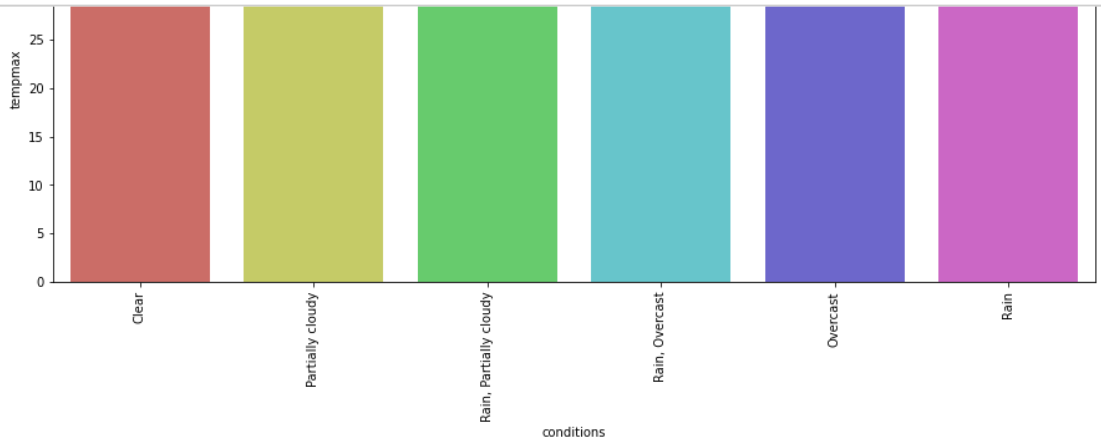
```
for i in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.violinplot(summer_data[i], palette='hls')
    plt.xticks(rotation = 90)
    plt.show()
```

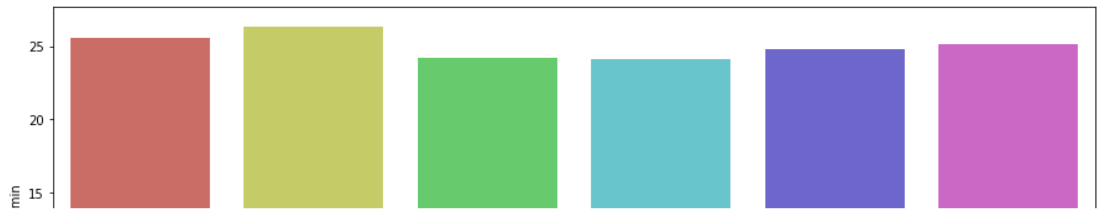




In [34]:

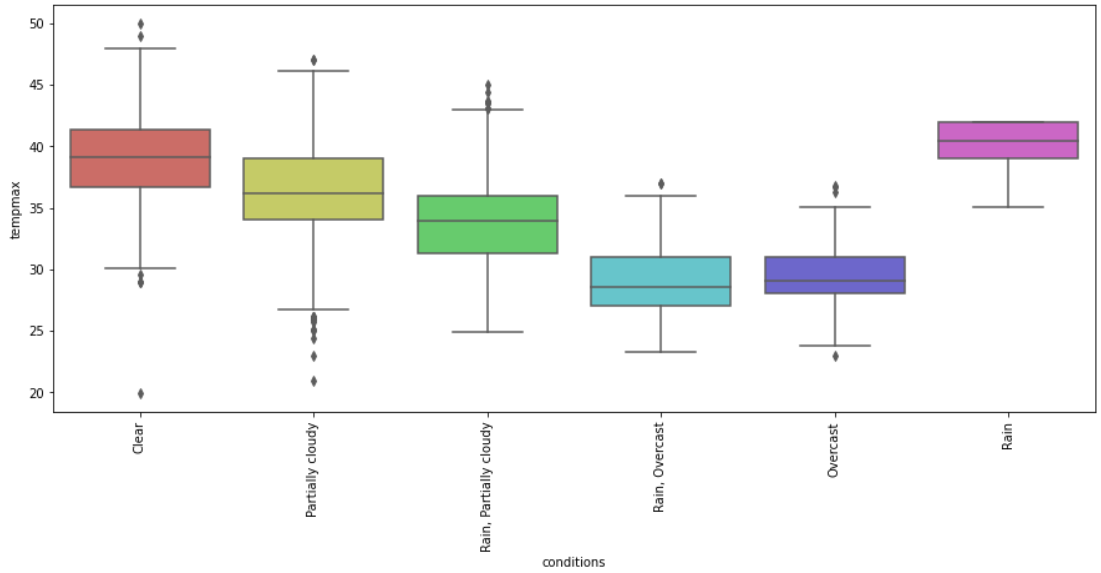
```
for j in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.barplot(x = summer_data['conditions'], y = summer_data[j], ci = None,
                palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```





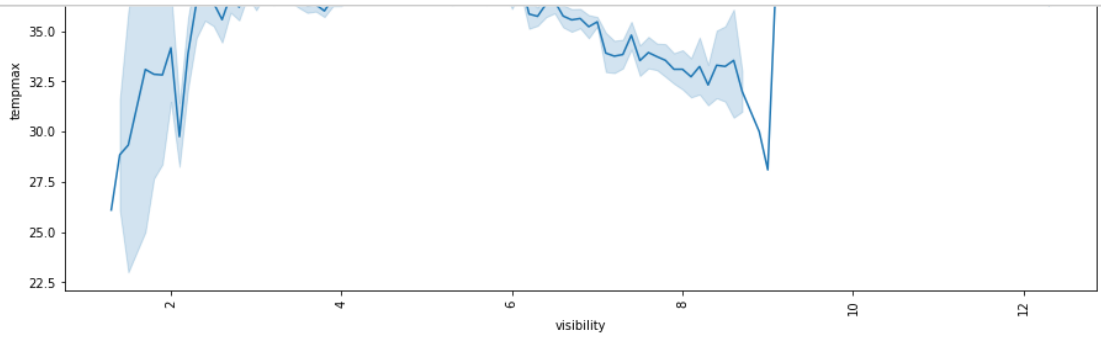
In [36]:

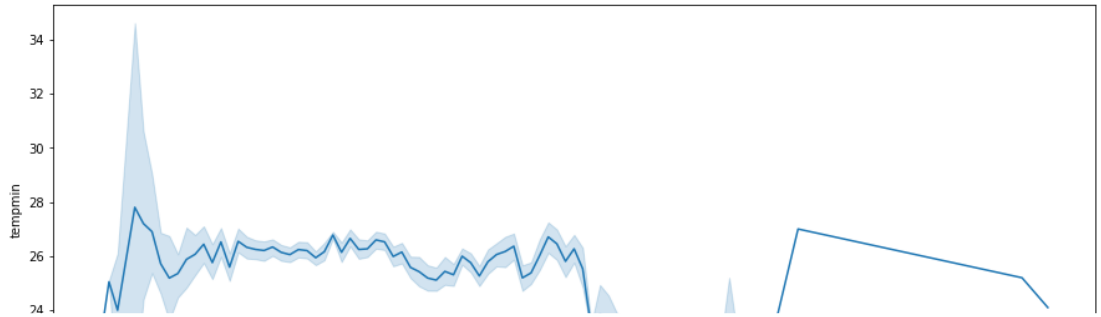
```
for j in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.boxplot(x = summer_data['conditions'], y = summer_data[j],
                palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



