In [2]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 X=pd.read_csv(r"C:\Users\Admin\Downloads\weather_classification_data.cs
4 X
```

Out[2]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season
0	14.0	73	9.5	82.0	partly cloudy	1010.82	2	Winter
1	39.0	96	8.5	71.0	partly cloudy	1011.43	7	Spring
2	30.0	64	7.0	16.0	clear	1018.72	5	Spring
3	38.0	83	1.5	82.0	clear	1026.25	7	Spring
4	27.0	74	17.0	66.0	overcast	990.67	1	Winter
...	...	...	...	...	...	...	...	...
13195	10.0	74	14.5	71.0	overcast	1003.15	1	Summer
13196	-1.0	76	3.5	23.0	cloudy	1067.23	1	Winter
13197	30.0	77	5.5	28.0	overcast	1012.69	3	Autumn
13198	3.0	76	10.0	94.0	overcast	984.27	0	Winter
13199	-5.0	38	0.0	92.0	overcast	1015.37	5	Autumn

13200 rows × 11 columns

In [3]:
```
1 pd.set_option('Display.max_columns',50)
```

In [4]:
```
1 X['Temperature C']=X['Temperature']-32*5/9
```

In [5]:
```
1 X
```

Out[5]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season
0	14.0	73	9.5	82.0	partly cloudy	1010.82	2	Winter
1	39.0	96	8.5	71.0	partly cloudy	1011.43	7	Spring
2	30.0	64	7.0	16.0	clear	1018.72	5	Spring
3	38.0	83	1.5	82.0	clear	1026.25	7	Spring
4	27.0	74	17.0	66.0	overcast	990.67	1	Winter
...	...	...	...	...	...	...	...	...
13195	10.0	74	14.5	71.0	overcast	1003.15	1	Summer
13196	-1.0	76	3.5	23.0	cloudy	1067.23	1	Winter
13197	30.0	77	5.5	28.0	overcast	1012.69	3	Autumn
13198	3.0	76	10.0	94.0	overcast	984.27	0	Winter
13199	-5.0	38	0.0	92.0	overcast	1015.37	5	Autumn

13200 rows × 12 columns

In [5]:
```
1 X.drop(columns='Temperature',inplace=True)
```

In [6]:
```
1 X
```

Out[6]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	ir
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	ir
2	64	7.0	16.0	clear	1018.72	5	Spring	5.5	mou
3	83	1.5	82.0	clear	1026.25	7	Spring	1.0	co
4	74	17.0	66.0	overcast	990.67	1	Winter	2.5	mou
...	...	...	...	...	...	...	...	...	
13195	74	14.5	71.0	overcast	1003.15	1	Summer	1.0	mou
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	co
13197	77	5.5	28.0	overcast	1012.69	3	Autumn	9.0	co
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	ir
13199	38	0.0	92.0	overcast	1015.37	5	Autumn	10.0	mou

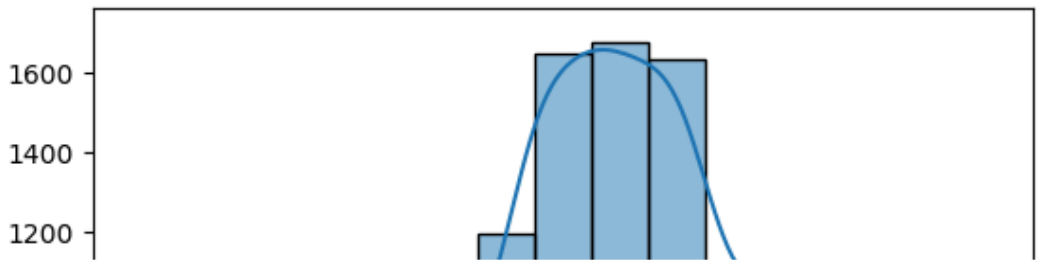
13200 rows × 11 columns

In [7]:
```
X.isnull().sum()
```

Out[7]:
```
Humidity                0
Wind Speed              0
Precipitation (%)       0
Cloud Cover             0
Atmospheric Pressure    0
UV Index                0
Season                  0
Visibility (km)         0
Location                0
Weather Type            0
Temperature C           0
dtype: int64
```

In [8]:
```
import seaborn as sns
for i in X.columns:
    if (X[i].dtypes=="int64")|(X[i].dtypes=="float64"):
        sns.histplot(X[i],bins=15,kde=True)
        print(i.upper())
        print("mean",(X[i].mean()))
        print("median",(X[i].median()))
        print("mode",(X[i].mode()[0]))
        print("min",(X[i].min()))
        print("max",(X[i].max()))

        plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):

HUMIDITY
mean 68.71083333333333
median 70.0
mode 76
min 20
max 109
```



In [103]:
```
A=X.loc[X['Location']=='inland']# INLAND AREA
A
```

Out[103]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	ir
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	ir
5	55	3.5	26.0	overcast	1010.03	2	Summer	5.0	ir
6	97	8.0	86.0	overcast	990.87	1	Winter	4.0	ir
7	85	6.0	96.0	partly cloudy	984.46	1	Winter	3.5	ir
...	...	...	...	...	...	...	...	...	
13189	49	7.5	11.0	clear	1022.86	7	Summer	9.5	ir
13191	48	6.5	14.0	clear	1029.37	8	Summer	8.0	ir
13192	24	8.0	5.0	clear	1029.61	8	Summer	9.0	ir
13193	65	15.5	50.0	overcast	982.57	1	Winter	5.0	ir
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	ir

4816 rows × 11 columns

In [10]:
```
A['Temperature C'].max()
```

Out[10]:
```
91.22222222222223
```

In [11]:
```
A['Cloud Cover'].value_counts()
```

Out[11]:
```
Cloud Cover
overcast         2391
partly cloudy    1616
clear             686
cloudy            123
Name: count, dtype: int64
```

In [12]:
```
A['Season'].value_counts()
```

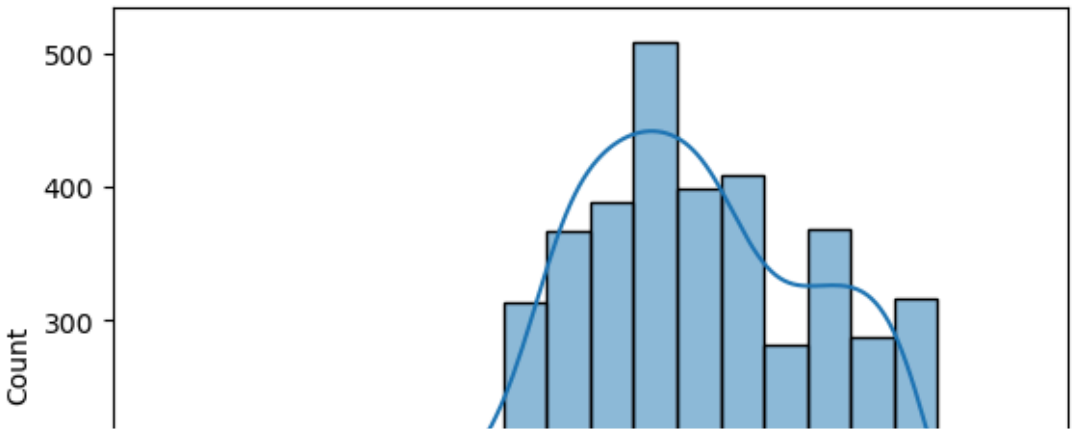
Out[12]:
```
Season
Winter    2372
Spring     859
Autumn     794
Summer     791
Name: count, dtype: int64
```

In [13]:
```
A['Weather Type'].value_counts()
```

Out[13]:
```
Weather Type
Snowy     1575
Cloudy    1107
Rainy     1069
Sunny     1065
Name: count, dtype: int64
```

In [14]:
```
import seaborn as sns
for j in A.columns:
    if(A[j].dtype=="int64")|(A[j].dtype=="float64"):
        print(j.upper())
        sns.histplot(A[j],bins=20,kde=True)
        plt.show()
```

```
HUMIDITY

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [15]:
```
A['Wind Speed'].mean()
```

Out[15]:
```
10.076308139534884
```

In [104]:
```
B1=A.loc[A['Season']=='Winter'] #WINTER:SNOWY
B1
```

Out[104]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	in
6	97	8.0	86.0	overcast	990.87	1	Winter	4.0	in
7	85	6.0	96.0	partly cloudy	984.46	1	Winter	3.5	in
13	87	15.0	67.0	overcast	986.19	0	Winter	1.5	in
21	88	12.5	98.0	overcast	980.31	1	Winter	3.0	in
...	...	...	...	...	...	...	...	...	
13177	94	8.0	50.0	overcast	987.63	1	Winter	4.0	in
13182	67	11.5	54.0	overcast	980.31	0	Winter	2.0	in
13188	34	3.5	16.0	clear	1022.64	10	Winter	9.5	in
13193	65	15.5	50.0	overcast	982.57	1	Winter	5.0	in
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	in

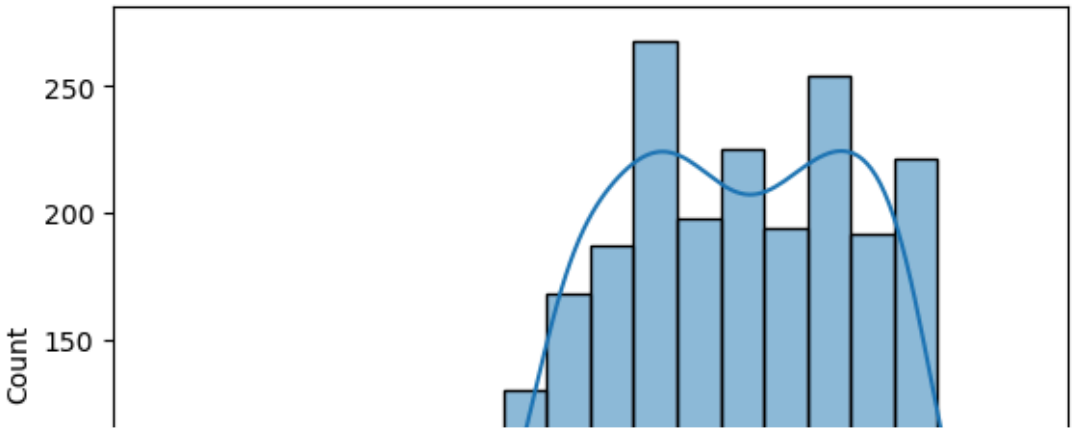
2372 rows × 11 columns

In [19]:
```
B1['Cloud Cover'].value_counts()
```

Out[19]:
```
Cloud Cover
overcast         1534
partly cloudy     630
clear             170
cloudy             38
Name: count, dtype: int64
```

In [20]:
```
import seaborn as sns
for i in B1.columns:
    if(B1[i].dtype=="int64")|(B1[i].dtype=="float64"):
        print(i.upper())
        sns.histplot(B1[i],bins=20,kde=True)
        plt.show()
```

```
HUMIDITY

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [105]:
```
D=A.loc[A['Season']=='Spring']#SPRING:SUNNY
D
```

Out[105]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	in
16	27	7.0	13.0	partly cloudy	1016.38	5	Spring	7.5	in
58	70	0.5	69.0	overcast	1074.07	3	Spring	12.5	in
73	62	6.5	84.0	overcast	1013.61	0	Spring	4.5	in
75	47	5.0	4.0	clear	1027.53	9	Spring	10.0	in
...	...	...	...	...	...	...	...	...	
13150	54	10.0	43.0	overcast	1000.58	1	Spring	5.0	in
13152	59	1.5	14.0	clear	1020.38	8	Spring	5.5	in
13159	72	0.5	17.0	overcast	1006.09	1	Spring	6.5	in
13168	45	7.0	28.0	partly cloudy	894.65	5	Spring	11.5	in
13174	65	2.5	20.0	partly cloudy	1016.81	1	Spring	8.0	in

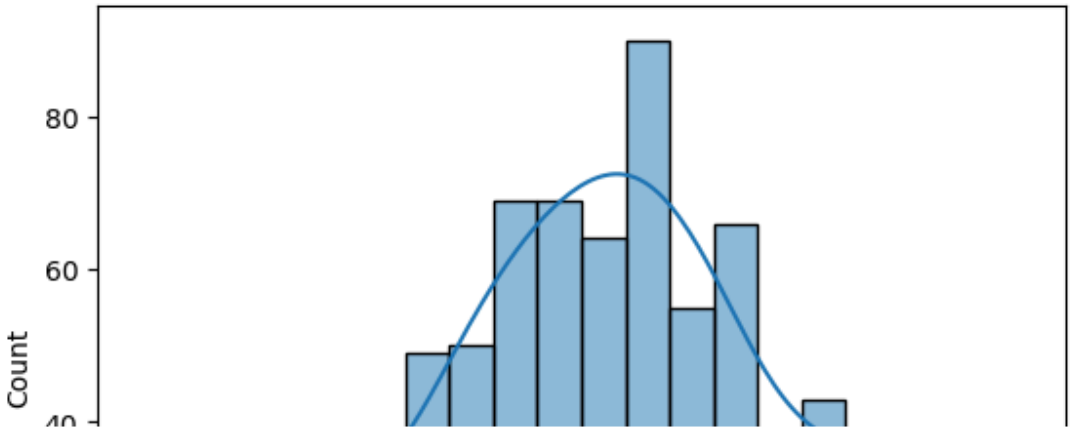
859 rows × 11 columns

In [22]:
```
D['Cloud Cover'].value_counts()
```

Out[22]:
```
Cloud Cover
partly cloudy    336
overcast         292
clear            194
cloudy            37
Name: count, dtype: int64
```

In [23]:
```
import seaborn as sns
for j in D.columns:
    if(D[j].dtype=="int64")|(D[j].dtype=="float64"):
        print(i.upper())
        sns.histplot(D[j],bins=20,kde=True)
        plt.show()
```

```
TEMPERATURE C

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [106]:
```
E=A.loc[A['Season']=='Autumn']#AUTUMN:CLOUDY
E
```

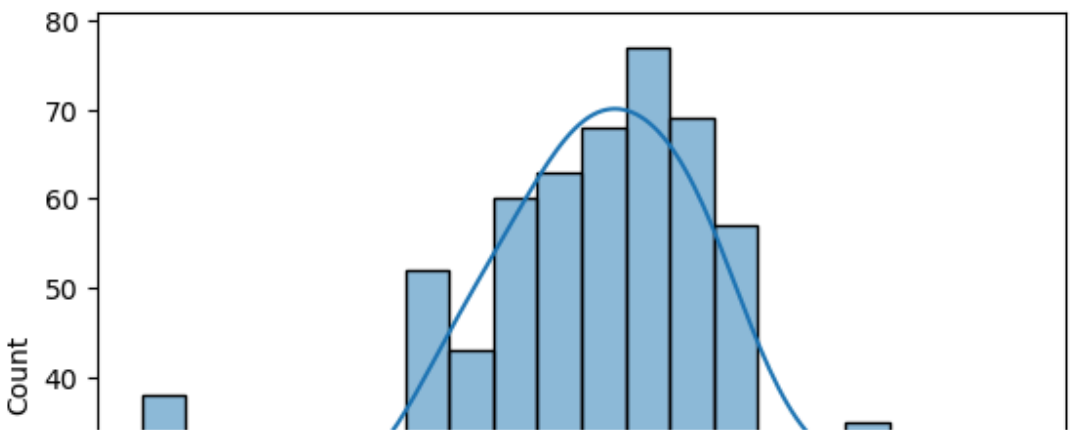
Out[106]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
11	43	2.0	16.0	clear	1029.16	11	Autumn	7.5	in
62	55	1.5	32.0	partly cloudy	1009.01	2	Autumn	5.5	in
94	38	11.0	15.0	partly cloudy	824.71	9	Autumn	12.5	in
134	77	0.5	91.0	cloudy	934.49	14	Autumn	17.0	in
139	50	6.5	21.0	partly cloudy	1013.93	4	Autumn	6.0	in
...	...	...	...	...	...	...	...	...	
13102	89	8.5	80.0	partly cloudy	1010.68	6	Autumn	5.5	in
13124	101	16.0	80.0	partly cloudy	1015.06	11	Autumn	1.5	in
13129	51	0.5	14.0	partly cloudy	1017.63	4	Autumn	6.5	in
13160	27	5.0	6.0	clear	1011.40	9	Autumn	8.0	in
13173	66	8.0	59.0	overcast	1006.99	2	Autumn	5.0	in

