In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('final.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	serial	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew
0	0	34.053151	24.478082	28.709863	39.757808	25.317808	32.306301	22.971233
1	1	34.086179	25.694309	29.464228	41.338211	28.140650	34.423577	23.484553
2	2	34.573984	25.417886	29.526829	40.464228	26.560163	33.085366	22.580488
3	3	33.020325	25.080488	28.727642	37.878049	26.193496	31.772358	21.752033
4	4	30.660976	24.230894	26.774797	36.586992	24.263415	28.943902	24.214634

5 rows × 26 columns

In [4]:

```
df.tail()
```

Out[4]:

	serial	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	p
597	597	32.3	24.4	28.5	35.9	24.4	31.2	23.3	75.0	
598	598	32.7	26.4	29.3	36.3	26.4	32.5	22.6	68.5	
599	599	33.0	26.3	29.8	40.5	26.3	34.5	23.9	71.1	
600	600	35.1	26.8	30.6	42.9	29.1	35.1	23.3	65.9	
601	601	34.0	26.3	30.2	38.1	26.3	33.1	21.5	60.4	

5 rows × 26 columns

In [5]:

```
df = df.drop('serial', axis = 1)
```

In [6]:

```
df.shape
```

Out[6]:

```
(602, 25)
```

In [7]:

```
df.columns
```

Out[7]:

```
Index(['tempmax', 'tempmin', 'temp', 'feelslikemax', 'feelslikemin',
       'feelslike', 'dew', 'humidity', 'precip', 'precipprob', 'precipcove
r',
       'snow', 'snowdepth', 'windspeed', 'winddir', 'sealevelpressure',
       'cloudcover', 'visibility', 'solarradiation', 'solarenergy', 'uvind
ex',
       'conditions', 'stations', 'cases', 'labels'],
      dtype='object')
```

In [8]:

```
df.duplicated().sum()
```

Out[8]:

```
0
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
tempmax            0
tempmin            0
temp               0
feelslikemax       0
feelslikemin       0
feelslike          0
dew                0
humidity           0
precip             0
precipprob         0
precipcover        0
snow               0
snowdepth          0
windspeed          0
winddir            0
sealevelpressure   0
cloudcover         0
visibility         0
solarradiation     0
solarenergy        0
uvindex            0
conditions         0
stations           0
cases              0
labels             0
dtype: int64
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 602 entries, 0 to 601
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   tempmax           602 non-null    float64
 1   tempmin           602 non-null    float64
 2   temp              602 non-null    float64
 3   feelslikemax      602 non-null    float64
 4   feelslikemin      602 non-null    float64
 5   feelslike         602 non-null    float64
 6   dew               602 non-null    float64
 7   humidity          602 non-null    float64
 8   precip            602 non-null    float64
 9   precipprob        602 non-null    float64
 10  precipcover       602 non-null    float64
 11  snow              602 non-null    int64  
 12  snowdepth         602 non-null    int64  
 13  windspeed         602 non-null    float64
 14  winddir           602 non-null    float64
 15  sealevelpressure  602 non-null    float64
 16  cloudcover        602 non-null    float64
 17  visibility        602 non-null    float64
 18  solarradiation    602 non-null    float64
 19  solarenergy       602 non-null    float64
 20  uvindex           602 non-null    float64
 21  conditions        602 non-null    float64
 22  stations          602 non-null    float64
 23  cases             602 non-null    int64  
 24  labels            602 non-null    object 
dtypes: float64(21), int64(3), object(1)
memory usage: 117.7+ KB
```

In [11]:

```
df.describe()
```

Out[11]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew
count	602.000000	602.000000	602.000000	602.000000	602.000000	602.000000	602.000000
mean	31.918079	24.588318	27.813181	38.476069	25.613154	31.485111	23.984349
std	2.737215	2.727919	2.412416	4.776400	4.178797	4.672951	2.668014
min	25.000000	12.740000	18.820000	25.000000	12.360000	18.626667	4.480000
25%	30.025000	23.200000	26.500000	35.600000	23.200000	28.100000	23.300000
50%	31.700000	25.000000	27.900000	38.400000	25.000000	31.500000	24.529268
75%	33.600000	26.500000	29.448171	41.900000	26.600000	34.875000	25.475000
max	41.200000	29.400000	33.300000	49.600000	37.900000	42.900000	28.100000

8 rows × 24 columns

In [12]:

```
df.nunique()
```

Out[12]:

```
tempmax             118
tempmin             126
temp                125
feelslikemax        192
feelslikemin        157
feelslike           202
dew                 123
humidity            295
precip              207
precipprob           17
precipcover          35
snow                  1
snowdepth             1
windspeed           165
winddir             539
sealevelpressure    221
cloudcover          408
visibility           77
solarradiation      523
solarenergy         196
uvindex              24
conditions           20
stations             17
cases               578
labels                1
dtype: int64
```

In [13]:

```
for i in range(df.shape[0]):
    if df.loc[i,'cases']>19000:
        df.loc[i,'labels']='Severe Risk'
    elif df.loc[i,'cases']>10000:
        df.loc[i,'labels']='High Risk'
    elif df.loc[i,'cases']>5000:
        df.loc[i,'labels']='Moderate Risk'
    elif df.loc[i,'cases']>1000:
        df.loc[i,'labels']='Low Risk'
    else:
        df.loc[i,'labels']='Minimal to No risk'
```

In [14]:

```
df.nunique()
```

Out[14]:

```
tempmax            118
tempmin            126
temp               125
feelslikemax       192
feelslikemin       157
feelslike          202
dew                123
humidity           295
precip             207
precipprob          17
precipcover         35
snow                 1
snowdepth            1
windspeed          165
winddir            539
sealevelpressure   221
cloudcover         408
visibility          77
solarradiation     523
solarenergy        196
uvindex             24
conditions          20
stations            17
cases              578
labels               5
dtype: int64
```

In [15]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [17]:

```
import numpy as np
```

In [18]:

```
df['labels'].unique()
```

Out[18]:

```
array(['Low Risk', 'Moderate Risk', 'High Risk', 'Minimal to No risk',
       'Severe Risk'], dtype=object)
```

In [19]:

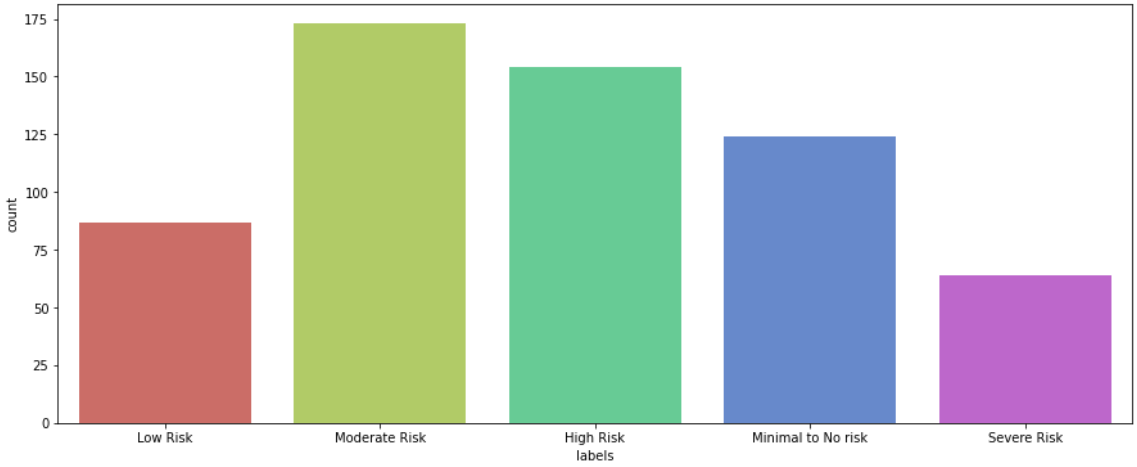
```
df['labels'].value_counts()
```

Out[19]:

```
Moderate Risk         173
High Risk             154
Minimal to No risk    124
Low Risk               87
Severe Risk            64
Name: labels, dtype: int64
```

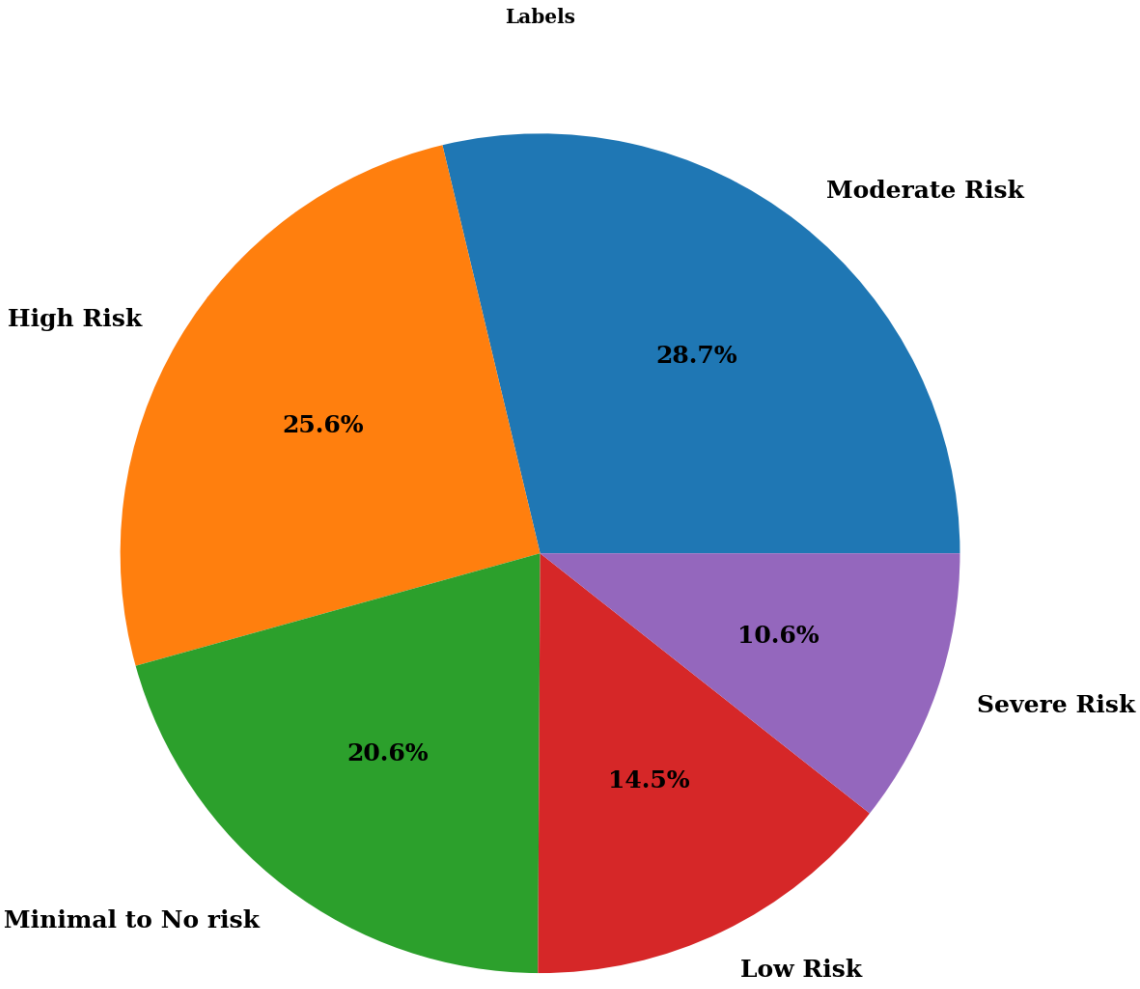
In [20]:

```
plt.figure(figsize=(15,6))
sns.countplot(df['labels'], data = df, palette = 'hls')
plt.show()
```



In [21]:

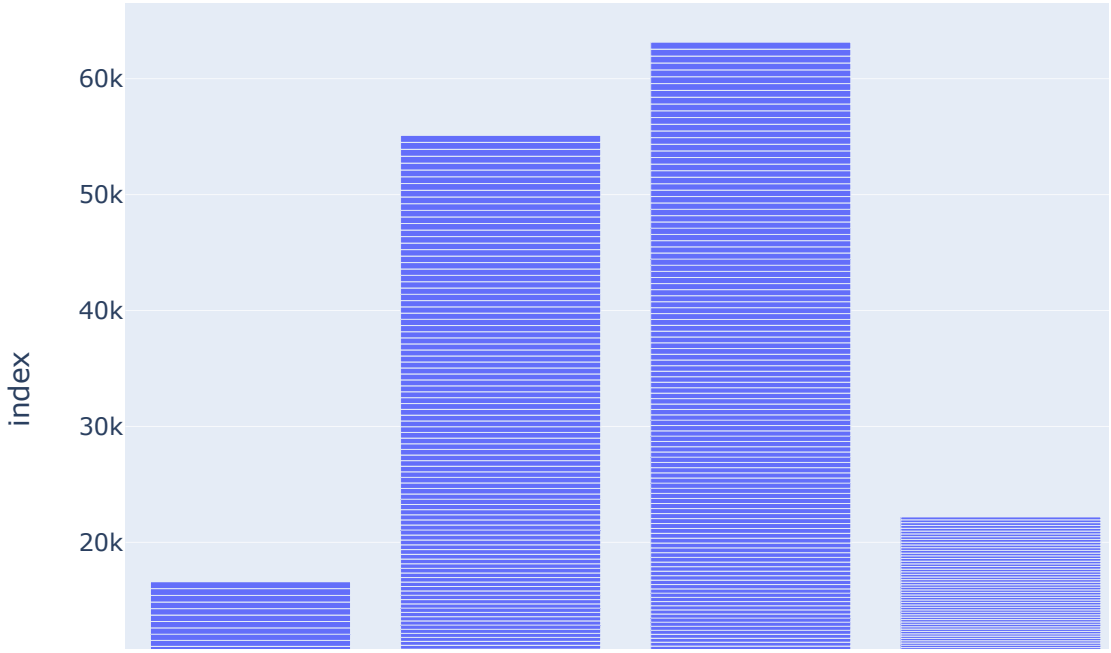
```
plt.figure(figsize=(30,20))
plt.pie(df['labels'].value_counts(), labels=df['labels'].value_counts().index, autopct='
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont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



In [22]:

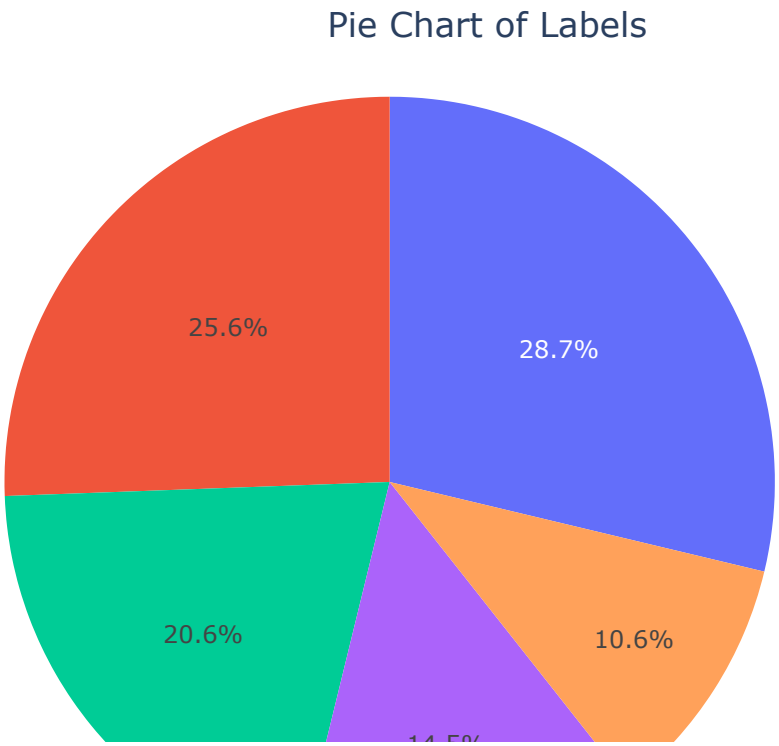
```
import plotly.express as px

fig = px.bar(df, x="labels", y= df.index)
fig.show()
```



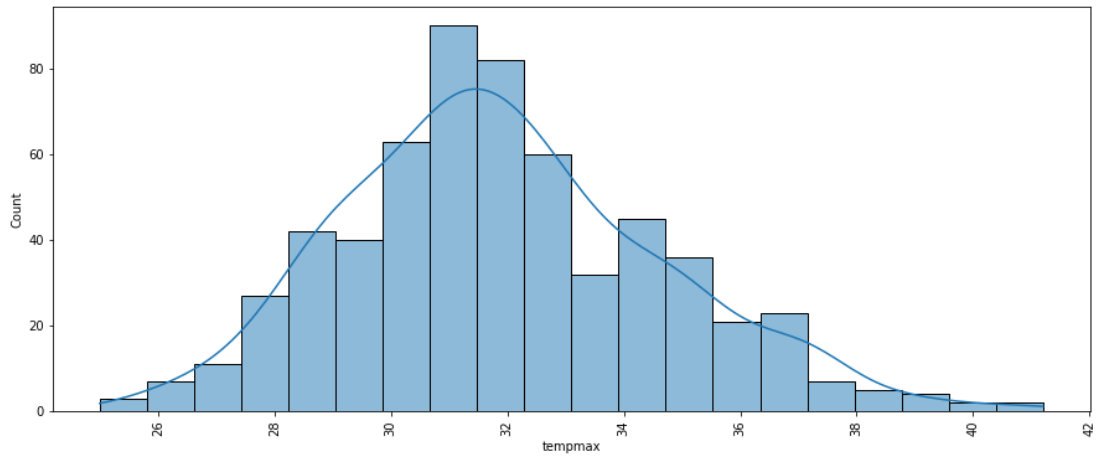
In [23]:

```
value_counts = df['labels'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Labels',
    title_x=0.5
)
fig.show()
```



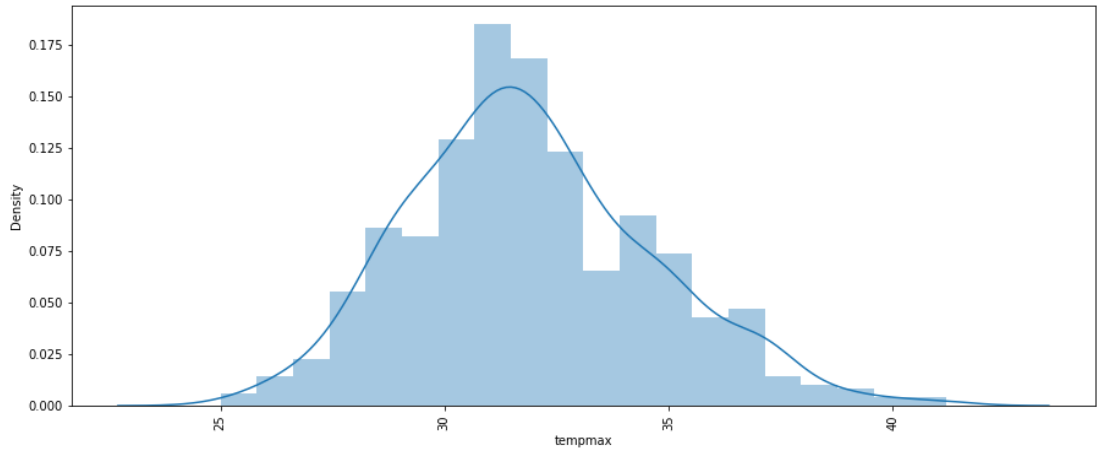
In [24]:

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



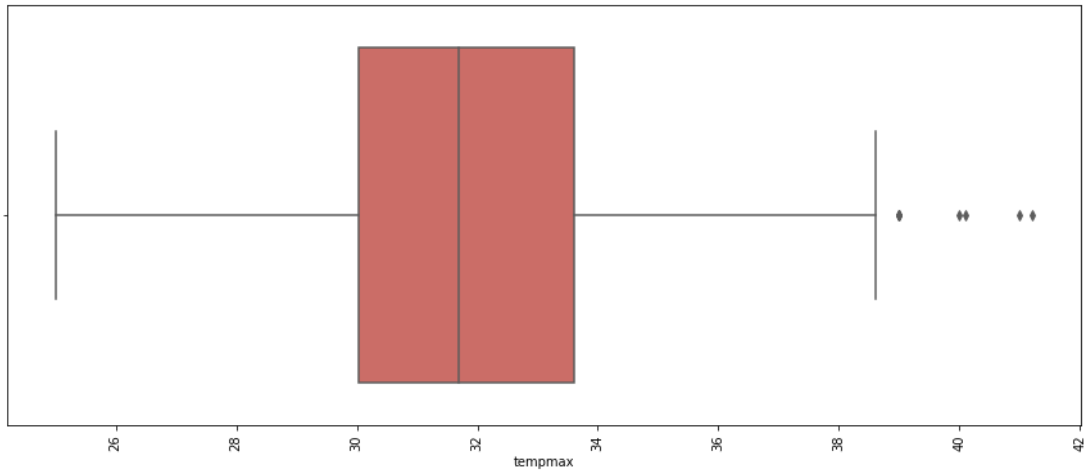
In [25]:

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



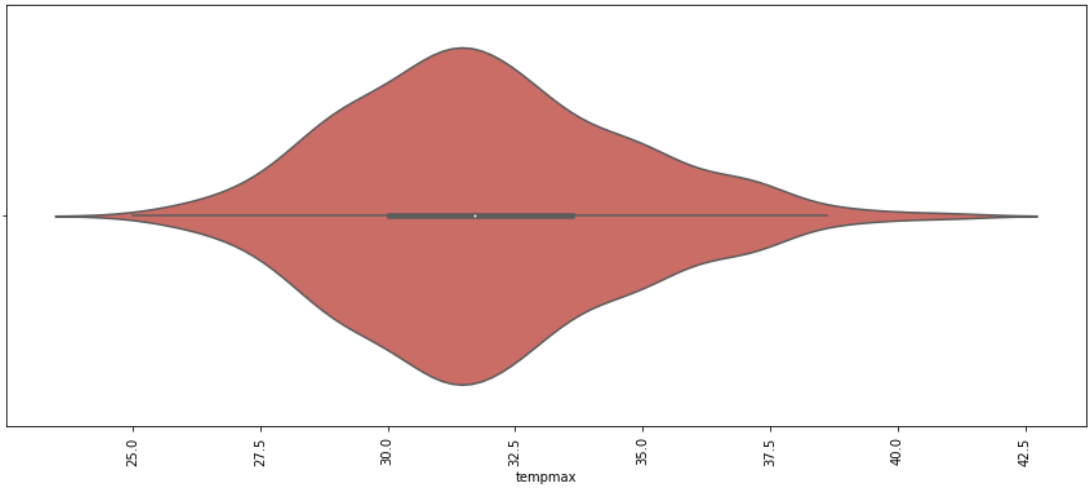
In [26]:

```
for i in df.columns:
    if i != 'labels':
        plt.figure(figsize=(15,6))
        sns.boxplot(df[i], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



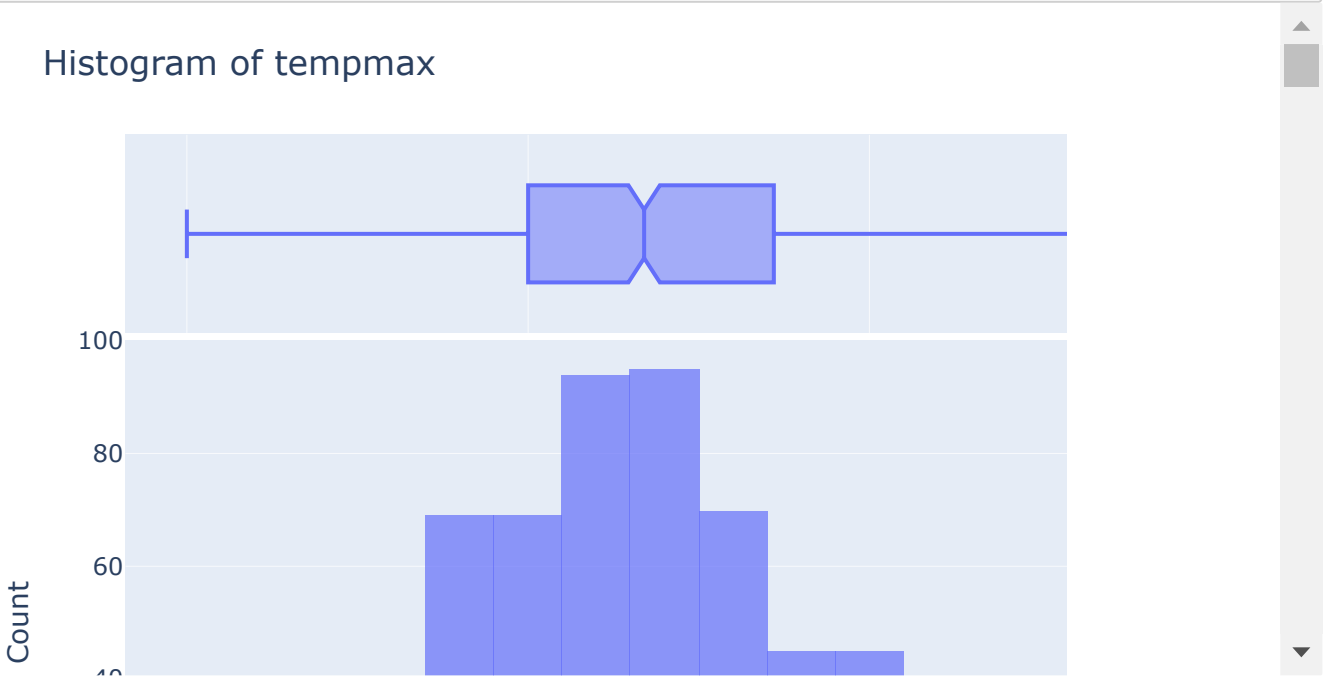
In [27]:

```
for i in df.columns:
    if i != 'labels':
        plt.figure(figsize=(15,6))
        sns.violinplot(df[i], data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



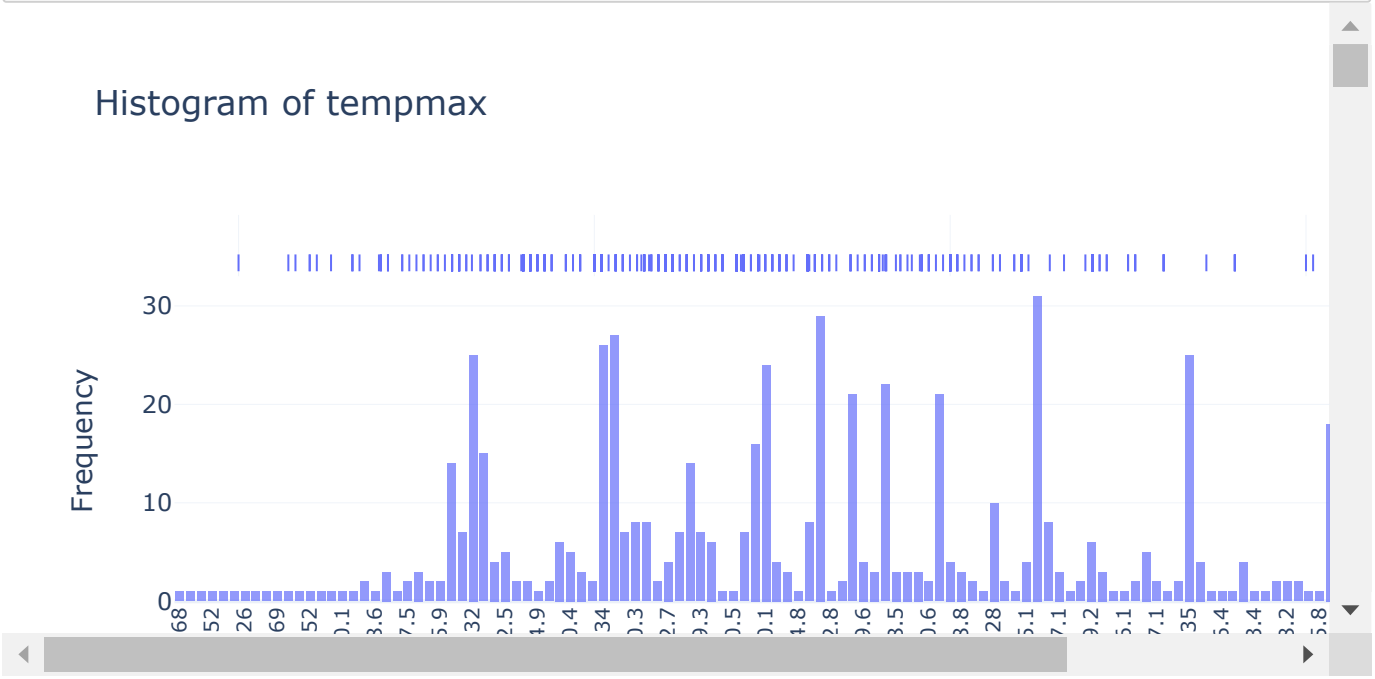
In [28]:

```
for i in df.columns:
    if i != 'labels':
        fig = px.histogram(df[i], nbins=20, color_discrete_sequence=['#636EFA'],
                           marginal='box', opacity=0.7)
        fig.update_layout(
            title=f'Histogram of {i}',
            xaxis_title=i,
            yaxis_title='Count',
            showlegend=False
        )
        fig.show()
```



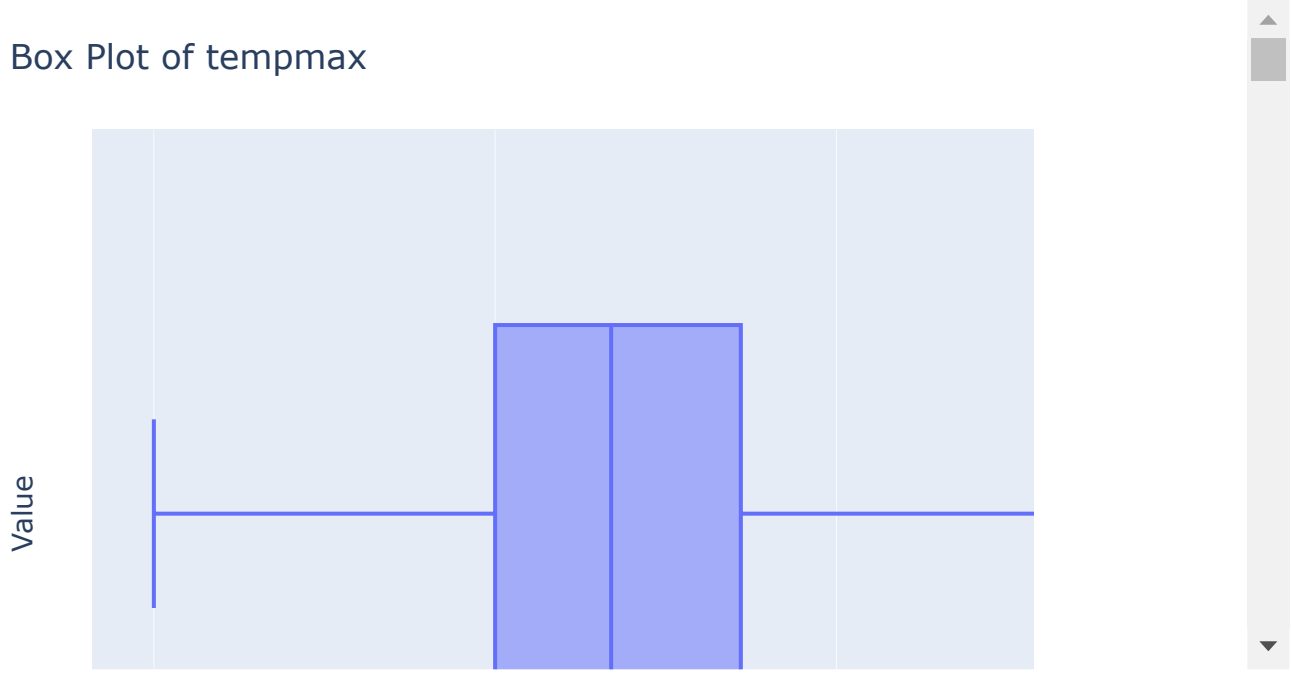
In [29]:

```
for i in df.columns:
    if i != 'labels':
        fig = px.histogram(df, x=i, color_discrete_sequence=['#636EFA'], nbins=20, opaci
                           title=f'Histogram of {i}', labels={'x': i, 'count': 'Frequenc
                           template='plotly_white', width=800, height=400,
                           marginal='rug', barmode='overlay',
                           hover_data=[df['labels']])
        fig.update_layout(xaxis=dict(type='category', title=i,
                                     tickangle=-90, tickfont=dict(size=10)),
                          yaxis_title='Frequency', showlegend=False)
        fig.show()
```



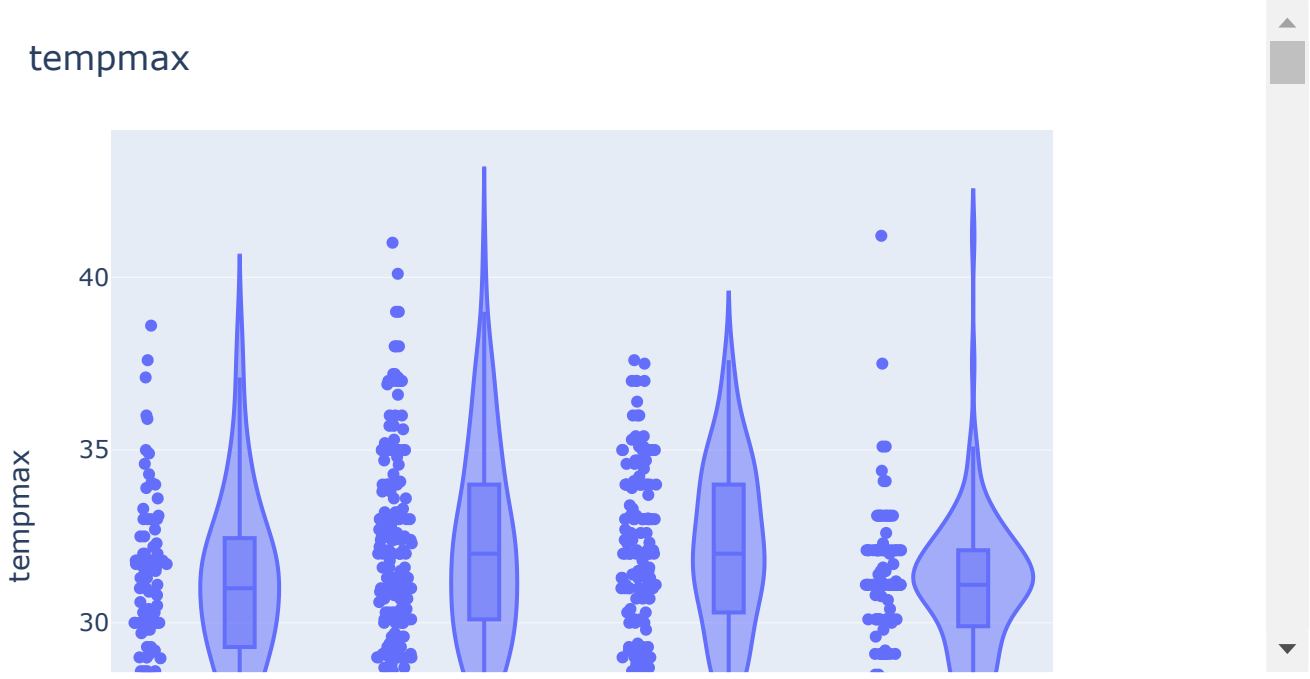
In [30]:

```
for i in df.columns:
    if i != 'labels':
        fig = px.box(df, x=df[i], color_discrete_sequence=['#636EFA'])
        fig.update_layout(
            title='Box Plot of ' + i,
            xaxis_title=i,
            yaxis_title='Value',
        )
        fig.show()
```



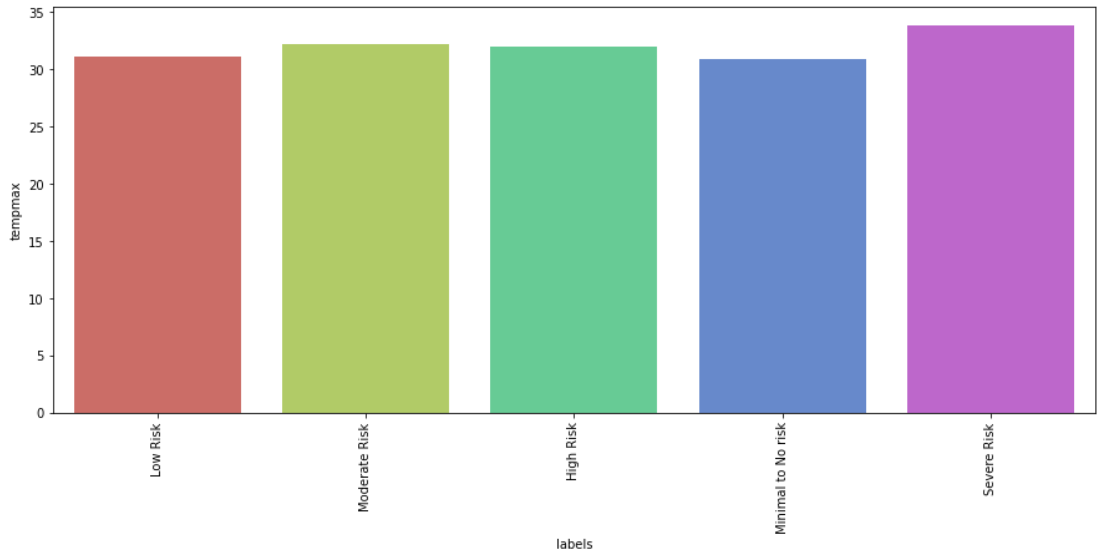
In [31]:

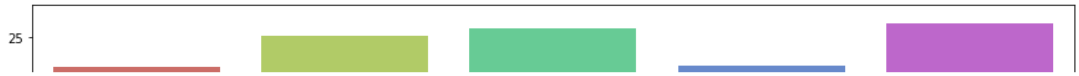
```
for i in df.columns:
    if i != 'labels':
        fig = px.violin(df, y=i, x='labels', box=True, points="all", hover_data=df.colum
        fig.update_layout(title=i, xaxis_title="Labels", yaxis_title=i)
        fig.show()
```



In [32]:

```
for i in df.columns:
    if i != 'labels':
        plt.figure(figsize=(15,6))
        sns.barplot(x = df['labels'], y = df[i], data = df, ci = None, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```



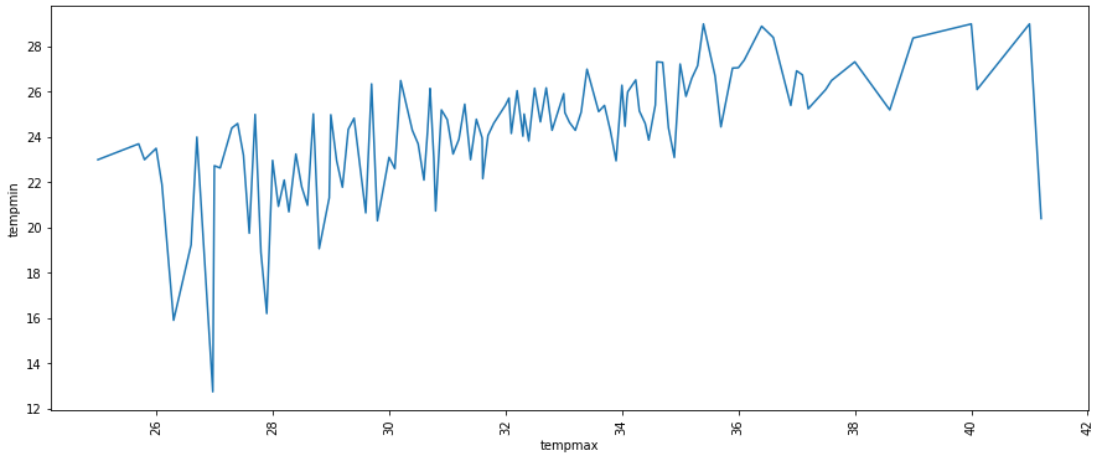


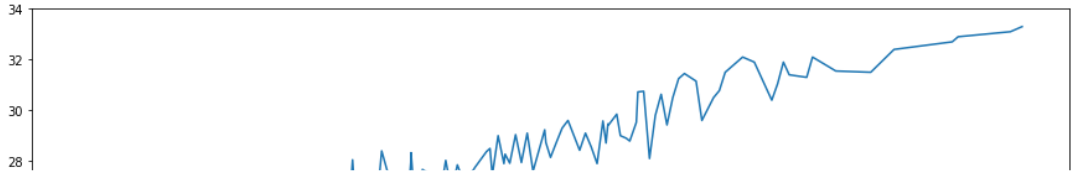
In [33]:

```
for i in df.columns:
    if i != 'labels':
        fig = px.bar(df, x="labels", y=i, color="labels", barmode="group")
        fig.update_layout(
            title=f"{i} by Labels",
            xaxis_title="Labels",
            yaxis_title=i,
            legend_title="Labels"
        )
        fig.show()
```