In [38]:

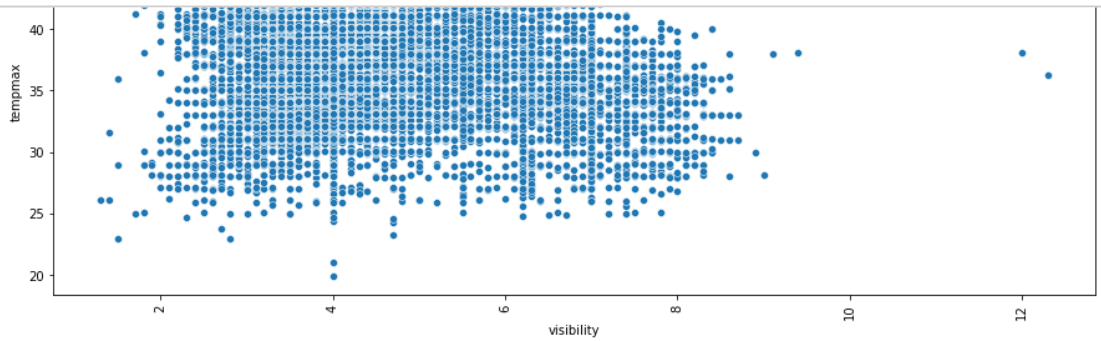
```
for j in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.lineplot(x = summer_data['visibility'], y = summer_data_1[j],
                 palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```

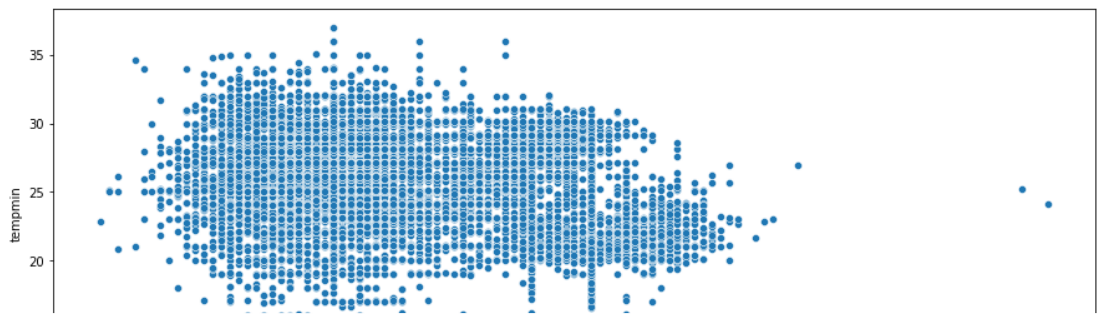




In [39]:

```
for j in summer_data_1.columns:
    plt.figure(figsize=(15,6))
    sns.scatterplot(x = summer_data['visibility'], y = summer_data_1[j],
                 palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```