794 rows × 11 columns

In [25]:
```
for i in E.columns:
    if(E[i].dtype=="int64")|(E[i].dtype=="float64"):
        print(i.upper())
        sns.histplot(E[i],bins=20,kde=True)
        plt.show()
```

```
HUMIDITY

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [107]:
```
F=A.loc[A['Season']=='Summer']#SUMMER:CLOUDY
F
```

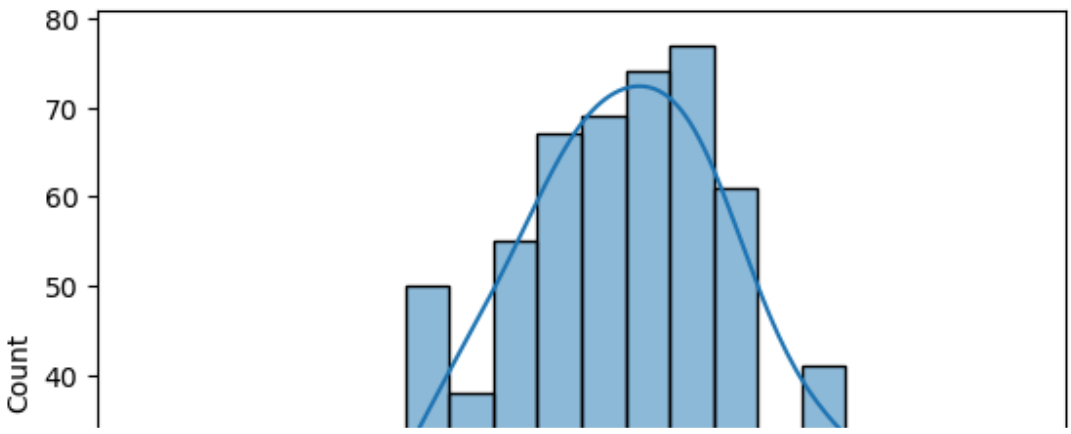
Out[107]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
5	55	3.5	26.0	overcast	1010.03	2	Summer	5.0	ir
19	102	12.0	72.0	clear	1012.25	4	Summer	8.0	ir
23	73	13.5	29.0	overcast	1012.02	1	Summer	9.0	ir
51	55	8.5	37.0	partly cloudy	1014.49	4	Summer	7.5	ir
72	52	6.0	97.0	overcast	1063.39	7	Summer	12.5	ir
...	...	...	...	...	...	...	...	...	
13108	81	10.5	96.0	clear	1022.66	11	Summer	3.5	ir
13136	74	0.0	103.0	partly cloudy	1012.40	10	Summer	5.5	ir
13189	49	7.5	11.0	clear	1022.86	7	Summer	9.5	ir
13191	48	6.5	14.0	clear	1029.37	8	Summer	8.0	ir
13192	24	8.0	5.0	clear	1029.61	8	Summer	9.0	ir

791 rows × 11 columns

In [27]:
```
for i in F.columns:
    if(F[i].dtype=="int64")|(F[i].dtype=="float64"):
        print(i.upper())
        sns.histplot(F[i],bins=20,kde=True)
        plt.show()
```

```
HUMIDITY

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [28]:

```
for i in B1.columns:
    if(B1[i].dtype=="int64")|(B1[i].dtype=="float64"):
        print(i.upper())
        print(B1[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        70     72
1        60     71
2        76     61
3        91     59
4        86     58
5        66     56
-----
WIND SPEED
   Wind Speed  count
0         8.0     76
1         9.0     76
2         2.5     73
3         7.0     69
4         6.5     68
5         1.0     67
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               72.0     50
1               94.0     48
2               89.0     47
3               96.0     46
4               91.0     46
5               71.0     45
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0                992.62      5
1                999.51      5
2                997.36      5
3                985.11      5
4                999.42      4
5                980.85      4
-----
UV INDEX
   UV Index  count
0         1    814
1         0    707
2         3    131
3         2    122
4         4     87
5         8     68
-----
VISIBILITY (KM)
   Visibility (km)  count
0              3.0    224
1              3.5    216
2              1.5    215
3              4.5    214
4              2.0    207
5              2.5    206
-----
TEMPERATURE C
   Temperature C  count
0     -17.777778    110
1     -15.777778    110
2     -16.777778    109
3     -20.777778    108
4     -19.777778    102
```

```
5     -25.777778    101
-----
```

In [29]:

```
for i in D.columns:
    if(D[i].dtype=="int64")|(D[i].dtype=="float64"):
        print(i.upper())
        print(D[i].value_counts().head(6).reset_index().sort_values(by=

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        68     26
1        69     23
2        78     23
3        62     21
4        71     21
5        77     20
-----
WIND SPEED
   Wind Speed  count
0         6.5     40
1         9.5     35
2         6.0     35
3         9.0     31
4         7.0     31
5        10.0     31
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               13.0     30
1               10.0     21
2               14.0     20
3               16.0     19
4                6.0     18
5               19.0     17
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1019.95      3
1               1011.24      3
2               1008.57      3
3               1018.46      3
4               1003.36      3
5               1013.67      3
-----
UV INDEX
   UV Index  count
0         3    128
1         1    124
2         2    118
3         4     67
4         5     61
5         0     56
-----
VISIBILITY (KM)
   Visibility (km)  count
0              7.5     64
1              5.5     64
2              8.0     63
3              6.5     49
4              7.0     47
5              6.0     47
-----
TEMPERATURE C
   Temperature C  count
0       4.222222     37
1       6.222222     36
2      13.222222     34
3      14.222222     34
4       9.222222     33
```

```
5       2.222222      31
-----
```

In [30]:

```
for i in E.columns:
    if(E[i].dtype=="int64")|(E[i].dtype=="float64"):
        print(i.upper())
        print(E[i].value_counts().head(6).reset_index().sort_values(by=
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        65     23
1        62     21
2        69     20
3        76     20
4        71     18
5        78     18
-----
WIND SPEED
   Wind Speed  count
0         3.5     37
1         9.5     32
2         5.0     30
3         1.5     30
4         8.5     28
5         6.5     27
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               10.0     23
1               14.0     21
2               16.0     20
3               19.0     19
4               18.0     16
5               15.0     16
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1006.14      3
1               1001.76      3
2               1000.36      3
3               1010.53      3
4               1006.42      3
5               1018.17      3
-----
UV INDEX
   UV Index  count
0         2    133
1         3    117
2         1     99
3         4     58
4         0     54
5        11     53
-----
VISIBILITY (KM)
   Visibility (km)  count
0              8.0     61
1              6.0     55
2              5.5     55
3              6.5     53
4              7.0     53
5              8.5     48
-----
TEMPERATURE C
   Temperature C  count
0       5.222222     34
1       8.222222     34
2      10.222222     32
3      12.222222     32
4       7.222222     31
```

```
5      14.222222     30
-----
```

In [31]:

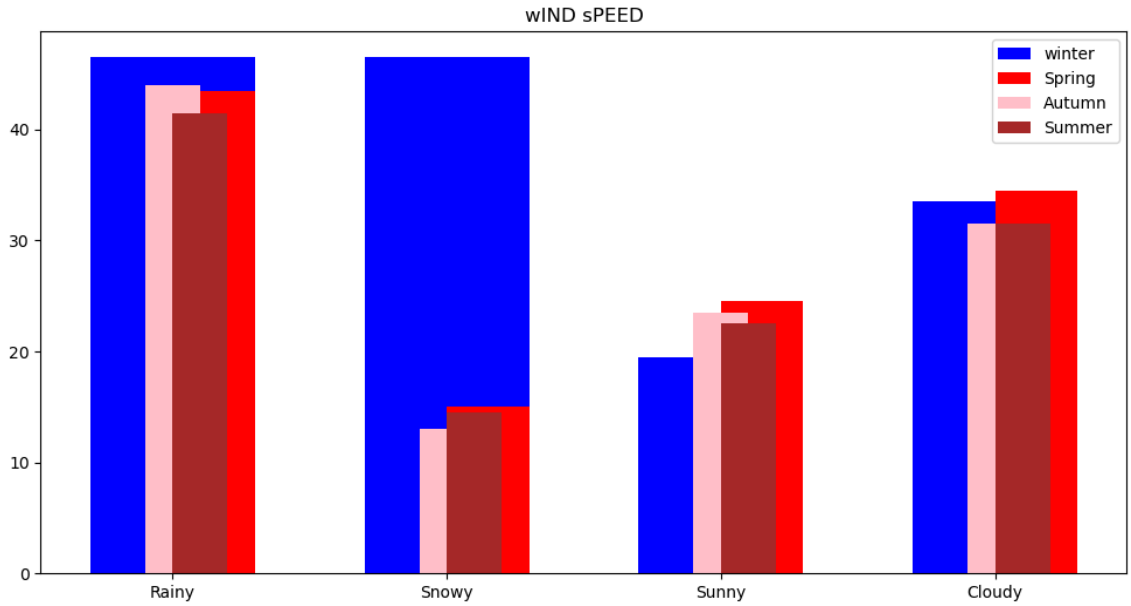
```
for i in F.columns:
    if(F[i].dtype=="int64")|(F[i].dtype=="float64"):
        print(i.upper())
        print(F[i].value_counts().head(6).reset_index().sort_values(by=

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        78     22
1        74     22
2        77     22
3        64     21
4        67     20
5        79     20
-----
WIND SPEED
   Wind Speed  count
0         7.0     36
1         6.5     32
2         9.0     30
3         2.5     30
4         9.5     28
5        14.5     27
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               19.0     20
1               18.0     20
2               11.0     19
3               17.0     18
4               16.0     17
5               14.0     16
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1014.49      4
1               1016.22      4
2               1003.65      3
3               1012.59      3
4               1019.68      3
5               1010.97      2
-----
UV INDEX
   UV Index  count
0         2    122
1         1    112
2         3    107
3         0     59
4         4     58
5        10     56
-----
VISIBILITY (KM)
   Visibility (km)  count
0              8.0     54
1              5.0     51
2              6.5     51
3              7.5     49
4              8.5     43
5              6.0     43
-----
TEMPERATURE C
   Temperature C  count
0       3.222222     40
1       8.222222     38
2       6.222222     32
3      12.222222     30
4      14.222222     29
```

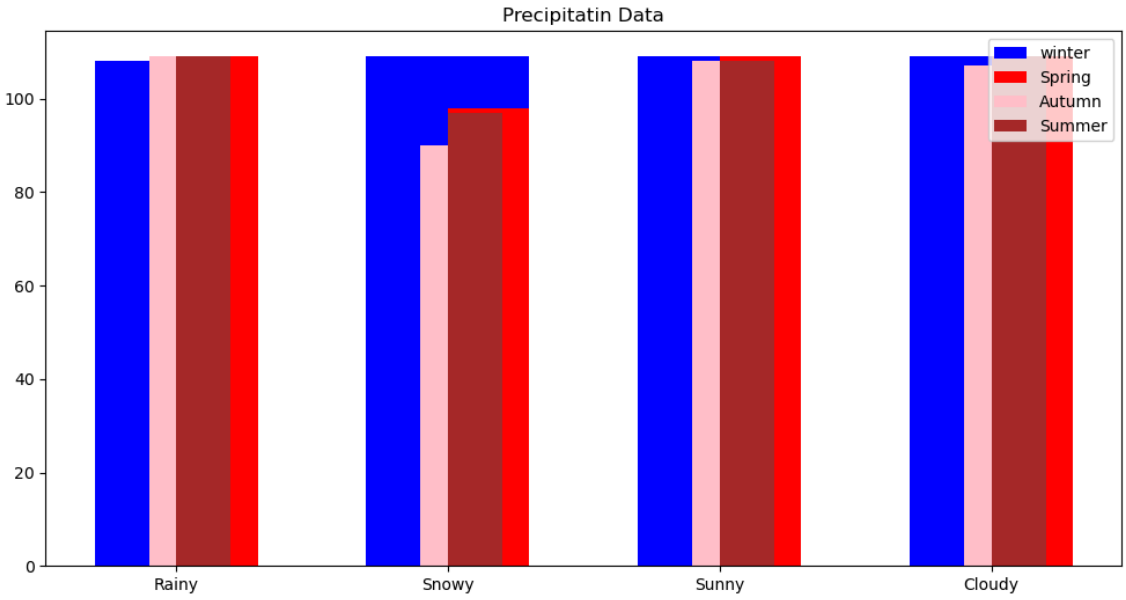
```
5       4.222222      26
-----
```

In [32]:

```
plt.figure(figsize=(12,6))
plt.bar(B1['Weather Type'],B1['Wind Speed'],color="blue",width=0.6,alig
plt.bar(D['Weather Type'],D['Wind Speed'],color="red",width=0.3,align='
plt.bar(E['Weather Type'],E['Wind Speed'],color="pink",width=0.2,align=
plt.bar(F['Weather Type'],F['Wind Speed'],color="brown",width=0.2,align
plt.title("wIND sPEED")
plt.legend()
plt.show()

```



In [33]:
```
plt.figure(figsize=(12,6))
plt.bar(B1['Weather Type'],B1['Precipitation (%)'],color="blue",width=0
plt.bar(D['Weather Type'],D['Precipitation (%)'],color="red",width=0.3,
plt.bar(E['Weather Type'],E['Precipitation (%)'],color="pink",width=0.2
plt.bar(F['Weather Type'],F['Precipitation (%)'],color="brown",width=0.
plt.title("Precipitatin Data")
plt.legend()
plt.show()

```



In [34]:
```
B1['Weather Type'].value_counts()
```

Out[34]:
```
Weather Type
Snowy     1516
Cloudy     296
Rainy      287
Sunny      273
Name: count, dtype: int64
```