In [34]:

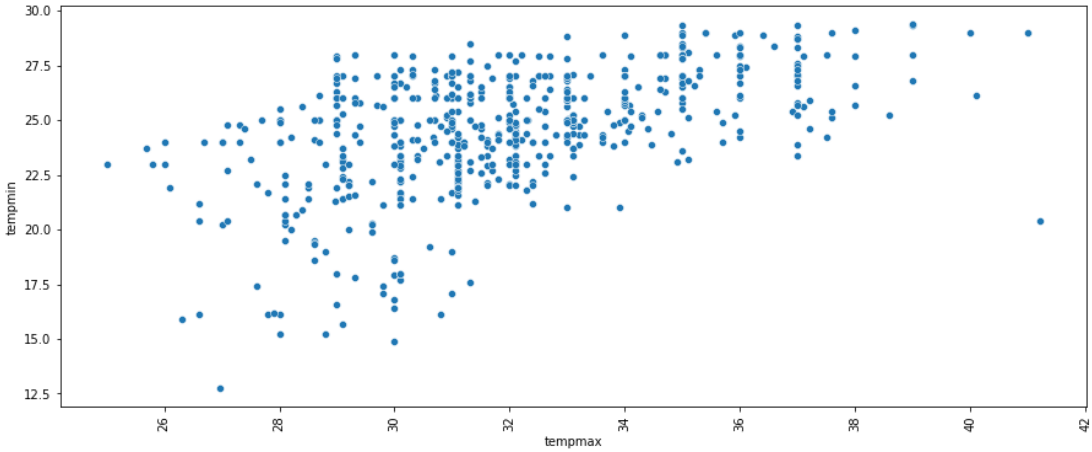
```
for i in df.columns:
    for j in df.columns:
        if i != j:
            if i != 'labels':
                plt.figure(figsize=(15,6))
                sns.lineplot(x = df[i], y = df[j], data = df, ci = None, palette = 'hls'
                plt.xticks(rotation = 90)
                plt.show()
```

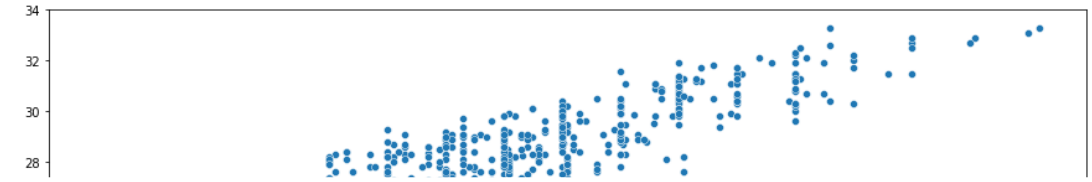




In [35]:

```
for i in df.columns:
    for j in df.columns:
        if i != j:
            if i != 'labels':
                plt.figure(figsize=(15,6))
                sns.scatterplot(x = df[i], y = df[j], data = df, ci = None, palette = 'h
                plt.xticks(rotation = 90)
                plt.show()
```

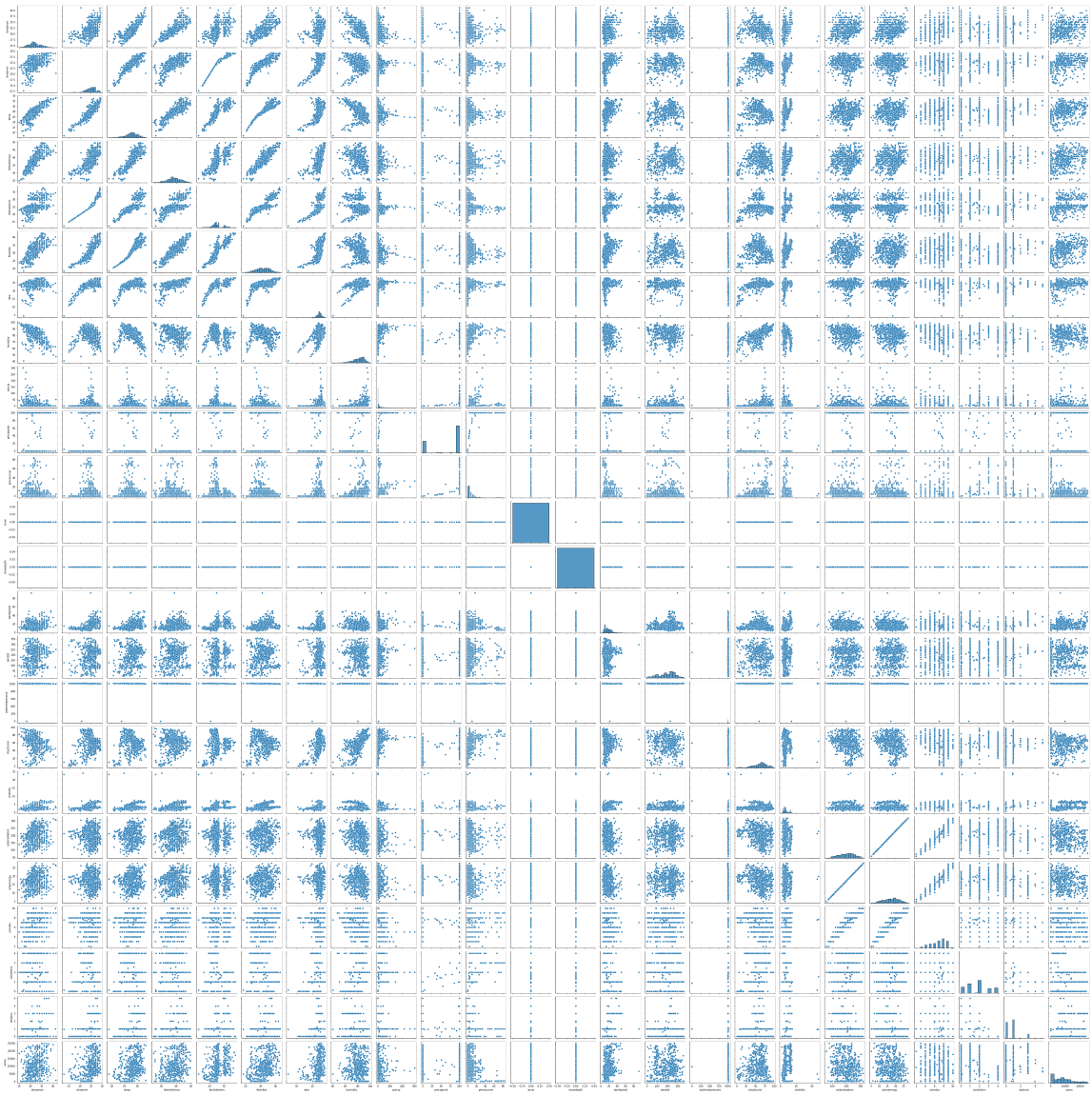




In [37]:

```
plt.figure(figsize=(15,6))
sns.pairplot(data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```