In [40]:

```
count_clear=len(summer_data[summer_data.conditions=="Clear"])
count_pcloudy=len(summer_data[summer_data.conditions=="Partially cloudy"])
count_rpcloudy=len(summer_data[summer_data.conditions=="Rain, Partially cloudy"])
count_ro=len(summer_data[summer_data.conditions=="Rain, Overcast"])
count_overcast=len(summer_data[summer_data.conditions=="Overcast"])
count_rain=len(summer_data[summer_data.conditions=="Rain"])
```

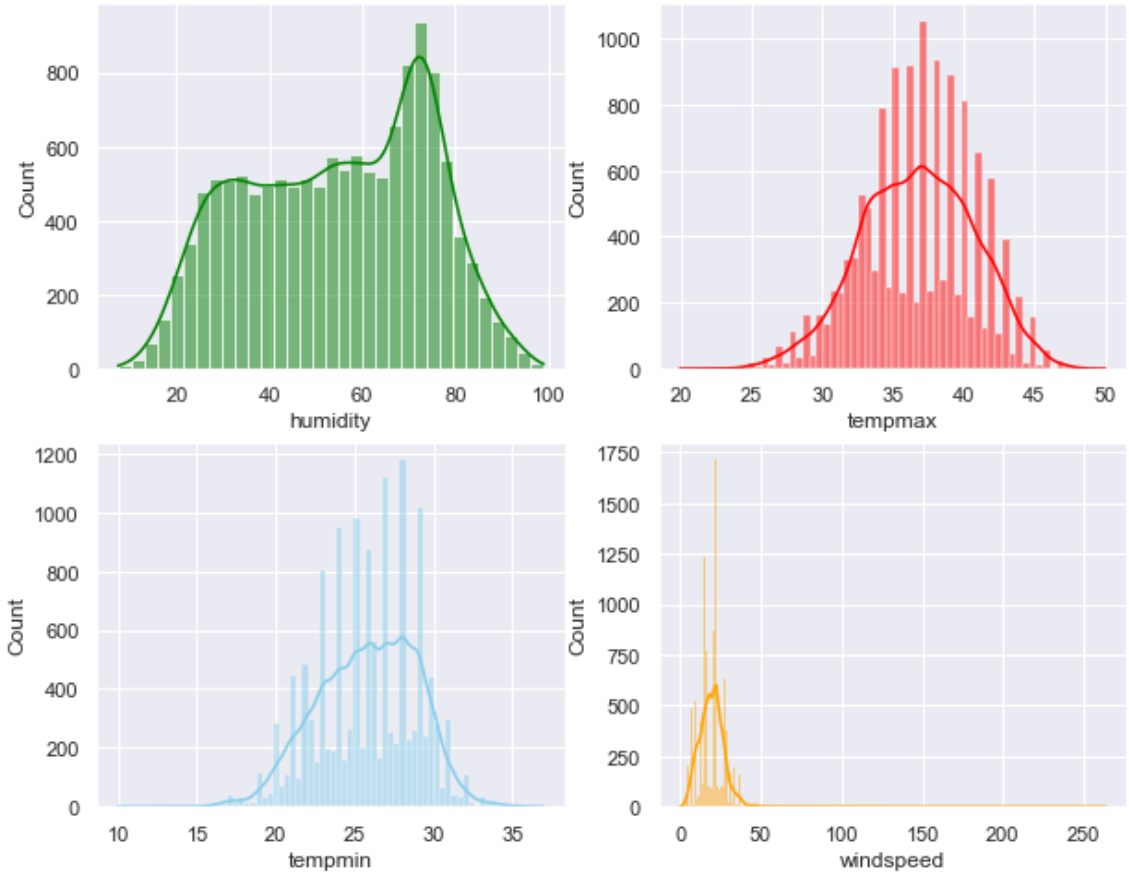
In [41]:

```
print("Percent of Clear:{:2f}%".format((count_clear/(len(summer_data.conditions))*100)))
print("Percent of Partial Cloudy:{:2f}%".format((count_pcloudy/(len(summer_data.conditio
print("Percent of Rain Partial Cloudy:{:2f}%".format((count_rpcloudy/(len(summer_data.co
print("Percent of Rain Overcast:{:2f}%".format((count_ro/(len(summer_data.conditions))*1
print("Percent of Overcast:{:2f}%".format((count_overcast/(len(summer_data.conditions))*
print("Percent of Rain:{:2f}%".format((count_rain/(len(summer_data.conditions))*100)))
```

```
Percent of Clear:28.723719%
Percent of Partial Cloudy:57.580693%
Percent of Rain Partial Cloudy:11.785609%
Percent of Rain Overcast:1.006811%
Percent of Overcast:0.836541%
Percent of Rain:0.066627%
```