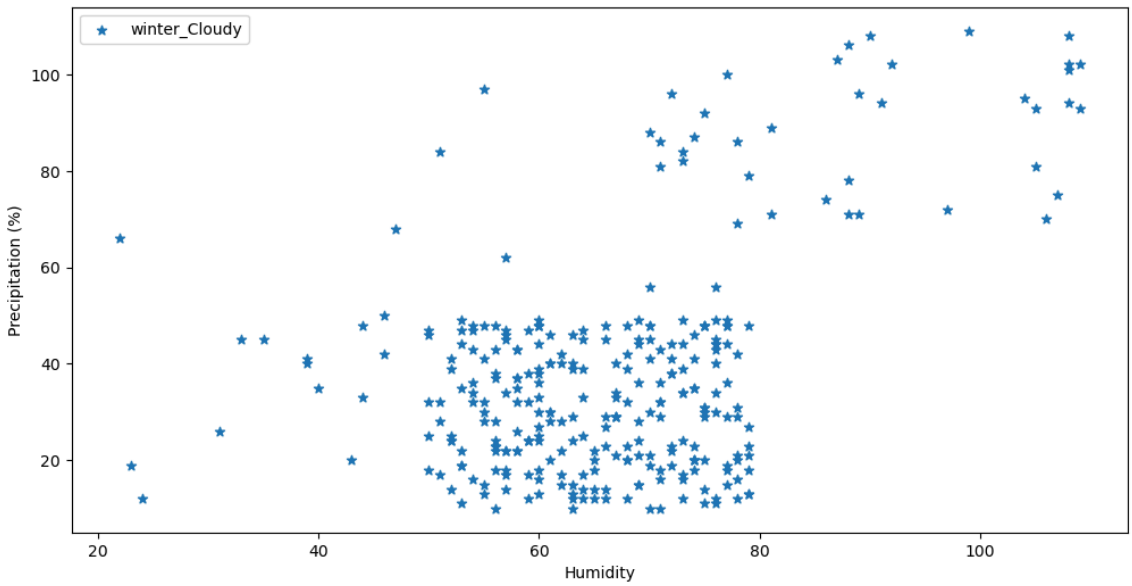
In [35]:
```
b1=B1.loc[B1['Weather Type']=='Cloudy']
b1
```

Out[35]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
90	81	28.5	71.0	partly cloudy	1008.00	4	Winter	6.0	in
99	72	17.0	96.0	partly cloudy	1013.85	13	Winter	3.5	in
172	58	1.0	43.0	overcast	1018.95	2	Winter	6.0	in
299	72	11.0	23.0	overcast	1015.73	2	Winter	5.5	in
342	22	1.0	66.0	partly cloudy	858.36	13	Winter	3.0	in
...	...	...	...	...	...	...	...	...	
12904	79	3.5	79.0	partly cloudy	1006.20	9	Winter	3.0	in
12924	69	1.0	15.0	partly cloudy	1001.50	4	Winter	6.0	in
13015	51	4.0	32.0	partly cloudy	1016.81	3	Winter	7.0	in
13075	54	13.0	32.0	partly cloudy	1002.81	3	Winter	7.0	in
13138	64	6.5	33.0	partly cloudy	1002.84	4	Winter	6.0	in

296 rows × 11 columns

In [36]:
```
plt.figure(figsize=(12,6))
plt.scatter(b1['Humidity'],b1['Precipitation (%)'],marker="*",label="wi
plt.legend()
plt.xlabel('Humidity')
plt.ylabel('Precipitation (%)')
plt.show()
```



In [37]:

```
for i in b1.columns:
    if(b1[i].dtype=="int64")|(b1[i].dtype=="float64"):
        print(i.upper())
        print(b1[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        60     16
1        76     11
2        56     11
3        73     11
4        75     11
5        77     10
-----
WIND SPEED
   Wind Speed  count
0         2.5     15
1        11.0     14
2         1.0     13
3        10.5     13
4        11.5     12
5        14.5     12
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               48.0     17
1               12.0     10
2               24.0     10
3               20.0     10
4               32.0      9
5               29.0      9
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1008.00      2
1               1019.44      2
2               1002.52      2
3               1008.59      2
4               1011.45      2
5               1003.68      2
-----
UV INDEX
   UV Index  count
0         1     75
1         4     65
2         2     54
3         3     53
4         6      7
5         5      7
-----
VISIBILITY (KM)
   Visibility (km)  count
0              8.0     41
1              5.5     35
2              6.5     34
3              8.5     31
4              7.5     29
5              6.0     28
-----
TEMPERATURE C
   Temperature C  count
0       4.222222     16
1      -0.777778     16
2      11.222222     14
3      -2.777778     13
4      12.222222     13
```

```
5       5.222222     12
-----
```

In [38]:
```
1 b1['Cloud Cover'].value_counts()
```

Out[38]:
```
Cloud Cover
partly cloudy    171
overcast         118
cloudy             7
Name: count, dtype: int64
```

In [39]:
```
1 b2=B1.loc[B1['Weather Type']=='Sunny']
2 b2
```

Out[39]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Locati
34	69	3.0	15.0	clear	1012.09	10	Winter	5.5	inla
40	37	0.5	9.0	partly cloudy	1013.99	11	Winter	8.5	inla
248	26	9.0	14.0	partly cloudy	1018.24	5	Winter	7.0	inla
450	24	7.5	18.0	clear	1027.27	5	Winter	8.5	inla
474	38	1.5	8.0	clear	1028.58	8	Winter	9.5	inla
...	...	...	...	...	...	...	...	...	
13037	67	5.5	8.0	clear	1018.44	8	Winter	6.0	inla
13082	23	7.0	5.0	clear	1025.93	10	Winter	5.0	inla
13119	67	4.5	9.0	partly cloudy	1012.05	11	Winter	8.5	inla
13140	57	7.5	9.0	clear	1024.03	7	Winter	10.0	inla
13188	34	3.5	16.0	clear	1022.64	10	Winter	9.5	inla

273 rows × 11 columns

In [40]:

```
for i in b2.columns:
    if(b2[i].dtype=="int64")|(b2[i].dtype=="float64"):
        print(i.upper())
        print(b2[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        37     11
1        48      8
2        49      7
3        42      7
4        59      7
5        30      7
-----
WIND SPEED
   Wind Speed  count
0         9.0     17
1         8.0     17
2         0.5     16
3         6.5     15
4         2.5     14
5         7.5     14
-----
PRECIPITATION (%)
   Precipitation (%)  count
0                5.0     20
1                9.0     16
2                8.0     15
3               14.0     14
4               19.0     14
5               16.0     11
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1022.31      3
1               1025.82      2
2               1012.08      2
3               1021.36      2
4               1028.58      2
5               1024.52      2
-----
UV INDEX
   UV Index  count
0         8     40
1        11     36
2         9     36
3        10     33
4         5     32
5         7     30
-----
VISIBILITY (KM)
   Visibility (km)  count
0              8.5     29
1              7.0     28
2              6.5     28
3              9.5     23
4              8.0     22
5              5.5     19
-----
TEMPERATURE C
   Temperature C  count
0      16.222222     14
1      24.222222     14
2      23.222222     13
3      11.222222     12
4       7.222222     12
```

```
5       20.222222      12
-----
```

In [41]:
```
1 b2['Cloud Cover'].value_counts()
```

Out[41]:
```
Cloud Cover
clear            170
partly cloudy     80
cloudy            12
overcast          11
Name: count, dtype: int64
```

In [42]:
```
1 b3=B1.loc[B1['Weather Type']=='Rainy']
2 b3
```

Out[42]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	L
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	
50	90	11.0	98.0	overcast	1017.49	3	Winter	4.0	
92	93	14.5	57.0	overcast	1018.44	0	Winter	4.0	
156	86	13.5	78.0	overcast	999.25	3	Winter	4.0	
160	88	15.5	88.0	overcast	997.61	1	Winter	3.5	
...	...	...	...	...	...	...	...	...	
12861	81	10.0	60.0	partly cloudy	994.20	2	Winter	5.0	
12867	47	3.5	49.0	partly cloudy	1190.68	1	Winter	6.0	
12881	87	9.0	65.0	overcast	1003.16	1	Winter	2.0	

In [43]:

```
for i in b3.columns:
    if(b3[i].dtype=="int64")|(b3[i].dtype=="float64"):
        print(i.upper())
        print(b3[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        91     15
1        97     13
2        70     13
3        95     11
4        86      9
5        98      9
-----
WIND SPEED
   Wind Speed  count
0        12.0     14
1         9.5     13
2        15.0     13
3         5.5     13
4        16.0     13
5        18.0     12
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               65.0     12
1               86.0     10
2               91.0     10
3               53.0     10
4               82.0      9
5               73.0      9
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0                996.49      2
1                991.27      2
2               1005.38      2
3               1003.98      2
4               1011.20      2
5               1014.51      2
-----
UV INDEX
   UV Index  count
0         1     78
1         3     59
2         0     53
3         2     51
4         6      7
5         8      7
-----
VISIBILITY (KM)
   Visibility (km)  count
0              4.5     38
1              2.5     37
2              4.0     35
3              3.5     28
4              2.0     27
5              1.5     26
-----
TEMPERATURE C
   Temperature C  count
0      16.222222     16
1      -6.777778     16
2      -0.777778     15
3       8.222222     14
4       5.222222     14
```

```
5       -3.777778      13
-----
```

In [44]:
```
1 b3['Cloud Cover'].value_counts()
```

Out[44]:
```
Cloud Cover
overcast         191
partly cloudy     84
cloudy            12
Name: count, dtype: int64
```

In [45]:
```
1 b4=B1.loc[B1['Weather Type']=='Snowy']
2 b4
```

Out[45]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
6	97	8.0	86.0	overcast	990.87	1	Winter	4.0	in
7	85	6.0	96.0	partly cloudy	984.46	1	Winter	3.5	in
13	87	15.0	67.0	overcast	986.19	0	Winter	1.5	in
21	88	12.5	98.0	overcast	980.31	1	Winter	3.0	in
25	79	1.0	85.0	partly cloudy	982.53	1	Winter	4.5	in
...	...	...	...	...	...	...	...	...	
13171	76	6.5	76.0	overcast	991.03	0	Winter	3.0	in
13177	94	8.0	50.0	overcast	987.63	1	Winter	4.0	in
13182	67	11.5	54.0	overcast	980.31	0	Winter	2.0	in
13193	65	15.5	50.0	overcast	982.57	1	Winter	5.0	in
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	in

1516 rows × 11 columns

In [46]:
```
1 b4['Cloud Cover'].value_counts()
```

Out[46]:
```
Cloud Cover
overcast         1214
partly cloudy     295
cloudy              7
Name: count, dtype: int64
```

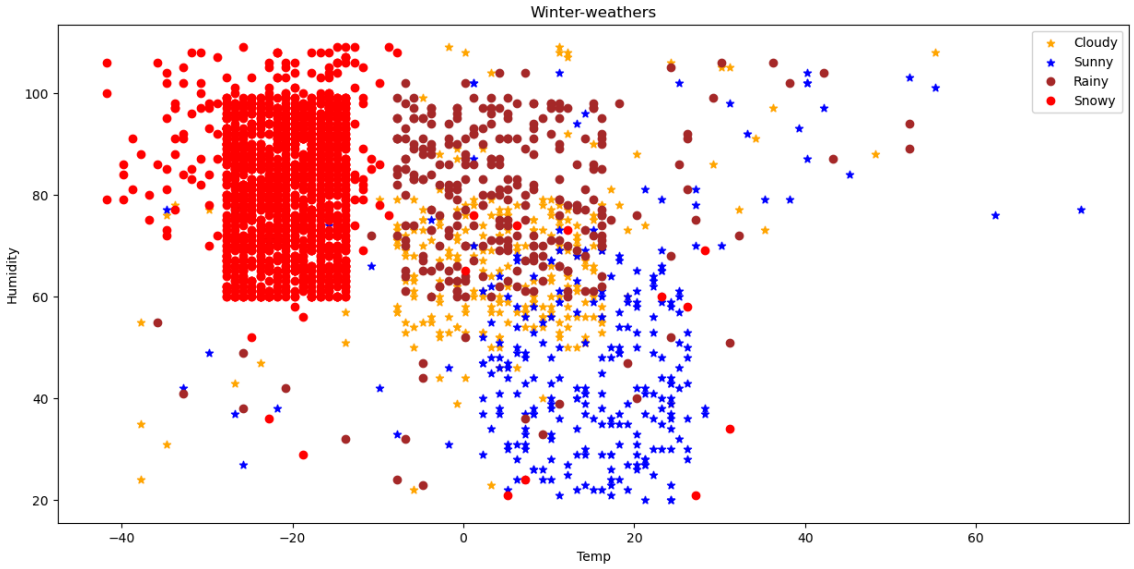
In [47]:

```
for i in b4.columns:
    if(b4[i].dtype=="int64")|(b4[i].dtype=="float64"):
        print(i.upper())
        print(b4[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        96     51
1        92     49
2        86     48
3        85     46
4        70     46
5        98     46
-----
WIND SPEED
   Wind Speed  count
0        13.5     45
1         7.0     44
2         1.0     44
3         2.5     42
4        14.0     41
5         9.0     40
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               72.0     43
1               89.0     42
2               94.0     40
3               76.0     40
4               96.0     39
5               90.0     38
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0                985.11      5
1                997.36      5
2                999.51      4
3                992.24      4
4                980.85      4
5                999.42      4
-----
UV INDEX
   UV Index  count
0         1    657
1         0    646
2        13     26
3        14     21
4         8     20
5         7     19
-----
VISIBILITY (KM)
   Visibility (km)  count
0              3.0    195
1              1.5    187
2              3.5    185
3              2.0    176
4              4.0    170
5              4.5    170
-----
TEMPERATURE C
   Temperature C  count
0     -17.777778    110
1     -15.777778    108
2     -16.777778    108
3     -20.777778    107
4     -19.777778    101
```

```
5     -25.777778     98
-----
```

In [48]:
```
plt.figure(figsize=(15,7))
plt.scatter(b1[['Temperature C']],b1[['Humidity']],c='orange',marker="*
plt.scatter(b2[['Temperature C']],b2[['Humidity']],c='blue',marker="*",
plt.scatter(b3[['Temperature C']],b3[['Humidity']],c='brown',marker="o"
plt.scatter(b4[['Temperature C']],b4[['Humidity']],c='red',marker="o",]
plt.legend()
plt.xlabel('Temp')
plt.ylabel('Humidity')
plt.title('Winter-weathers')
plt.show()
```



In [49]:
```
#Spring D
```

In [50]:
```
D['Weather Type'].value_counts()
```

Out[50]:
```
Weather Type
Sunny     291
Cloudy    283
Rainy     264
Snowy      21
Name: count, dtype: int64
```

In [51]:
```
D['Cloud Cover'].value_counts()
```

Out[51]:
```
Cloud Cover
partly cloudy    336
overcast         292
clear            194
cloudy            37
Name: count, dtype: int64
```

In [52]:
```
D1=D.loc[D['Weather Type']=='Cloudy']
D1
```

Out[52]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	in
220	61	8.5	13.0	partly cloudy	1009.60	2	Spring	6.5	in
245	59	0.5	18.0	partly cloudy	1016.89	2	Spring	7.0	in
286	76	0.0	12.0	partly cloudy	1006.38	3	Spring	5.5	in
339	61	5.0	19.0	partly cloudy	1011.70	2	Spring	6.5	in
...	...	...	...	...	...	...	...	...	
13010	53	7.0	16.0	overcast	1001.23	3	Spring	8.5	in
13117	99	17.5	83.0	partly cloudy	1007.02	7	Spring	1.0	in
13150	54	10.0	43.0	overcast	1000.58	1	Spring	5.0	in
13159	72	0.5	17.0	overcast	1006.09	1	Spring	6.5	in
13174	65	2.5	20.0	partly cloudy	1016.81	1	Spring	8.0	in

283 rows × 11 columns

In [53]:
```
D1['Cloud Cover'].value_counts()
```

Out[53]:
```
Cloud Cover
partly cloudy    171
overcast         103
cloudy             9
Name: count, dtype: int64
```