```
<Figure size 1080x432 with 0 Axes>
```



In [38]:

```
df_corr = df.corr()
```

In [39]:

```
df_corr
```

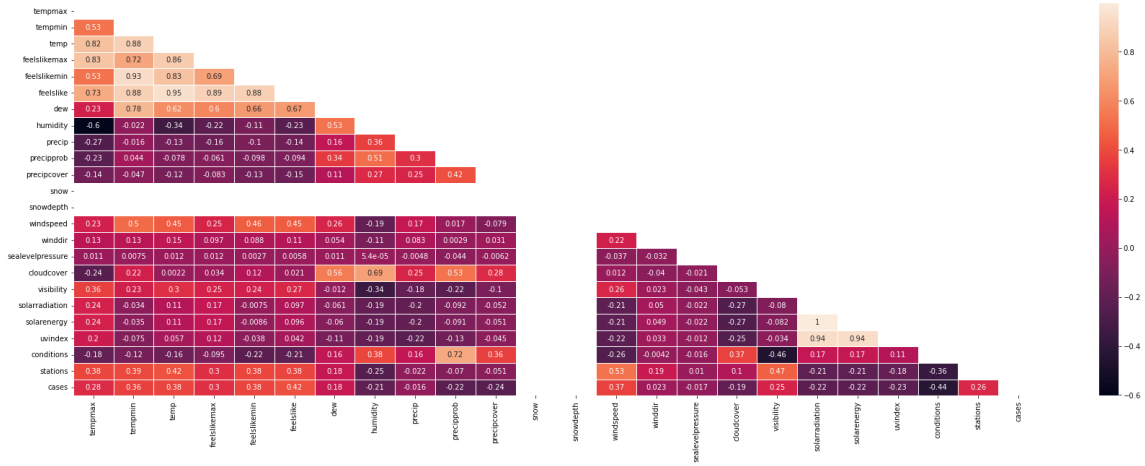
Out[39]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	
tempmax	1.000000	0.527246	0.820941	0.832811	0.526838	0.732782	0.228
tempmin	0.527246	1.000000	0.881988	0.716619	0.931412	0.883030	0.784
temp	0.820941	0.881988	1.000000	0.863626	0.829122	0.949866	0.621
feelslikemax	0.832811	0.716619	0.863626	1.000000	0.693604	0.891305	0.596
feelslikemin	0.526838	0.931412	0.829122	0.693604	1.000000	0.883670	0.664
feelslike	0.732782	0.883030	0.949866	0.891305	0.883670	1.000000	0.671
dew	0.228436	0.784664	0.621103	0.596466	0.664503	0.671107	1.000
humidity	-0.602729	-0.022102	-0.335037	-0.219550	-0.112346	-0.231987	0.525
precip	-0.270202	-0.015915	-0.130827	-0.155520	-0.099895	-0.138739	0.155
precipprob	-0.226290	0.043788	-0.077521	-0.061133	-0.097888	-0.093543	0.337
precipcover	-0.139013	-0.047444	-0.118306	-0.083107	-0.131179	-0.150176	0.108
snow	NaN	NaN	NaN	NaN	NaN	NaN	N
snowdepth	NaN	NaN	NaN	NaN	NaN	NaN	N
windspeed	0.234752	0.502966	0.451224	0.254755	0.460448	0.448998	0.255
winddir	0.131311	0.130978	0.151874	0.096730	0.087622	0.114765	0.053
sealevelpressure	0.011288	0.007517	0.011662	0.011925	0.002738	0.005758	0.010
cloudcover	-0.237572	0.223664	0.002212	0.034238	0.115816	0.021153	0.557
visibility	0.358248	0.225410	0.299479	0.245154	0.235159	0.273086	-0.012
solarradiation	0.239410	-0.034087	0.114995	0.167350	-0.007501	0.096916	-0.060
solarenergy	0.239365	-0.034765	0.114794	0.167075	-0.008587	0.096444	-0.060
uvindex	0.198627	-0.075002	0.056545	0.115035	-0.038326	0.042182	-0.109
conditions	-0.184010	-0.115677	-0.157189	-0.094710	-0.218707	-0.209585	0.159
stations	0.381730	0.388823	0.421828	0.295325	0.382937	0.376258	0.175
cases	0.284181	0.364691	0.378149	0.304087	0.376015	0.415344	0.179

24 rows × 24 columns

In [41]:

```
plt.figure(figsize=(30, 10))
matrix = np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap="rocket");
plt.show()
```



In [44]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()


df['labels']=le.fit_transform(df['labels'])
X=df.iloc[:,:-1]
y=df['labels']
```

In [45]:

```
X
```

Out[45]:

	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	hum
0	34.053151	24.478082	28.709863	39.757808	25.317808	32.306301	22.971233	73.50
1	34.086179	25.694309	29.464228	41.338211	28.140650	34.423577	23.484553	72.06
2	34.573984	25.417886	29.526829	40.464228	26.560163	33.085366	22.580488	69.42
3	33.020325	25.080488	28.727642	37.878049	26.193496	31.772358	21.752033	69.29
4	30.660976	24.230894	26.774797	36.586992	24.263415	28.943902	24.214634	86.65
...	...	...	...	...	...	...	...	
597	32.300000	24.400000	28.500000	35.900000	24.400000	31.200000	23.300000	75.00
598	32.700000	26.400000	29.300000	36.300000	26.400000	32.500000	22.600000	68.50
599	33.000000	26.300000	29.800000	40.500000	26.300000	34.500000	23.900000	71.10
600	35.100000	26.800000	30.600000	42.900000	29.100000	35.100000	23.300000	65.90
601	34.000000	26.300000	30.200000	38.100000	26.300000	33.100000	21.500000	60.40

602 rows × 24 columns

In [46]:

```
from sklearn.model_selection import train_test_split
```

In [47]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify = df['
                                                    random_state=42)
```

In [48]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [49]:

```
reg=RandomForestRegressor()
reg.fit(X_train,y_train)
```

Out[49]:

```
▾ RandomForestRegressor

RandomForestRegressor()
```

In [50]:

```
y_pred=reg.predict(X_test)
```

In [53]:

```
reg.score(X_test,y_test)
```

Out[53]:

```
0.9943796068075117
```

In [54]:

```
from sklearn.model_selection import cross_val_score
acc=cross_val_score(reg,X_test,y_test,cv=5).mean()
print(acc)
```

```
0.9404900487112752
```