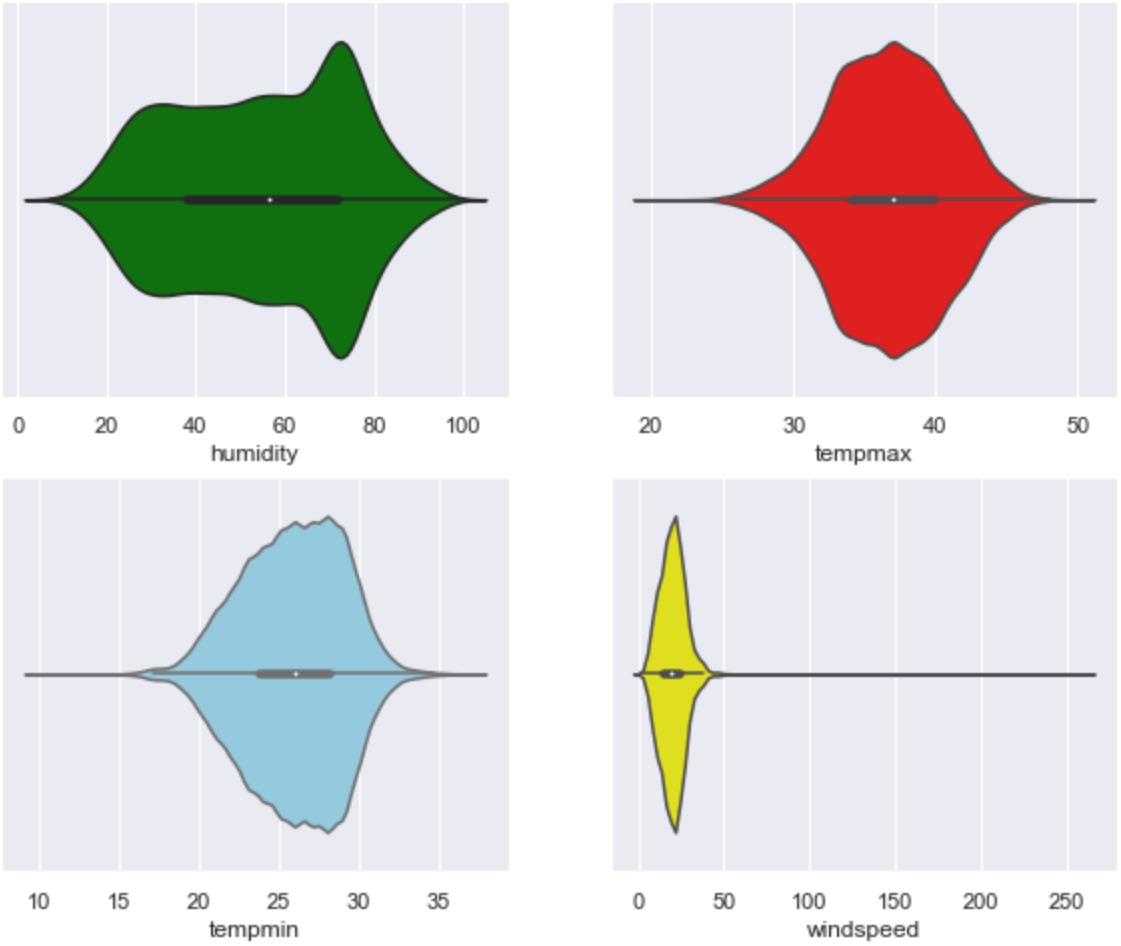
In [43]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=summer_data,x="humidity",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=summer_data,x="tempmax",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=summer_data,x="tempmin",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=summer_data,x="windspeed",kde=True,ax=axs[1,1],color='orange')
plt.show()
```



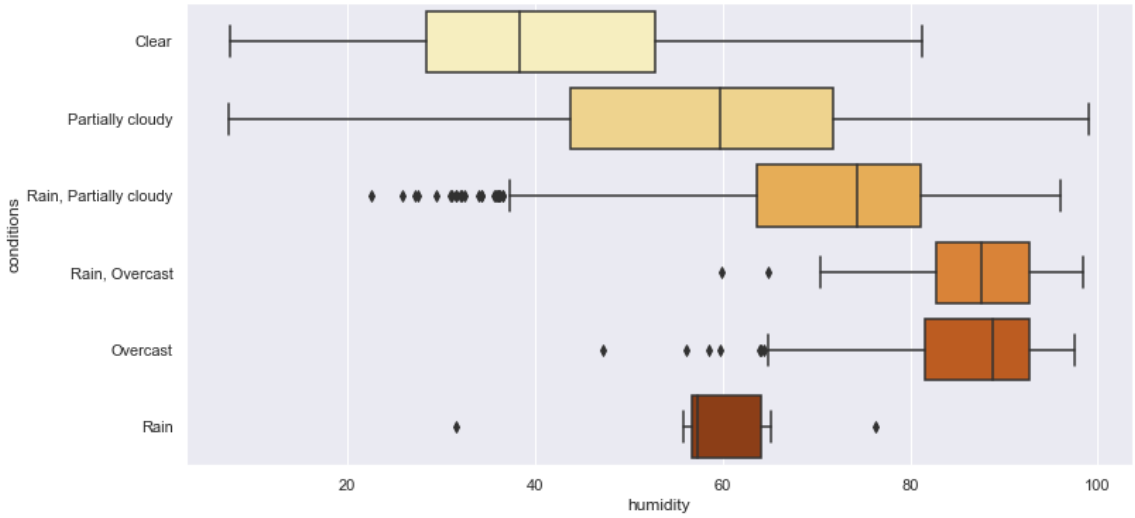
In [44]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.violinplot(data=summer_data,x="humidity",kde=True,ax=axs[0,0],color='green')
sns.violinplot(data=summer_data,x="tempmax",kde=True,ax=axs[0,1],color='red')
sns.violinplot(data=summer_data,x="tempmin",kde=True,ax=axs[1,0],color='skyblue')
sns.violinplot(data=summer_data,x="windspeed",kde=True,ax=axs[1,1],color='yellow')
plt.show()
```