In [54]:

```
for i in D1.columns:
    if(D1[i].dtype=="int64")|(D1[i].dtype=="float64"):
        print(i.upper())
        print(D1[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        68     17
1        59     14
2        56     12
3        54     11
4        71     11
5        78     11
-----
WIND SPEED
   Wind Speed  count
0         7.0     12
1         2.0     12
2         6.0     12
3        10.5     11
4        14.0     11
5         9.5     11
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               43.0      9
1               35.0      9
2               36.0      9
3               21.0      9
4               47.0      8
5               18.0      8
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1008.57      3
1               1004.21      2
2               1009.27      2
3               1018.46      2
4               1015.78      2
5               1010.84      2
-----
UV INDEX
   UV Index  count
0         2     65
1         3     58
2         4     58
3         1     51
4         7      7
5        11      7
-----
VISIBILITY (KM)
   Visibility (km)  count
0              7.5     42
1              8.0     41
2              7.0     34
3              6.5     31
4              5.5     24
5              8.5     23
-----
TEMPERATURE C
   Temperature C  count
0      -5.777778     16
1       9.222222     15
2       1.222222     15
3      -4.777778     14
4      -6.777778     14
```

```
5       14.222222      13
-----
```

In [55]:

```
D2=D.loc[D['Weather Type']=='Sunny']
D2
```

Out[55]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
16	27	7.0	13.0	partly cloudy	1016.38	5	Spring	7.5	in
58	70	0.5	69.0	overcast	1074.07	3	Spring	12.5	in
75	47	5.0	4.0	clear	1027.53	9	Spring	10.0	in
119	29	8.0	17.0	clear	1011.53	11	Spring	10.0	in
141	48	6.0	15.0	clear	1020.00	5	Spring	9.5	in
...	...	...	...	...	...	...	...	...	
13004	48	4.5	10.0	clear	1013.20	5	Spring	9.5	in
13008	52	8.5	3.0	clear	1011.75	7	Spring	9.5	in
13089	99	12.0	105.0	clear	1028.08	12	Spring	6.0	in
13152	59	1.5	14.0	clear	1020.38	8	Spring	5.5	in
13168	45	7.0	28.0	partly cloudy	894.65	5	Spring	11.5	in

291 rows × 11 columns

In [56]:

```
for i in D2.columns:
    if(D2[i].dtype=="int64")|(D2[i].dtype=="float64"):
        print(i.upper())
        print(D2[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        35     11
1        40      9
2        45      8
3        43      8
4        62      7
5        61      7
-----
WIND SPEED
   Wind Speed  count
0         6.5     20
1         8.5     20
2         1.0     19
3         0.5     17
4         7.5     17
5         8.0     16
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               13.0     23
1                6.0     18
2               10.0     15
3               16.0     14
4                9.0     13
5                3.0     13
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1021.16      2
1               1013.20      2
2               1029.29      2
3               1029.28      2
4               1025.55      2
5               1015.62      2
-----
UV INDEX
   UV Index  count
0         5     51
1         7     44
2         9     35
3        11     34
4        10     33
5         6     32
-----
VISIBILITY (KM)
   Visibility (km)  count
0              5.5     38
1              9.5     33
2              9.0     28
3              6.0     27
4              8.0     22
5              8.5     21
-----
TEMPERATURE C
   Temperature C  count
0      15.222222     16
1      25.222222     15
2      23.222222     13
3       4.222222     13
4      26.222222     13
```

```
5       2.222222     12
-----
```

In [57]:
```
1 D2['Cloud Cover'].value_counts()
```

Out[57]:
```
Cloud Cover
clear            194
partly cloudy     78
cloudy            12
overcast           7
Name: count, dtype: int64
```

In [58]:
```
1 D3=D.loc[D['Weather Type']=='Rainy']
2 D3
```

Out[58]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
73	62	6.5	84.0	overcast	1013.61	0	Spring	4.5	in
127	76	9.0	68.0	overcast	1004.27	3	Spring	3.5	in
148	72	2.5	26.0	cloudy	973.08	6	Spring	8.5	in
161	77	12.0	51.0	partly cloudy	1007.88	0	Spring	3.5	in
177	88	19.5	73.0	overcast	1018.01	1	Spring	4.0	in
...	...	...	...	...	...	...	...	...	
13058	87	9.5	67.0	overcast	1015.28	0	Spring	3.0	in
13092	89	13.0	68.0	partly cloudy	991.42	3	Spring	3.0	in
13114	60	10.0	79.0	overcast	929.98	13	Spring	9.0	in
13133	96	15.0	93.0	partly cloudy	996.28	0	Spring	1.5	in
13148	79	20.0	67.0	partly cloudy	994.00	2	Spring	3.0	in

264 rows × 11 columns

In [59]:
```
1 D3['Cloud Cover'].value_counts()
```

Out[59]:
```
Cloud Cover
overcast         175
partly cloudy     80
cloudy             9
Name: count, dtype: int64
```

In [60]:

```
for i in D3.columns:
    if(D3[i].dtype=="int64")|(D3[i].dtype=="float64"):
        print(i.upper())
        print(D3[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        96     12
1        87     10
2        62      9
3        71      9
4        78      9
5        81      8
-----
WIND SPEED
   Wind Speed  count
0        13.0     14
1         6.5     13
2         9.0     13
3        11.5     12
4        10.0     11
5         9.5      9
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               72.0     11
1               95.0     10
2               52.0      9
3               91.0      9
4               70.0      8
5               90.0      8
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1003.36      3
1               1011.38      2
2                997.16      2
3               1000.57      2
4               1000.69      2
5               1004.05      2
-----
UV INDEX
   UV Index  count
0         3     66
1         1     65
2         2     49
3         0     42
4        13      7
5         6      5
-----
VISIBILITY (KM)
   Visibility (km)  count
0              1.5     36
1              3.5     34
2              3.0     34
3              4.0     29
4              2.0     27
5              2.5     25
-----
TEMPERATURE C
   Temperature C  count
0      -1.777778     15
1      13.222222     13
2       6.222222     13
3      11.222222     12
4      12.222222     12
```

```
5       8.222222     12
-----
```

In [61]:
```
D4=D.loc[D['Weather Type']=='Snowy']
D4
```

Out[61]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
584	76	2.5	64.0	partly cloudy	1000.66	6	Spring	0.0	in
1410	30	5.5	92.0	cloudy	1186.34	10	Spring	7.5	in
1510	33	5.0	14.0	partly cloudy	1135.31	9	Spring	16.0	in
1797	32	7.5	28.0	overcast	840.97	5	Spring	9.5	in
2102	62	7.0	82.0	cloudy	1173.16	12	Spring	0.5	in
4155	67	6.0	87.0	overcast	1044.79	4	Spring	10.5	in
5855	28	10.0	98.0	overcast	896.93	2	Spring	11.0	in
5924	78	1.0	40.0	overcast	892.61	0	Spring	15.5	in
6043	37	5.5	29.0	overcast	965.32	3	Spring	14.5	in
6451	23	10.5	82.0	overcast	1172.49	11	Spring	9.5	in
7217	69	4.5	31.0	partly cloudy	1041.60	0	Spring	11.0	in
7334	24	13.0	10.0	cloudy	814.44	5	Spring	1.0	in
7434	21	11.0	25.0	partly cloudy	1165.61	1	Spring	5.0	in
7842	39	14.0	39.0	overcast	858.41	10	Spring	8.5	in
8103	28	14.5	15.0	cloudy	938.78	7	Spring	2.0	in
10014	21	9.0	31.0	cloudy	805.36	9	Spring	16.0	in
10660	74	14.0	42.0	partly cloudy	1166.80	9	Spring	17.5	in
11007	24	10.5	74.0	partly cloudy	1115.30	8	Spring	9.5	in
11028	51	7.0	11.0	partly cloudy	1181.54	10	Spring	14.5	in
11330	42	15.0	14.0	cloudy	1121.53	0	Spring	13.5	in
12196	64	7.5	23.0	cloudy	1028.04	10	Spring	9.5	in

In [62]:
```
D4['Cloud Cover'].value_counts()
```

Out[62]:
```
Cloud Cover
partly cloudy    7
cloudy           7
overcast         7
Name: count, dtype: int64
```

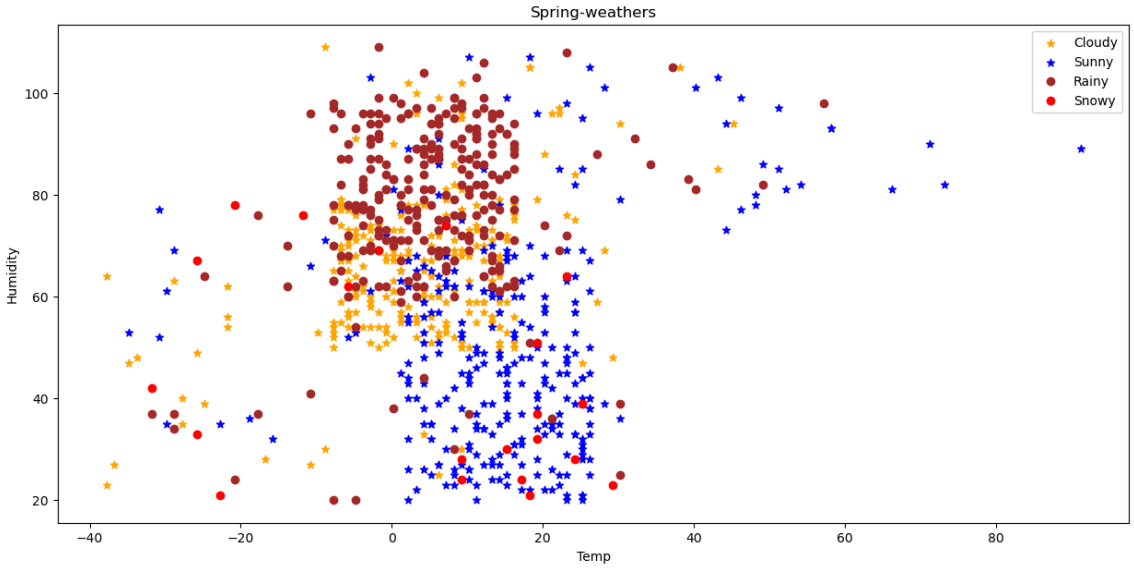
In [63]:

```
for i in D4.columns:
    if(D4[i].dtype=="int64")|(D4[i].dtype=="float64"):
        print(i.upper())
        print(D4[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        28      2
1        21      2
2        24      2
3        76      1
4        69      1
5        42      1
-----
WIND SPEED
   Wind Speed  count
0         5.5      2
1         7.5      2
2         7.0      2
3        10.5      2
4        14.0      2
5         2.5      1
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               31.0      2
1               14.0      2
2               82.0      2
3               10.0      1
4               11.0      1
5               74.0      1
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1000.66      1
1                814.44      1
2               1121.53      1
3               1181.54      1
4               1115.30      1
5               1166.80      1
-----
UV INDEX
   UV Index  count
0        10      4
1         9      3
2         0      3
3         5      2
4         6      1
5        12      1
-----
VISIBILITY (KM)
   Visibility (km)  count
0              9.5      4
1             16.0      2
2             11.0      2
3             14.5      2
4              0.0      1
5              7.5      1
-----
TEMPERATURE C
   Temperature C  count
0      19.222222      3
1     -25.777778      2
2       9.222222      2
3     -11.777778      1
4      25.222222      1
```

```
5     -31.777778      1
-----
```

In [64]:
```
plt.figure(figsize=(15,7))
plt.scatter(D1[['Temperature C']],D1[['Humidity']],c='orange',marker="*
plt.scatter(D2[['Temperature C']],D2[['Humidity']],c='blue',marker="*",
plt.scatter(D3[['Temperature C']],D3[['Humidity']],c='brown',marker="o"
plt.scatter(D4[['Temperature C']],D4[['Humidity']],c='red',marker="o",
plt.legend()
plt.xlabel('Temp')
plt.ylabel('Humidity')
plt.title('Spring-weathers')
plt.show()#temp and Humidity graph same as winter
```



In [65]:
```
#Autumn E
```

In [66]:
```
E['Cloud Cover'].value_counts()
```

Out[66]:
```
Cloud Cover
partly cloudy    316
overcast         280
clear            169
cloudy            29
Name: count, dtype: int64
```

In [67]:
```
E['Weather Type'].value_counts()
```

Out[67]:
```
Weather Type
Cloudy    282
Sunny     256
Rainy     240
Snowy      16
Name: count, dtype: int64
```

In [68]:
```
E1=E.loc[E['Weather Type']=="Cloudy"]
E1
```

Out[68]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
62	55	1.5	32.0	partly cloudy	1009.01	2	Autumn	5.5	in
139	50	6.5	21.0	partly cloudy	1013.93	4	Autumn	6.0	in
142	50	5.5	13.0	overcast	1002.04	4	Autumn	8.0	in
153	62	14.0	13.0	overcast	1003.74	1	Autumn	9.0	in
154	75	9.5	40.0	overcast	1005.39	2	Autumn	8.0	in
...	...	...	...	...	...	...	...	...	
13063	74	5.0	20.0	partly cloudy	1015.39	2	Autumn	8.0	in
13096	59	10.0	26.0	overcast	1007.97	2	Autumn	7.0	in
13097	90	7.5	73.0	overcast	1019.65	8	Autumn	4.0	in
13102	89	8.5	80.0	partly cloudy	1010.68	6	Autumn	5.5	in
13129	51	0.5	14.0	partly cloudy	1017.63	4	Autumn	6.5	in

282 rows × 11 columns

In [69]:

```
for i in E1.columns:
    if(E1[i].dtype=="int64")|(E1[i].dtype=="float64"):
        print(i.upper())
        print(E1[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        51     14
1        62     13
2        74     12
3        71     10
4        75     10
5        70     10
-----
WIND SPEED
   Wind Speed  count
0         3.5     13
1         9.5     13
2        13.0     12
3        10.5     12
4        11.5     11
5         5.5     11
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               48.0     10
1               14.0     10
2               26.0     10
3               42.0      9
4               29.0      9
5               15.0      8
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1006.42      3
1               1013.74      2
2               1002.20      2
3               1007.43      2
4               1008.14      2
5               1010.53      2
-----
UV INDEX
   UV Index  count
0         2     68
1         3     63
2         4     51
3         1     51
4         6      8
5         5      7
-----
VISIBILITY (KM)
   Visibility (km)  count
0              7.0     39
1              5.5     34
2              6.5     33
3              8.0     32
4              8.5     30
5              6.0     26
-----
TEMPERATURE C
   Temperature C  count
0      -7.777778     20
1      14.222222     14
2      10.222222     14
3      -2.777778     13
4      12.222222     13
```

```
5       4.222222     12
-----
```

In [70]:
```
1 E1['Cloud Cover'].value_counts()
```

Out[70]:
```
Cloud Cover
partly cloudy    176
overcast          99
cloudy             7
Name: count, dtype: int64
```

In [71]:
```
1 E2=E.loc[E['Weather Type']=="Sunny"]
2 E2
```

Out[71]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Locati
11	43	2.0	16.0	clear	1029.16	11	Autumn	7.5	inla
134	77	0.5	91.0	cloudy	934.49	14	Autumn	17.0	inla
184	54	0.0	3.0	clear	1024.65	8	Autumn	9.5	inla
243	69	8.0	5.0	clear	1026.28	9	Autumn	9.5	inla
244	47	11.5	24.0	partly cloudy	1105.69	9	Autumn	1.5	inla
...	...	...	...	...	...	...	...	...	
12996	30	5.0	15.0	clear	1024.28	9	Autumn	6.0	inla
13040	50	1.5	10.0	clear	1015.95	5	Autumn	9.0	inla
13044	61	5.0	19.0	clear	1027.48	7	Autumn	8.0	inla
13045	43	0.5	3.0	partly cloudy	1018.47	10	Autumn	8.0	inla
13160	27	5.0	6.0	clear	1011.40	9	Autumn	8.0	inla

256 rows × 11 columns

In [72]:

```
for i in E2.columns:
    if(E2[i].dtype=="int64")|(E2[i].dtype=="float64"):
        print(i.upper())
        print(E2[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        30      9
1        48      9
2        41      8
3        24      8
4        39      7
5        23      7
-----
WIND SPEED
   Wind Speed  count
0         1.5     21
1         3.5     19
2         5.0     18
3         8.5     16
4         2.0     14
5         6.5     14
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               10.0     15
1                3.0     14
2               16.0     13
3               19.0     12
4                7.0     12
5                9.0     11
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1012.12      2
1               1022.43      2
2               1021.88      2
3               1015.70      2
4               1028.87      2
5               1025.91      2
-----
UV INDEX
   UV Index  count
0        11     43
1         9     41
2         8     38
3        10     34
4         5     25
5         7     24
-----
VISIBILITY (KM)
   Visibility (km)  count
0              6.0     28
1              8.0     27
2              9.5     25
3              9.0     25
4              7.5     20
5              6.5     19
-----
TEMPERATURE C
   Temperature C  count
0       7.222222     13
1       5.222222     12
2       9.222222     11
3      18.222222     11
4      17.222222     10
```

```
5      10.222222     10
-----
```

In [73]:
```
1 E2['Cloud Cover'].value_counts()
```

Out[73]:
```
Cloud Cover
clear            169
partly cloudy     69
cloudy             9
overcast           9
Name: count, dtype: int64
```

In [74]:
```
1 E3=E.loc[E['Weather Type']=="Rainy"]
2 E3
```

Out[74]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
208	99	34.0	74.0	partly cloudy	1016.03	9	Autumn	1.0	in
222	78	15.0	60.0	overcast	991.36	2	Autumn	4.0	in
234	91	18.5	97.0	overcast	1011.14	0	Autumn	4.0	in
263	89	17.5	75.0	overcast	997.65	2	Autumn	2.0	in
354	81	17.0	51.0	overcast	995.93	0	Autumn	1.0	in
...	...	...	...	...	...	...	...	...	
13042	69	7.5	56.0	overcast	1003.59	2	Autumn	1.0	in
13053	60	12.0	83.0	overcast	1005.30	3	Autumn	3.5	in
13076	75	26.0	80.0	partly cloudy	1010.88	8	Autumn	5.5	in
13124	101	16.0	80.0	partly cloudy	1015.06	11	Autumn	1.5	in
13173	66	8.0	59.0	overcast	1006.99	2	Autumn	5.0	in

240 rows × 11 columns

In [75]:

```
for i in E3.columns:
    if(E3[i].dtype=="int64")|(E3[i].dtype=="float64"):
        print(i.upper())
        print(E3[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        81     11
1        63     11
2        65      9
3        95      9
4        94      8
5        77      7
-----
WIND SPEED
   Wind Speed  count
0        16.0     11
1         9.5     10
2         7.5     10
3        15.0     10
4        11.0      9
5        14.5      9
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               88.0      8
1               63.0      8
2               84.0      8
3               59.0      7
4               82.0      7
5               54.0      6
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1019.47      2
1                992.26      2
2                998.45      2
3               1006.14      2
4                990.87      2
5               1011.51      2
-----
UV INDEX
   UV Index  count
0         2     58
1         3     49
2         0     47
3         1     45
4        12      6
5        10      6
-----
VISIBILITY (KM)
   Visibility (km)  count
0              2.5     33
1              4.0     31
2              1.5     28
3              3.0     26
4              2.0     24
5              3.5     22
-----
TEMPERATURE C
   Temperature C  count
0      -5.777778     14
1       8.222222     14
2       5.222222     12
3      -2.777778     12
4      12.222222     11
```

```
5       16.222222      10
-----
```

In [76]:
```
1 E3['Cloud Cover'].value_counts()
```

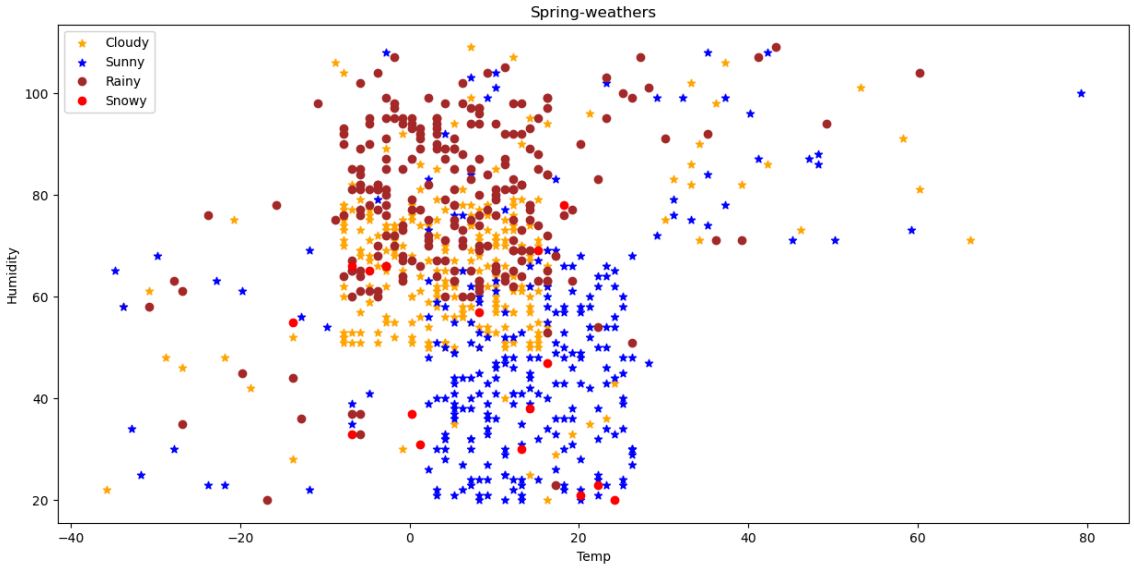
Out[76]:
```
Cloud Cover
overcast         168
partly cloudy     65
cloudy             7
Name: count, dtype: int64
```

In [77]:
```
1 E4=E.loc[E['Weather Type']=="Snowy"]
2 E4
```

Out[77]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
94	38	11.0	15.0	partly cloudy	824.71	9	Autumn	12.5	in
475	65	6.5	86.0	cloudy	899.08	6	Autumn	19.0	in
511	69	5.0	81.0	partly cloudy	994.68	12	Autumn	16.5	in
2099	20	5.5	40.0	overcast	1058.82	4	Autumn	15.0	in
4436	57	6.0	90.0	cloudy	975.43	4	Autumn	15.0	in
4703	78	13.0	77.0	cloudy	982.29	5	Autumn	12.5	in
6739	55	3.5	49.0	partly cloudy	992.07	7	Autumn	13.5	in
6871	33	3.0	90.0	overcast	1065.17	11	Autumn	3.0	in
7687	21	11.0	52.0	partly cloudy	931.23	8	Autumn	15.5	in
7703	31	9.5	82.0	cloudy	985.10	3	Autumn	6.5	in
7807	30	2.5	82.0	overcast	1095.73	10	Autumn	13.0	in
9980	47	2.0	10.0	overcast	1152.88	2	Autumn	15.5	in
9989	66	12.5	28.0	partly cloudy	1146.89	1	Autumn	2.5	in
11265	66	5.0	85.0	cloudy	829.98	0	Autumn	4.5	in
11912	23	4.5	51.0	cloudy	1023.37	12	Autumn	17.0	in
12601	37	3.0	11.0	partly cloudy	888.19	11	Autumn	1.0	in

In [78]:
```
plt.figure(figsize=(15,7))
plt.scatter(E1[['Temperature C']],E1[['Humidity']],c='orange',marker="*
plt.scatter(E2[['Temperature C']],E2[['Humidity']],c='blue',marker="*",
plt.scatter(E3[['Temperature C']],E3[['Humidity']],c='brown',marker="o"
plt.scatter(E4[['Temperature C']],E4[['Humidity']],c='red',marker="o",l
plt.legend()
plt.xlabel('Temp')
plt.ylabel('Humidity')
plt.title('Spring-weathers')
plt.show()# Autumn has same pattern as winter and spring for Temp & Hum
```



In [79]:
```
# Summer F
```

In [80]:
```
F['Weather Type'].value_counts()
```

Out[80]:
```
Weather Type
Rainy     278
Cloudy    246
Sunny     245
Snowy      22
Name: count, dtype: int64
```

In [81]:
```
F['Cloud Cover'].value_counts()
```

Out[81]:
```
Cloud Cover
partly cloudy    334
overcast         285
clear            153
cloudy            19
Name: count, dtype: int64
```

In [82]:
```
F1=F.loc[F['Weather Type']=="Cloudy"]
F1
```

Out[82]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
5	55	3.5	26.0	overcast	1010.03	2	Summer	5.0	ir
23	73	13.5	29.0	overcast	1012.02	1	Summer	9.0	ir
51	55	8.5	37.0	partly cloudy	1014.49	4	Summer	7.5	ir
121	65	1.5	18.0	overcast	1008.47	2	Summer	9.0	ir
155	105	7.5	75.0	overcast	1006.45	10	Summer	4.0	ir
...	...	...	...	...	...	...	...	...	
12923	68	9.0	48.0	partly cloudy	1001.12	2	Summer	6.5	ir
12984	88	16.5	71.0	partly cloudy	1014.15	4	Summer	7.5	ir
13039	79	2.5	17.0	overcast	1002.40	4	Summer	9.0	ir
13083	71	3.0	43.0	overcast	920.13	10	Summer	15.0	ir
13136	74	0.0	103.0	partly cloudy	1012.40	10	Summer	5.5	ir

246 rows × 11 columns

In [83]:
```
F1['Cloud Cover'].value_counts()
```

Out[83]:
```
Cloud Cover
partly cloudy    160
overcast          82
cloudy             4
Name: count, dtype: int64
```

In [84]:

```
for i in F1.columns:
    if(F1[i].dtype=="int64")|(F1[i].dtype=="float64"):
        print(i.upper())
        print(F1[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        77     12
1        79     10
2        61      9
3        71      9
4        76      9
5        67      9
-----
WIND SPEED
   Wind Speed  count
0         2.5     13
1        10.5     12
2         3.5     11
3        14.5     11
4         9.5     10
5        11.0     10
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               19.0      9
1               21.0      8
2               30.0      8
3               17.0      8
4               27.0      8
5               31.0      7
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1016.22      3
1               1003.00      2
2               1013.03      2
3               1013.21      2
4               1010.16      2
5               1003.12      2
-----
UV INDEX
   UV Index  count
0         2     62
1         4     49
2         1     46
3         3     43
4        14      8
5        10      7
-----
VISIBILITY (KM)
   Visibility (km)  count
0              8.0     30
1              6.0     25
2              7.0     24
3              7.5     24
4              8.5     24
5              6.5     24
-----
TEMPERATURE C
   Temperature C  count
0       8.222222     17
1       0.222222     14
2       3.222222     14
3      14.222222     12
4      -1.777778     11
```

```
5       -0.777778      11
-----
```

In [85]:
```
F2=F.loc[F['Weather Type']=="Sunny"]
F2
```

Out[85]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Locati
19	102	12.0	72.0	clear	1012.25	4	Summer	8.0	inla
106	47	8.5	2.0	clear	1027.33	11	Summer	5.5	inla
247	108	7.5	83.0	partly cloudy	1018.84	1	Summer	4.0	inla
257	24	7.0	1.0	clear	1020.87	9	Summer	7.0	inla
269	57	3.5	0.0	partly cloudy	1016.85	10	Summer	8.5	inla
...	...	...	...	...	...	...	...	...	
13079	44	2.5	18.0	clear	1019.85	5	Summer	7.0	inla
13108	81	10.5	96.0	clear	1022.66	11	Summer	3.5	inla
13189	49	7.5	11.0	clear	1022.86	7	Summer	9.5	inla
13191	48	6.5	14.0	clear	1029.37	8	Summer	8.0	inla
13192	24	8.0	5.0	clear	1029.61	8	Summer	9.0	inla

245 rows × 11 columns

In [86]:
```
F2['Cloud Cover'].value_counts()
```

Out[86]:
```
Cloud Cover
clear            153
partly cloudy     75
overcast          13
cloudy             4
Name: count, dtype: int64
```

In [87]:

```
for i in F2.columns:
    if(F2[i].dtype=="int64")|(F2[i].dtype=="float64"):
        print(i.upper())
        print(F2[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        49     10
1        32      9
2        24      8
3        64      8
4        50      6
5        58      6
-----
WIND SPEED
   Wind Speed  count
0         6.5     17
1         7.0     16
2         2.5     15
3         0.5     14
4         7.5     12
5         3.5     12
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               18.0     15
1               11.0     14
2               16.0     13
3               19.0     11
4               13.0     11
5                0.0     10
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1012.25      2
1               1012.59      2
2               1016.92      2
3               1028.41      2
4               1012.67      2
5               1019.78      2
-----
UV INDEX
   UV Index  count
0        10     38
1         9     37
2         5     31
3        11     29
4         6     29
5         7     27
-----
VISIBILITY (KM)
   Visibility (km)  count
0              6.5     26
1              7.5     25
2              8.0     24
3              9.0     23
4              5.0     19
5              9.5     18
-----
TEMPERATURE C
   Temperature C  count
0      17.222222     13
1       3.222222     13
2       6.222222     12
3      24.222222     11
4      18.222222     11
```

```
5       15.222222      11
-----
```

In [88]:
```
F3=F.loc[F['Weather Type']=="Rainy"]
F3
```

Out[88]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
78	62	10.0	83.0	partly cloudy	1009.48	1	Summer	3.5	ir
109	75	0.5	79.0	overcast	1057.69	0	Summer	10.0	ir
118	61	5.5	53.0	overcast	995.09	2	Summer	4.0	ir
157	60	11.5	50.0	partly cloudy	995.67	3	Summer	3.5	ir
284	87	18.0	58.0	overcast	1001.57	1	Summer	1.5	ir
...	...	...	...	...	...	...	...	...	
12950	60	9.0	74.0	partly cloudy	1006.74	2	Summer	1.5	ir
12986	82	15.5	90.0	overcast	1007.60	3	Summer	5.0	ir
13028	87	14.0	89.0	overcast	1015.70	0	Summer	5.0	ir
13066	102	31.0	108.0	partly cloudy	1005.89	10	Summer	2.0	ir
13086	87	6.0	97.0	partly cloudy	1005.50	3	Summer	4.0	ir

278 rows × 11 columns

In [89]:
```
F3['Cloud Cover'].value_counts()
```

Out[89]:
```
Cloud Cover
overcast         184
partly cloudy     89
cloudy             5
Name: count, dtype: int64
```