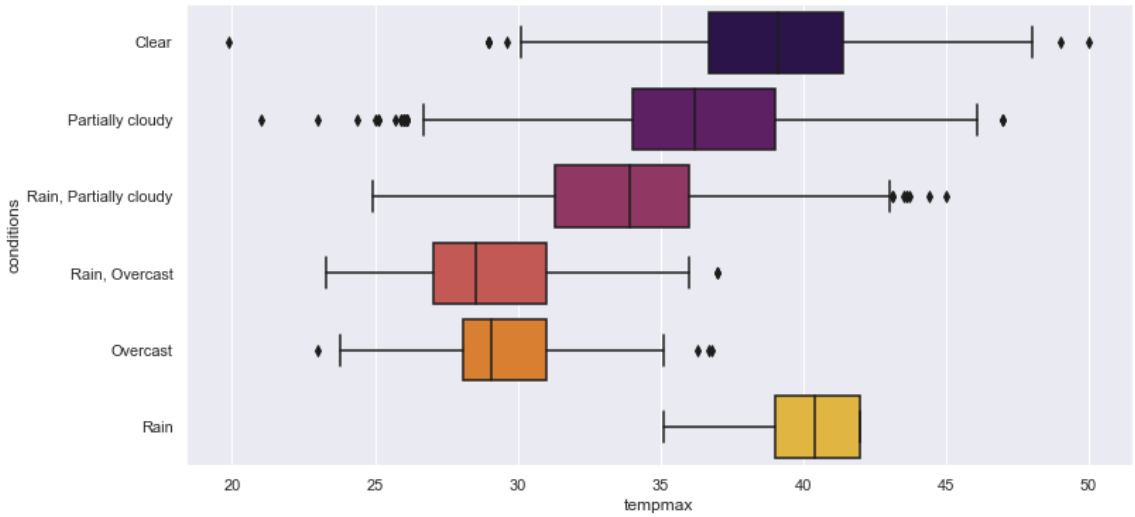
In [45]:

```
plt.figure(figsize=(12,6))
sns.boxplot("humidity","conditions",data=summer_data,palette="YlOrBr")
plt.show()
```



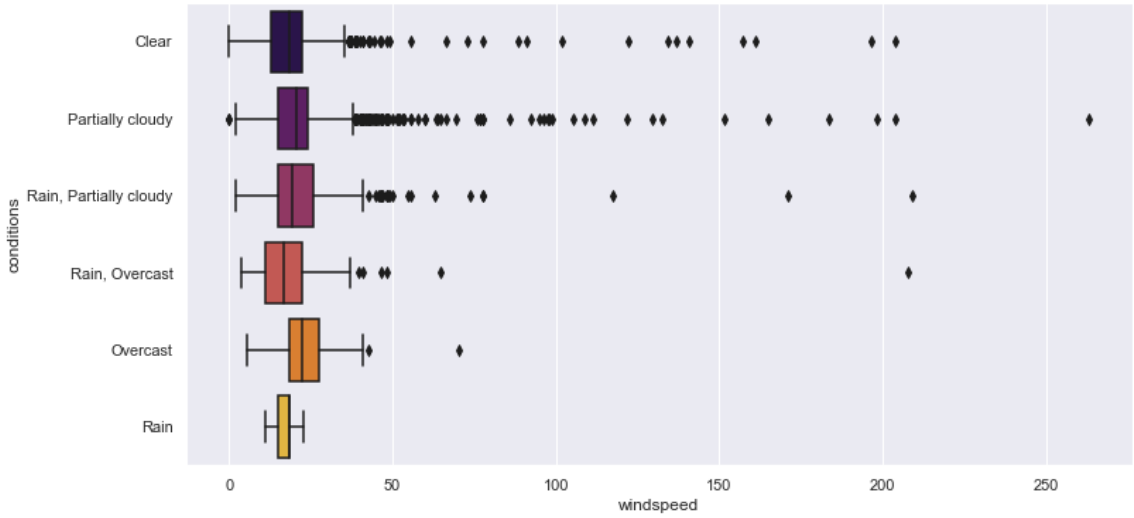
In [46]:

```
plt.figure(figsize=(12,6))
sns.boxplot("tempmax","conditions",data=summer_data,palette="inferno")
plt.show()
```