In [90]:

```
for i in F3.columns:
    if(F3[i].dtype=="int64")|(F3[i].dtype=="float64"):
        print(i.upper())
        print(F3[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
HUMIDITY
   Humidity  count
0        78     12
1        74     12
2        70     12
3        75     11
4        87     10
5        85      9
-----
WIND SPEED
   Wind Speed  count
0        11.5     13
1        14.5     12
2        13.5     12
3        10.0     11
4        12.5     11
5         9.0     11
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               50.0      9
1               85.0      8
2               62.0      7
3               70.0      7
4               58.0      7
5               75.0      7
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1003.65      3
1               1001.87      2
2                995.09      2
3               1019.68      2
4                996.03      2
5                998.55      2
-----
UV INDEX
   UV Index  count
0         1     59
1         3     59
2         2     57
3         0     53
4        10     11
5         9      8
-----
VISIBILITY (KM)
   Visibility (km)  count
0              3.5     38
1              2.5     33
2              1.5     31
3              2.0     30
4              3.0     29
5              4.0     28
-----
TEMPERATURE C
   Temperature C  count
0      -7.777778     14
1      11.222222     14
2       3.222222     13
3       8.222222     13
4       1.222222     12
```

```
5       -6.777778      12
-----
```

In [91]:

```
F4=F.loc[F['Weather Type']=="Snowy"]
F4
```

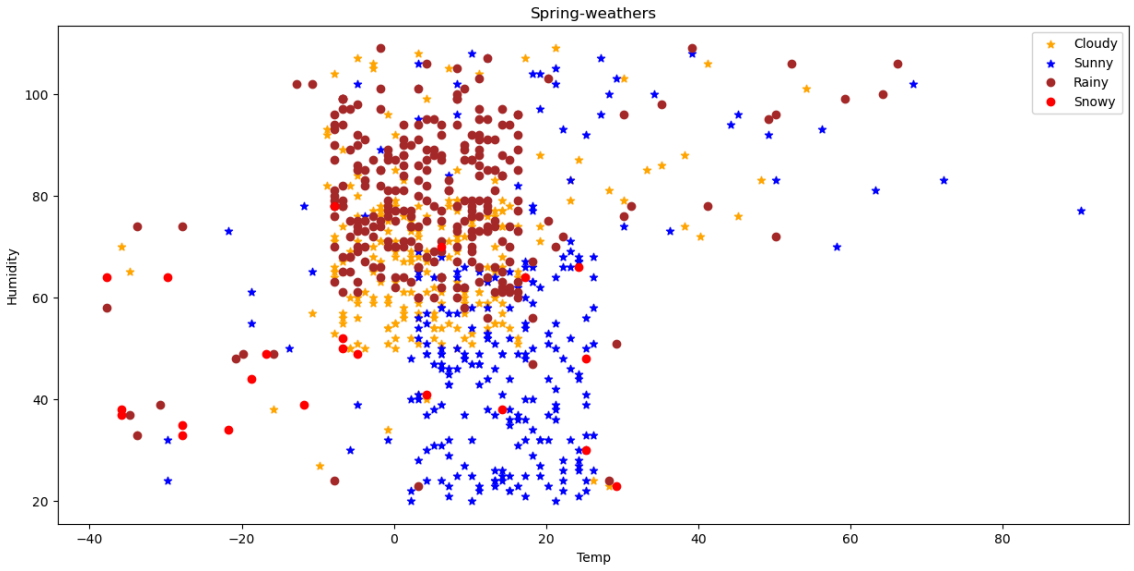
Out[91]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
72	52	6.0	97.0	overcast	1063.39	7	Summer	12.5	ir
949	30	4.0	97.0	partly cloudy	995.64	11	Summer	19.0	ir
1121	64	11.5	78.0	partly cloudy	1007.36	14	Summer	16.5	ir
3247	34	11.0	42.0	overcast	1150.77	8	Summer	15.0	ir
3653	48	5.0	26.0	partly cloudy	936.13	7	Summer	9.0	ir
6944	49	2.5	49.0	cloudy	1058.18	12	Summer	18.5	ir
7394	50	10.0	93.0	partly cloudy	985.63	5	Summer	11.5	ir
7646	33	14.5	18.0	partly cloudy	1008.33	1	Summer	18.0	ir
7752	64	12.5	18.0	partly cloudy	1174.48	4	Summer	6.5	ir
8190	35	10.5	48.0	cloudy	1187.94	12	Summer	4.0	ir
8332	41	9.0	91.0	partly cloudy	993.39	13	Summer	19.5	ir
9009	39	3.5	83.0	cloudy	1015.20	6	Summer	8.5	ir
9693	49	13.0	12.0	cloudy	1157.75	7	Summer	20.0	ir
10162	23	1.5	89.0	partly cloudy	895.35	9	Summer	17.5	ir
10222	44	10.5	91.0	overcast	1090.45	1	Summer	20.0	ir
11459	37	13.0	56.0	overcast	966.84	13	Summer	11.0	ir
11544	38	8.0	93.0	partly cloudy	803.29	12	Summer	14.5	ir
11594	64	14.5	39.0	overcast	978.38	9	Summer	5.0	ir
11657	38	7.0	10.0	cloudy	983.75	11	Summer	2.5	ir
11684	70	10.5	15.0	partly cloudy	1137.19	12	Summer	0.0	ir
12226	78	7.5	94.0	cloudy	997.80	5	Summer	8.5	ir
12869	66	12.0	44.0	overcast	939.18	14	Summer	12.0	ir

In [92]:
```
1 F4['Cloud Cover'].value_counts()
```

Out[92]:
```
Cloud Cover
partly cloudy    10
overcast          6
cloudy            6
Name: count, dtype: int64
```

In [93]:
```
1  plt.figure(figsize=(15,7))
2  plt.scatter(F1[['Temperature C']],F1[['Humidity']],c='orange',marker="*
3  plt.scatter(F2[['Temperature C']],F2[['Humidity']],c='blue',marker="*",
4  plt.scatter(F3[['Temperature C']],F3[['Humidity']],c='brown',marker="o"
5  plt.scatter(F4[['Temperature C']],F4[['Humidity']],c='red',marker="o",l
6  plt.legend()
7  plt.xlabel('Temp')
8  plt.ylabel('Humidity')
9  plt.title('Spring-weathers')
10 plt.show()
```



#for Mountain area

In [7]:
```
B=X.loc[X['Location']=='mountain']
B
```

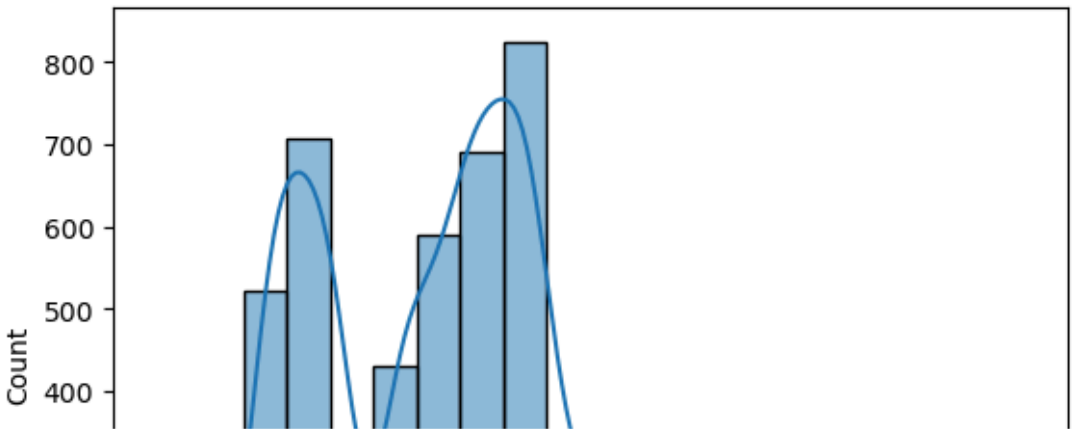
Out[7]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season
2	30.0	64	7.0	16.0	clear	1018.72	5	Spring
4	27.0	74	17.0	66.0	overcast	990.67	1	Winter
8	3.0	83	6.0	66.0	overcast	999.44	0	Winter
10	35.0	45	6.0	86.0	partly cloudy	879.88	2	Spring
12	12.0	59	10.5	25.0	partly cloudy	1016.08	3	Autumn
...	...	...	...	...	...	...	...	...
13184	3.0	62	7.5	14.0	overcast	1128.35	3	Summer
13187	4.0	71	18.5	88.0	overcast	992.81	1	Winter
13190	30.0	24	3.5	16.0	partly cloudy	1017.54	11	Summer
13195	10.0	74	14.5	71.0	overcast	1003.15	1	Summer
13199	-5.0	38	0.0	92.0	overcast	1015.37	5	Autumn

4813 rows × 12 columns

In [8]:
```
# FOR MOUNTAIN AREA
import seaborn as sns
for i in B.columns:
    if(B[i].dtype=="int64")|(B[i].dtype=="float64"):
        print(i.upper())
        sns.histplot(B[i],bins=20,kde=True)
        plt.show()
```

```
TEMPERATURE

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [9]:
```
1 B['Season'].value_counts()
```

Out[9]:
```
Season
Winter    2308
Spring     851
Summer     848
Autumn     806
Name: count, dtype: int64
```

In [10]:
```
1 G=B.loc[B['Season']=='Winter']
2 G
```

Out[10]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
4	27.0	74	17.0	66.0	overcast	990.67	1	Winter	
8	3.0	83	6.0	66.0	overcast	999.44	0	Winter	
20	-10.0	67	8.5	75.0	overcast	991.53	1	Winter	
29	30.0	70	16.0	54.0	partly cloudy	1007.75	0	Winter	
33	15.0	97	23.0	91.0	overcast	1009.36	13	Winter	
...	...	...	...	...	...	...	...	...	
13166	26.0	57	2.5	34.0	overcast	1014.82	3	Winter	
13175	-6.0	87	9.0	97.0	partly cloudy	995.42	1	Winter	
13179	-4.0	98	1.5	76.0	overcast	988.72	0	Winter	
13181	-7.0	97	0.5	99.0	overcast	983.91	0	Winter	
13187	4.0	71	18.5	88.0	overcast	992.81	1	Winter	

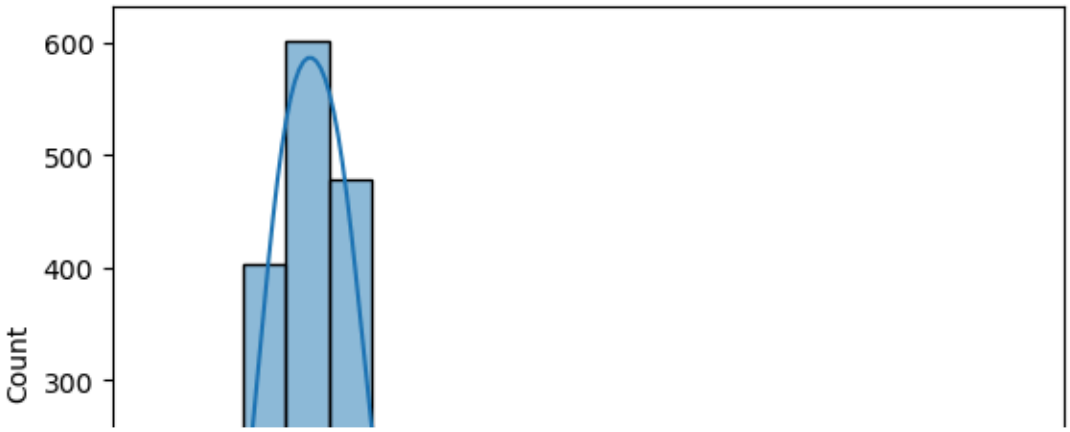
2308 rows × 12 columns

In [11]:
```
1 G['Weather Type'].value_counts()
```

Out[11]:
```
Weather Type
Snowy     1534
Cloudy     296
Rainy      239
Sunny      239
Name: count, dtype: int64
```

In [12]:
```
import seaborn as sns
for j in G.columns:
    if(G[j].dtype=="int64")|(G[j].dtype=="float64"):
        print(j.upper())
        sns.histplot(G[j],bins=20,kde=True)
        plt.show()
```

```
TEMPERATURE

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [13]:
```
G1=G.loc[G['Weather Type']=="Snowy"]
G1
```

Out[13]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
8	3.0	83	6.0	66.0	overcast	999.44	0	Winter	
20	-10.0	67	8.5	75.0	overcast	991.53	1	Winter	
37	2.0	105	19.0	109.0	overcast	991.68	7	Winter	
49	-2.0	98	18.5	86.0	overcast	981.26	0	Winter	
52	-4.0	96	20.0	88.0	partly cloudy	990.72	1	Winter	
...	...	...	...	...	...	...	...	...	
13162	-5.0	81	4.0	99.0	overcast	987.68	1	Winter	
13175	-6.0	87	9.0	97.0	partly cloudy	995.42	1	Winter	
13179	-4.0	98	1.5	76.0	overcast	988.72	0	Winter	
13181	-7.0	97	0.5	99.0	overcast	983.91	0	Winter	
13187	4.0	71	18.5	88.0	overcast	992.81	1	Winter	

1534 rows × 12 columns

In [14]:
```
1 G1['Cloud Cover'].value_counts()
```

Out[14]:
```
Cloud Cover
overcast         1196
partly cloudy     328
cloudy             10
Name: count, dtype: int64
```

In [15]:
```
1 G1.groupby('Cloud Cover')[['Atmospheric Pressure']].mean()
```

Out[15]:

Cloud Cover	Atmospheric Pressure
cloudy	980.276000
overcast	990.106614
partly cloudy	990.666921

In [16]:
```
1 G1['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[16]:

	UV Index	count
0	1	683
1	0	660
2	12	20
3	5	18
4	14	17
5	4	16
6	10	16
7	13	16
8	7	14
9	6	14

In [17]:
```
1 G1['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[17]:

	Visibility (km)	count
0	1.5	208
1	2.0	193
2	4.0	185
3	3.5	182
4	3.0	176
5	2.5	174
6	4.5	159
7	5.0	103
8	1.0	100
9	0.5	22

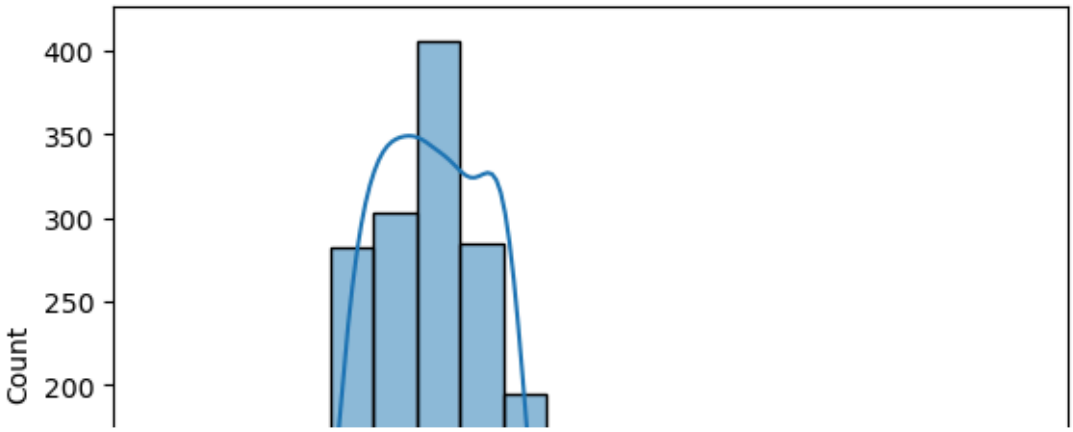
In [18]:
```
1 G1['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[18]:

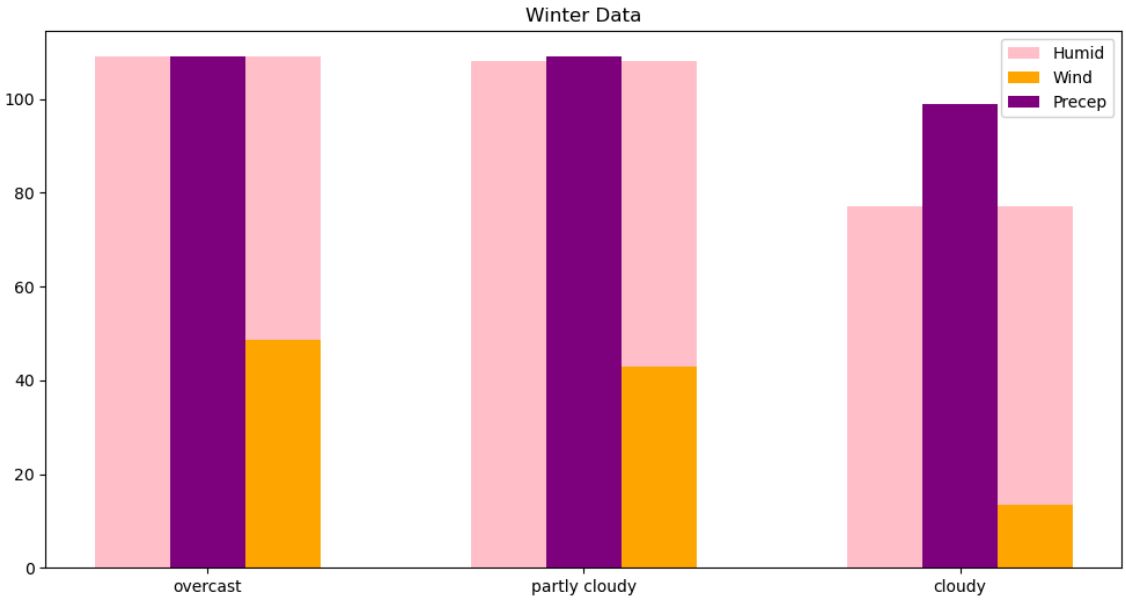
	Temperature C	count
0	-21.777778	120
1	-15.777778	111
2	-24.777778	111
3	-19.777778	110
4	-27.777778	100
5	-23.777778	98
6	-18.777778	97
7	-14.777778	94
8	-22.777778	94
9	-16.777778	90

In [19]:
```
import seaborn as sns
for j in G1.columns:
    if(G1[j].dtype=="int64")|(G1[j].dtype=="float64"):
        print(j.upper())
        sns.histplot(G1[j],bins=20,kde=True)
        plt.show()
```

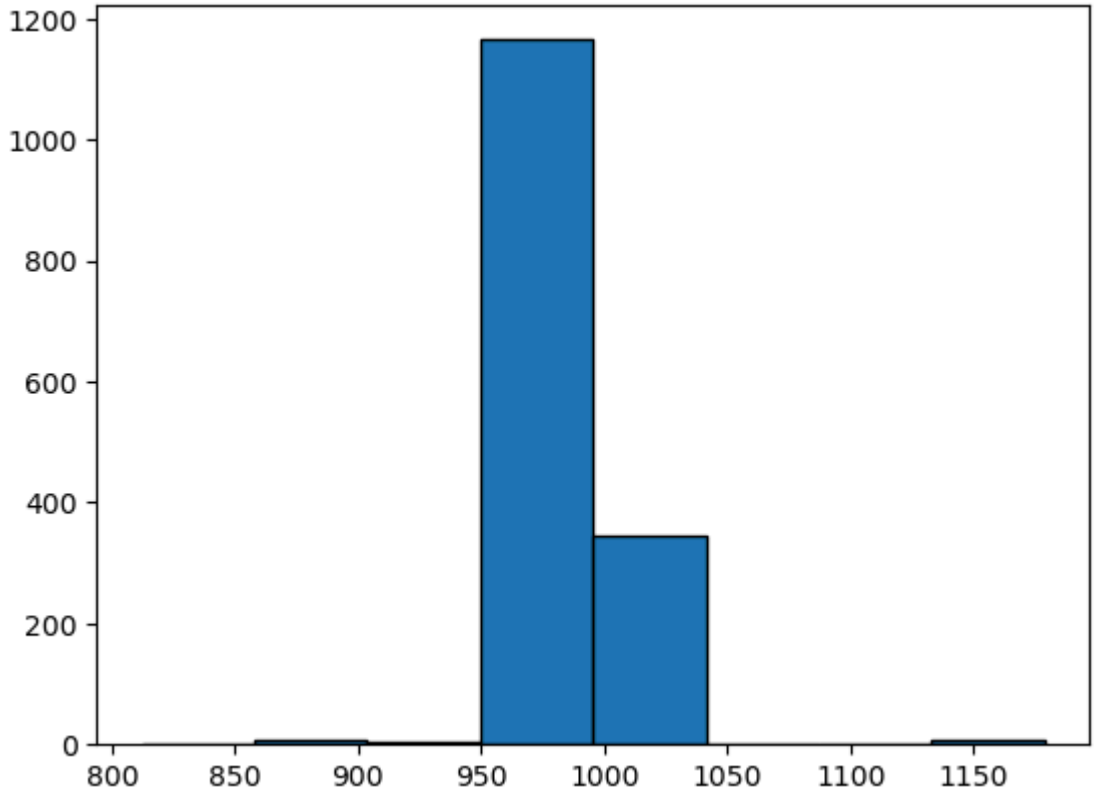
```
TEMPERATURE

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [20]:
```
plt.figure(figsize=(12,6))
plt.bar(G1['Cloud Cover'],G1['Humidity'],color="pink",width=0.6,align='
plt.bar(G1['Cloud Cover'],G1['Wind Speed'],color="orange",width=0.3,ali
plt.bar(G1['Cloud Cover'],G1['Precipitation (%)'],color="purple",width=

plt.title("Winter Data")
plt.legend()
plt.show()
```



In [21]:
```
plt.hist(G1['Atmospheric Pressure'],bins=8,edgecolor="black")
plt.show()#IN winter atm pressure is below 1000
```



In [22]:
```
G2=G.loc[G['Weather Type']=='Rainy']
G2
```

Out[22]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
4	27.0	74	17.0	66.0	overcast	990.67	1	Winter	
29	30.0	70	16.0	54.0	partly cloudy	1007.75	0	Winter	
33	15.0	97	23.0	91.0	overcast	1009.36	13	Winter	
36	11.0	90	25.5	76.0	partly cloudy	996.17	0	Winter	
61	29.0	63	6.0	68.0	overcast	992.45	0	Winter	
...	...	...	...	...	...	...	...	...	
12669	11.0	91	34.5	82.0	overcast	1007.44	7	Winter	
12714	31.0	66	7.0	38.0	partly cloudy	820.69	9	Winter	
12779	11.0	78	6.5	96.0	overcast	1003.38	2	Winter	
12854	20.0	92	11.5	53.0	overcast	1005.88	1	Winter	
12973	23.0	76	12.5	52.0	overcast	1011.87	3	Winter	

239 rows × 12 columns

In [23]:
```
1 G2['Cloud Cover'].value_counts()
```

Out[23]:
```
Cloud Cover
overcast         162
partly cloudy     71
cloudy             6
Name: count, dtype: int64
```

In [24]:

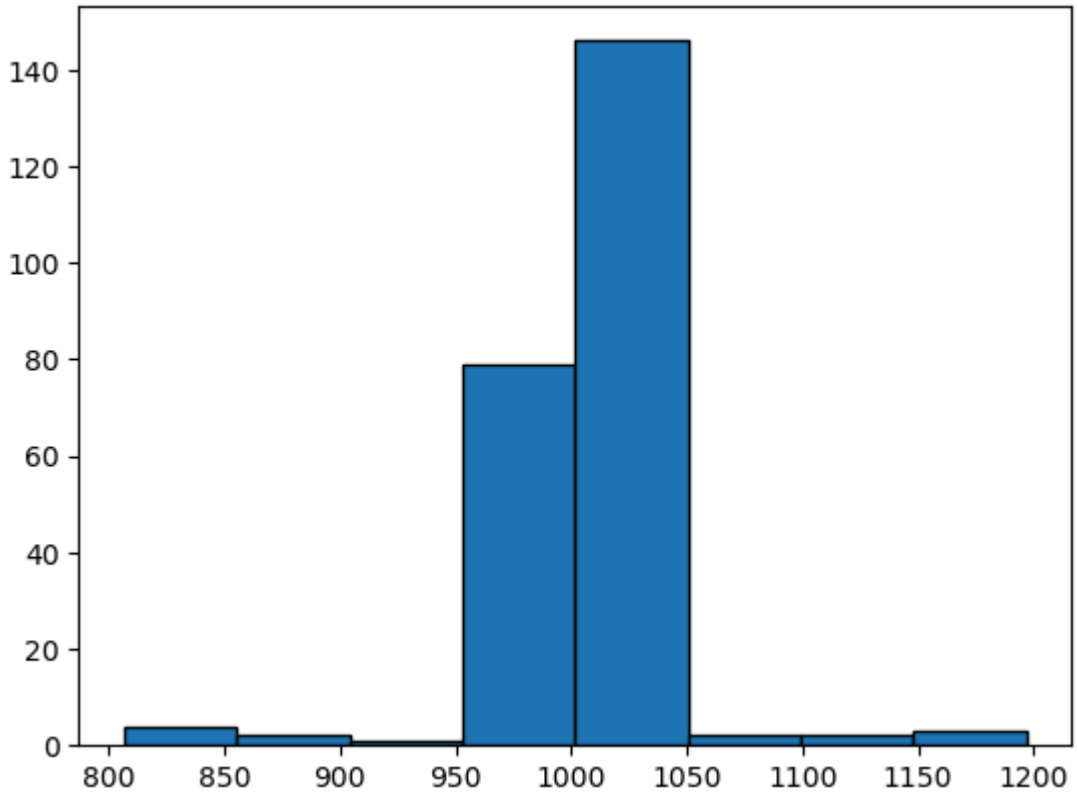
```
for i in G2.columns:
    if(G2[i].dtype=="int64")|(G2[i].dtype=="float64"):
        print(i.upper())
        print(G2[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
TEMPERATURE
   Temperature  count
0         11.0     16
1         24.0     12
2         15.0     11
3         28.0     11
4         31.0     11
5         12.0     11
-----
HUMIDITY
   Humidity  count
0        94     12
1        78      9
2        97      9
3        85      9
4        95      8
5        93      8
-----
WIND SPEED
   Wind Speed  count
0        12.5     12
1        10.5     12
2         9.0     12
3         6.5     12
4        16.0     11
5        18.0     11
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               82.0      9
1               87.0      8
2               91.0      8
3               53.0      8
4               75.0      8
5               86.0      8
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1015.07      2
1               1014.02      2
2                990.67      1
3               1014.97      1
4               1169.16      1
5               1014.82      1
-----
UV INDEX
   UV Index  count
0         3     53
1         0     52
2         2     51
3         1     50
4        13      5
5         6      5
-----
VISIBILITY (KM)
   Visibility (km)  count
0              1.5     39
1              4.0     30
2              3.0     29
3              2.5     28
4              3.5     23
```

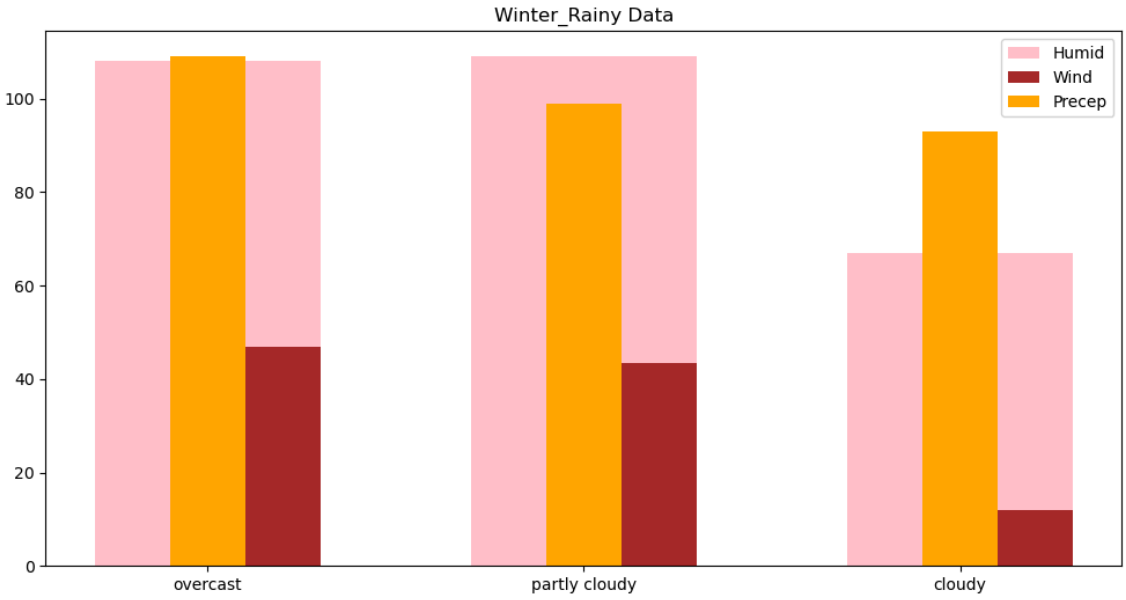
```
5              2.0     22
-----
TEMPERATURE C
   Temperature C  count
0      -6.777778     16
1       6.222222     12
2      -2.777778     11
3      10.222222     11
4      13.222222     11
5      -5.777778     11
-----
```

In [25]:

```
plt.hist(G2['Atmospheric Pressure'],bins=8,edgecolor="black")
plt.show()
```



In [26]:
```
plt.figure(figsize=(12,6))
plt.bar(G2['Cloud Cover'],G2['Humidity'],color="pink",width=0.6,align='
plt.bar(G2['Cloud Cover'],G2['Wind Speed'],color="brown",width=0.3,alig
plt.bar(G2['Cloud Cover'],G2['Precipitation (%)'],color="orange",width=

plt.title("Winter_Rainy Data")
plt.legend()
plt.show()
```



In [27]:
```
G3=G.loc[G['Weather Type']=='Sunny']
G3
```

Out[27]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Vis
47	26.0	25	2.5	14.0	partly cloudy	1010.34	8	Winter	
101	20.0	24	3.0	6.0	clear	1013.68	9	Winter	
304	35.0	58	3.0	19.0	clear	1015.40	5	Winter	
356	23.0	28	5.0	0.0	partly cloudy	1016.71	11	Winter	
442	59.0	106	8.5	109.0	clear	1023.39	13	Winter	
...	...	...	...	...	...	...	...	...	
12907	44.0	54	2.5	14.0	clear	1015.23	10	Winter	
12915	34.0	66	4.5	1.0	clear	1022.74	6	Winter	
12989	25.0	24	2.5	7.0	partly cloudy	1014.93	6	Winter	
13065	97.0	95	7.5	79.0	clear	1029.30	9	Winter	
13154	31.0	55	7.0	7.0	partly cloudy	1013.42	9	Winter	