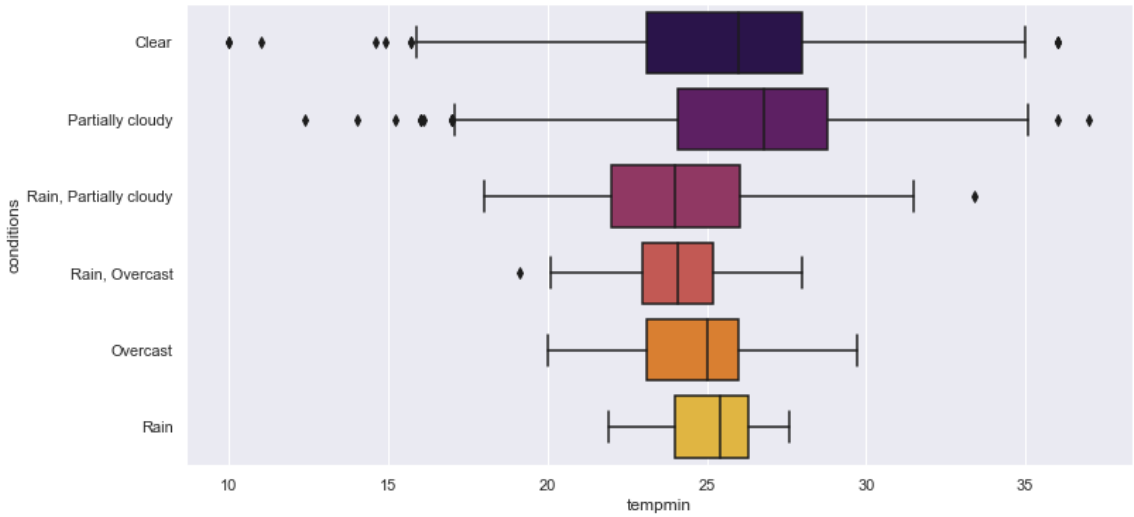
In [47]:

```
plt.figure(figsize=(12,6))
sns.boxplot("windspeed","conditions",data=summer_data,palette="inferno")
plt.show()
```



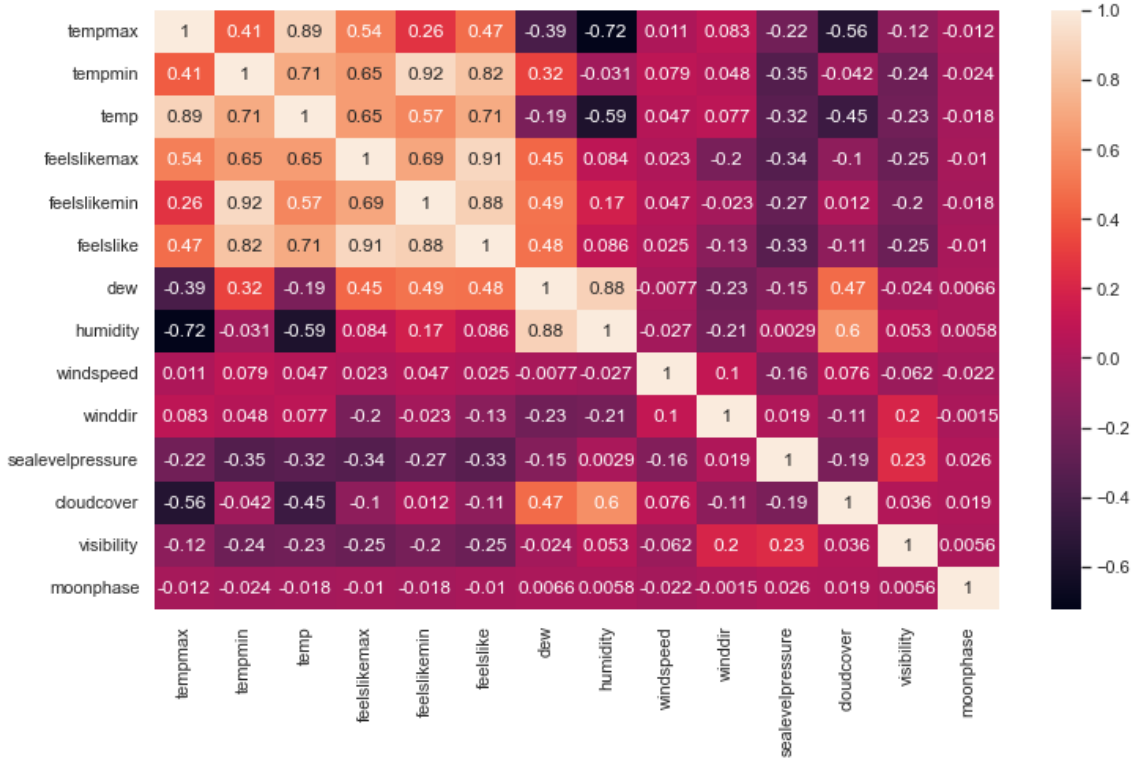
In [48]:

```
plt.figure(figsize=(12,6))
sns.boxplot("tempmin","conditions",data=summer_data,palette="inferno")
plt.show()
```



In [49]:

```
plt.figure(figsize=(12,7))
sns.heatmap(summer_data.corr(),annot=True,cmap='rocket')
plt.show()
```



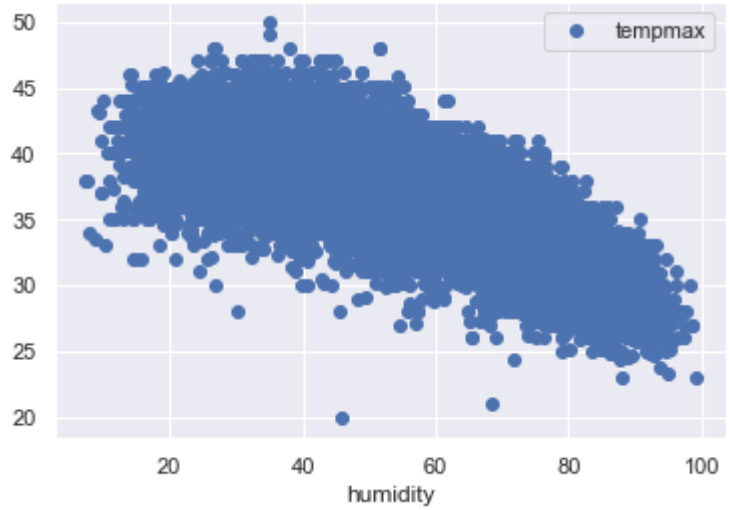
In [50]:

```
from scipy import stats
```

In [51]:

```
summer_data.plot("humidity","tempmax",style='o')
print("Pearson correlation:",summer_data["humidity"].corr(summer_data["tempmax"]))
print("T Test and P value:",stats.ttest_ind(summer_data["humidity"],summer_data["tempmax
```

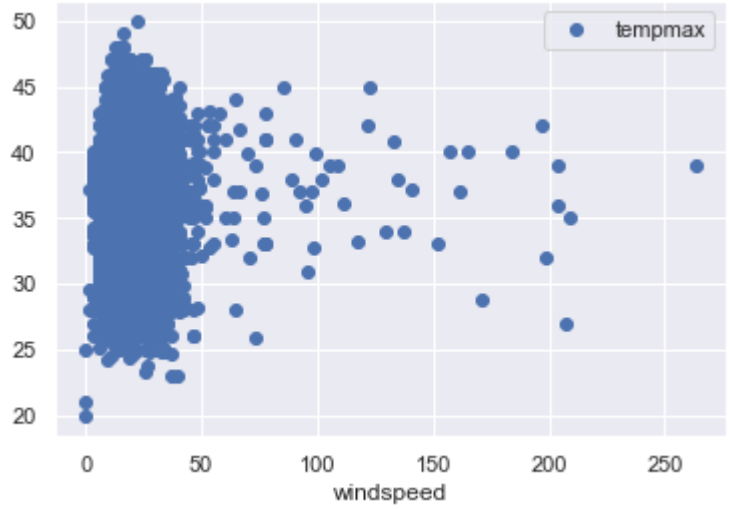
```
Pearson correlation: -0.7237550068149694
T Test and P value: Ttest_indResult(statistic=104.46321913378385, pvalue=
0.0)
```



In [52]:

```
summer_data.plot("windspeed","tempmax",style='o')
print("Pearson correlation:",summer_data["windspeed"].corr(summer_data["tempmax"]))
print("T Test and P value:",stats.ttest_ind(summer_data["windspeed"],summer_data["tempma
```

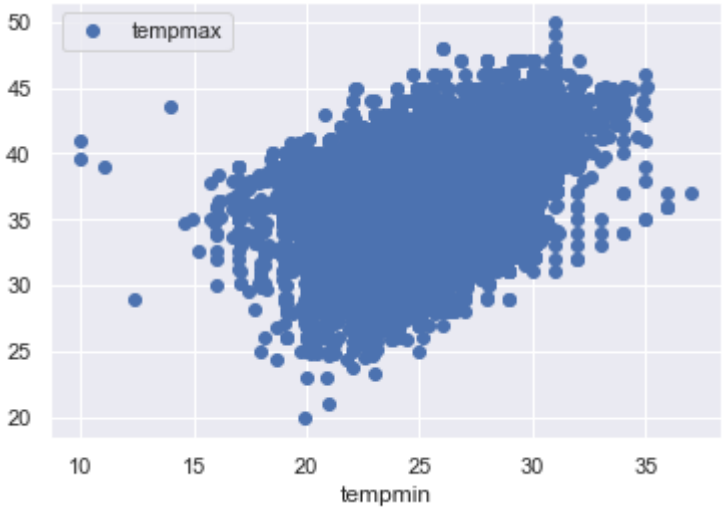
```
Pearson correlation: 0.010966452916681172
T Test and P value: Ttest_indResult(statistic=-181.31112581219472, pvalue=
0.0)
```



In [53]:

```
summer_data.plot("tempmin","tempmax",style='o')
print("Pearson correlation:",summer_data["tempmin"].corr(summer_data["tempmax"]))
print("T Test and P value:",stats.ttest_ind(summer_data["tempmin"],summer_data["tempmax"
```

```
Pearson correlation: 0.4104991573833356
T Test and P value: Ttest_indResult(statistic=-249.37293385092545, pvalue=
0.0)
```



In [54]:

```
df=summer_data.drop(['Date', 'sunrise', 'sunset', 'description'],axis=1)
```

In [55]:

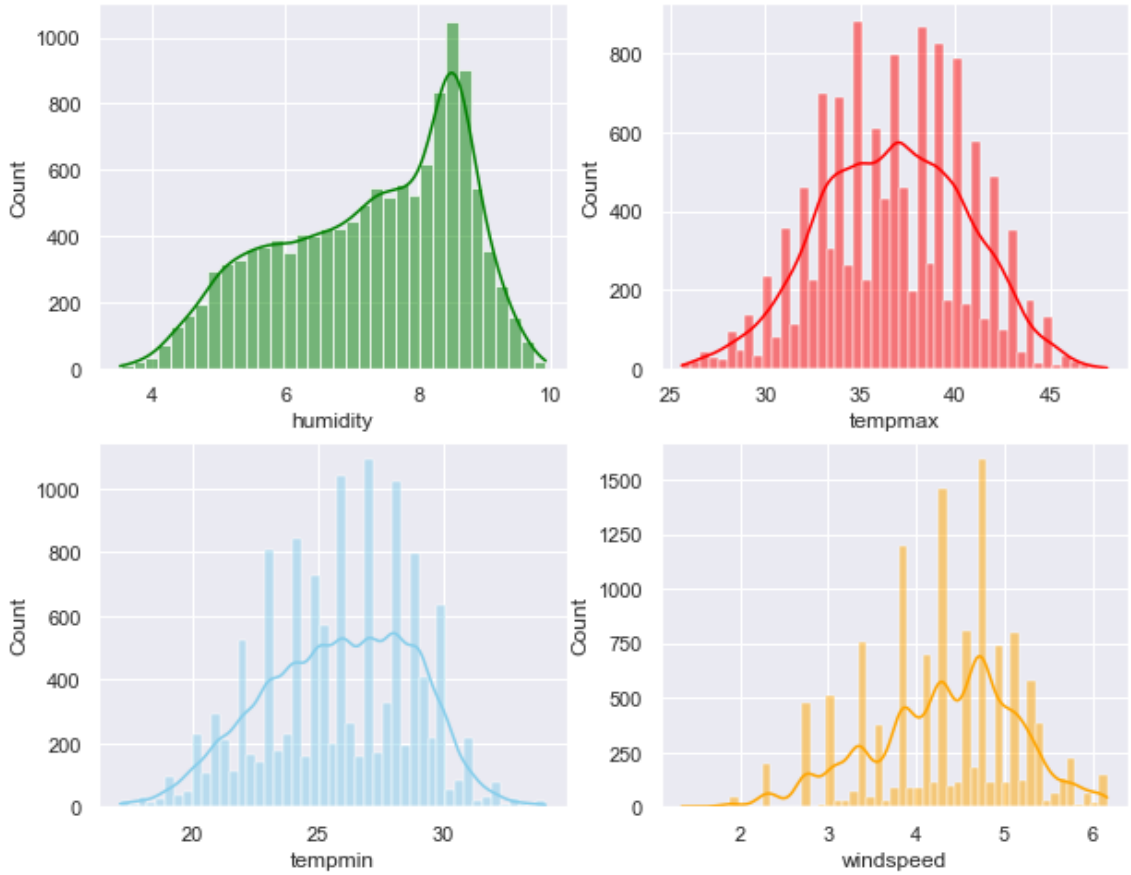
```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
df=df[~((df<(Q1-1.5*IQR))|(df>(Q3+1.5*IQR))).any(axis=1)]
```

In [56]:

```
df.humidity=np.sqrt(df.humidity)
df.windspeed=np.sqrt(df.windspeed)
```

In [57]:

```
sns.set(style="darkgrid")
fig,axs=plt.subplots(2,2,figsize=(10,8))
sns.histplot(data=df,x="humidity",kde=True,ax=axs[0,0],color='green')
sns.histplot(data=df,x="tempmax",kde=True,ax=axs[0,1],color='red')
sns.histplot(data=df,x="tempmin",kde=True,ax=axs[1,0],color='skyblue')
sns.histplot(data=df,x="windspeed",kde=True,ax=axs[1,1],color='orange')
plt.show()
```



In [58]:

```
df1 = df.drop(['City'], axis = 1)
```

In [59]:

```
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [60]:

```
lc=LabelEncoder()
df1["conditions"]=lc.fit_transform(df1["conditions"])
```

In [61]:

```
x=((df1.loc[:,df1.columns!="conditions"]).astype(int)).values[:,0:]
y=df1["conditions"].values
```

In [62]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [63]:

```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
print("KNN Accuracy:{:.2f}%".format(knn.score(x_test,y_test)*100))
```

```
KNN Accuracy:85.60%
```

In [64]:

```
svm=SVC()
svm.fit(x_train,y_train)
print("SVM Accuracy:{:.2f}%".format(svm.score(x_test,y_test)*100))
```

```
SVM Accuracy:84.09%
```