239 rows × 12 columns

In [28]:
```
1 G3['Cloud Cover'].value_counts()
```

Out[28]:
```
Cloud Cover
clear            168
partly cloudy     57
cloudy            10
overcast           4
Name: count, dtype: int64
```

In [29]:

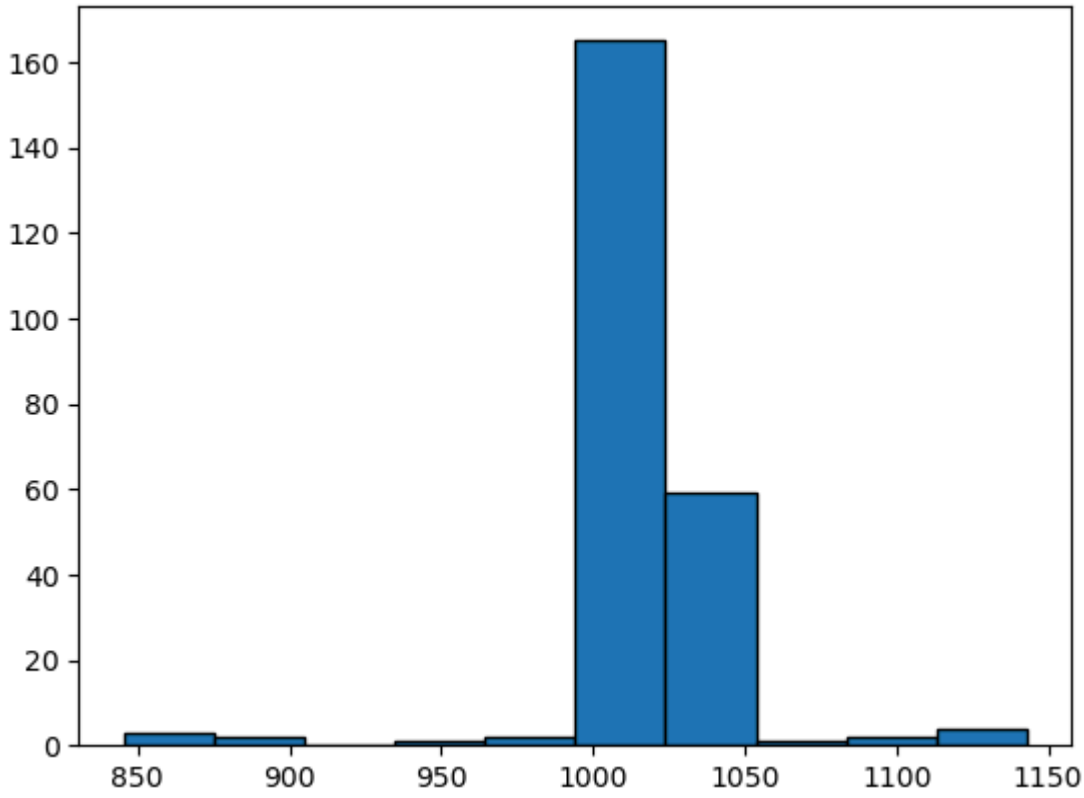
```
for i in G3.columns:
    if(G3[i].dtype=="int64")|(G3[i].dtype=="float64"):
        print(i.upper())
        print(G3[i].value_counts().head(6).reset_index().sort_values(by

        print("-"*5)
```

```
TEMPERATURE
   Temperature  count
0         32.0     16
1         41.0     13
2         28.0     11
3         29.0     11
4         34.0     10
5         40.0     10
-----
HUMIDITY
   Humidity  count
0        24      9
1        51      8
2        66      8
3        46      8
4        47      7
5        65      6
-----
WIND SPEED
   Wind Speed  count
0         2.5     16
1         5.5     15
2         6.0     15
3         8.5     14
4         1.5     13
5         9.0     13
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               19.0     19
1               11.0     13
2                5.0     12
3               14.0     11
4                2.0     11
5                4.0     11
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1015.11      2
1               1020.05      2
2               1020.17      2
3               1022.87      2
4               1020.33      2
5               1015.43      2
-----
UV INDEX
   UV Index  count
0         5     43
1         9     39
2        10     30
3         6     29
4        11     27
5         7     26
-----
VISIBILITY (KM)
   Visibility (km)  count
0              5.5     28
1              7.0     26
2              9.5     26
3              7.5     22
4              8.5     21
```

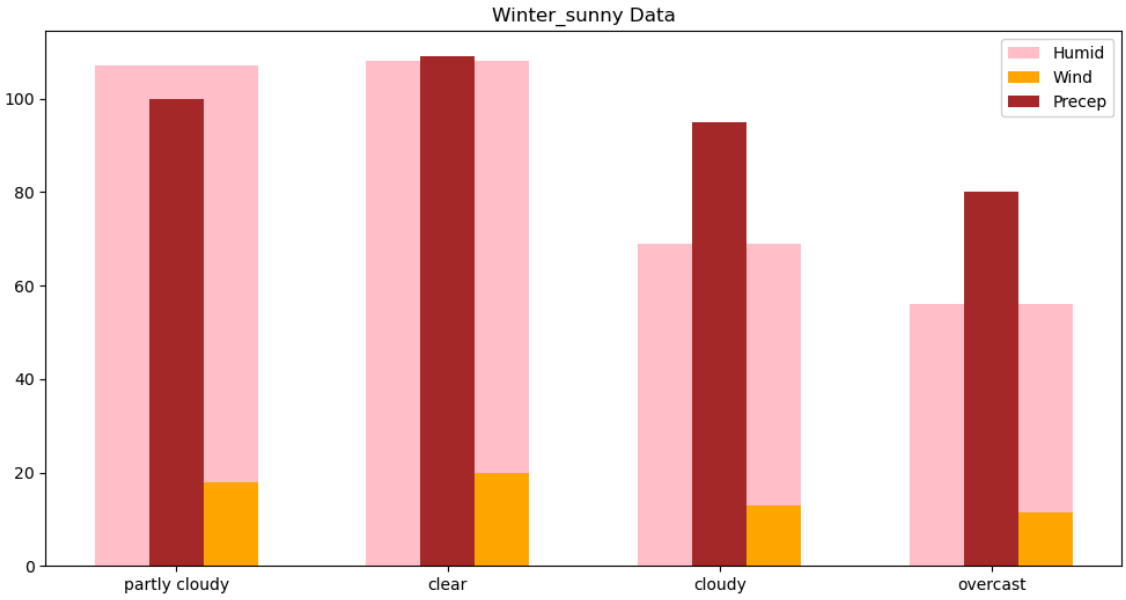
```
5              6.5     20
-----
TEMPERATURE C
   Temperature C  count
0      14.222222     16
1      23.222222     13
2      10.222222     11
3      11.222222     11
4      16.222222     10
5      22.222222     10
-----
```

In [30]:

```
plt.hist(G3['Atmospheric Pressure'],bins=10,edgecolor="black")
plt.show()
```



In [31]:
```
plt.figure(figsize=(12,6))
plt.bar(G3['Cloud Cover'],G3['Humidity'],color="pink",width=0.6,align='
plt.bar(G3['Cloud Cover'],G3['Wind Speed'],color="orange",width=0.3,ali
plt.bar(G3['Cloud Cover'],G3['Precipitation (%)'],color="brown",width=0

plt.title("Winter_sunny Data")
plt.legend()
plt.show()
```



In [32]:
```
G4=G.loc[G['Weather Type']=="Cloudy"]
G4
```

Out[32]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
42	28.0	54	15.0	39.0	partly cloudy	1018.86	3	Winter	
71	21.0	75	3.5	41.0	partly cloudy	1017.74	2	Winter	
88	12.0	56	9.5	19.0	partly cloudy	1014.40	4	Winter	
149	33.0	77	1.0	22.0	partly cloudy	1018.78	4	Winter	
310	23.0	62	10.0	20.0	overcast	1003.85	3	Winter	
...	...	...	...	...	...	...	...	...	
12993	16.0	93	7.5	103.0	partly cloudy	1013.65	9	Winter	
13006	11.0	53	1.0	24.0	overcast	1011.77	4	Winter	
13027	10.0	65	0.0	44.0	overcast	1010.10	1	Winter	
13139	18.0	69	12.0	28.0	partly cloudy	1004.65	1	Winter	
13166	26.0	57	2.5	34.0	overcast	1014.82	3	Winter	

296 rows × 12 columns

In [33]:
```
1 G4['Cloud Cover'].value_counts()
```

Out[33]:
```
Cloud Cover
partly cloudy    167
overcast         124
cloudy             5
Name: count, dtype: int64
```

In [34]:

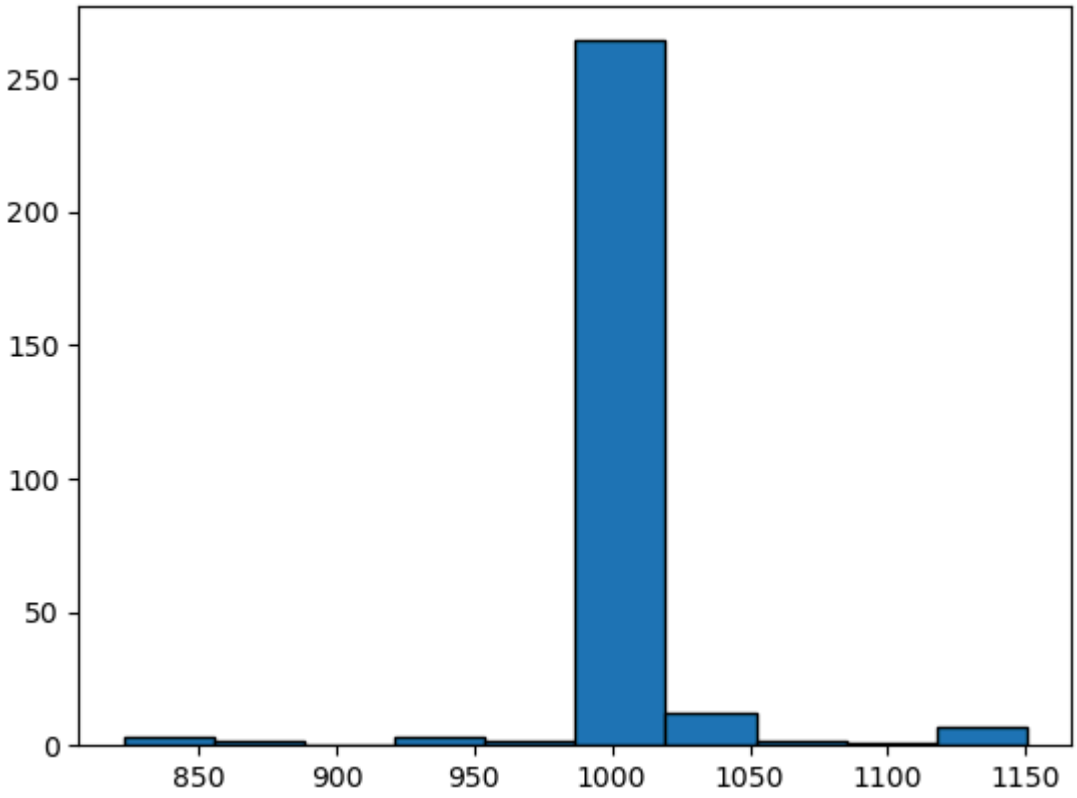
```
for i in G4.columns:
    if(G4[i].dtype=="int64")|(G4[i].dtype=="float64"):
        print(i.upper())
        print(G4[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
TEMPERATURE
   Temperature  count
0         11.0     17
1         24.0     16
2         33.0     16
3         13.0     15
4         31.0     15
5         22.0     14
-----
HUMIDITY
   Humidity  count
0        77     15
1        53     14
2        67     14
3        76     13
4        52     13
5        75     12
-----
WIND SPEED
   Wind Speed  count
0         2.0     16
1         4.0     15
2         3.0     13
3         3.5     13
4        12.0     12
5         2.5     11
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               26.0     13
1               43.0     12
2               27.0     10
3               20.0      9
4               35.0      8
5               22.0      8
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1005.74      2
1               1000.66      2
2               1005.17      2
3               1010.57      2
4               1010.05      2
5               1003.17      2
-----
UV INDEX
   UV Index  count
0         2     69
1         3     68
2         1     63
3         4     54
4        13      8
5         6      5
-----
VISIBILITY (KM)
   Visibility (km)  count
0              7.5     46
1              8.0     37
2              5.5     36
3              6.0     33
4              8.5     33
```

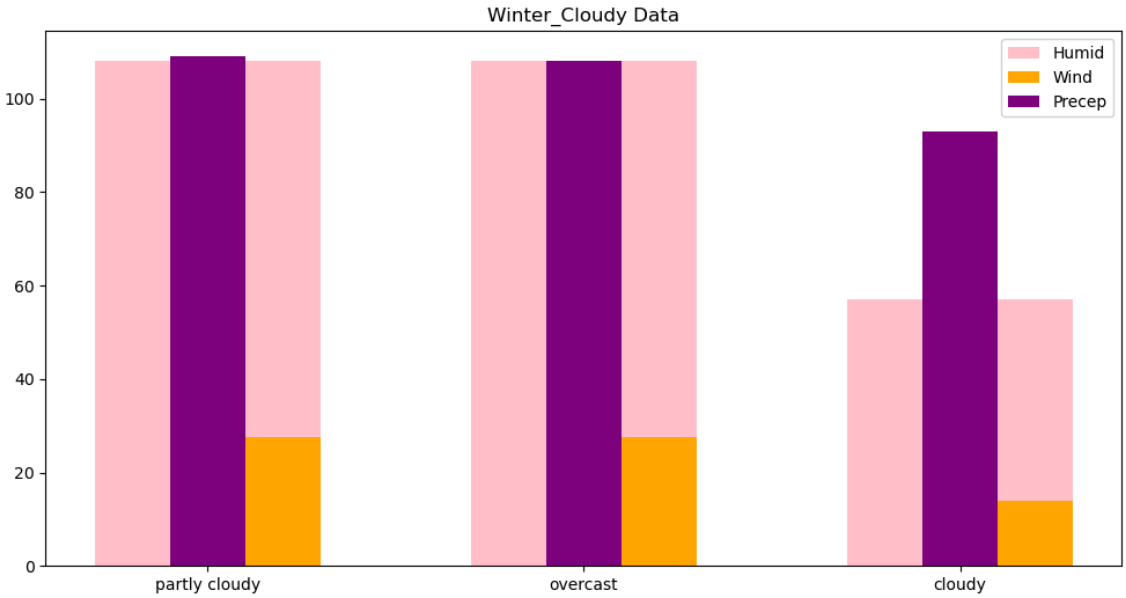
```
5              7.0     31
-----
TEMPERATURE C
   Temperature C  count
0      -6.777778     17
1       6.222222     16
2      15.222222     16
3      -4.777778     15
4      13.222222     15
5       4.222222     14
-----
```

In [35]:

```
1 plt.hist(G4['Atmospheric Pressure'],bins=10,edgecolor="black")
2 plt.show()
```



In [36]:
```
plt.figure(figsize=(12,6))
plt.bar(G4['Cloud Cover'],G4['Humidity'],color="pink",width=0.6,align='
plt.bar(G4['Cloud Cover'],G4['Wind Speed'],color="orange",width=0.3,ali
plt.bar(G4['Cloud Cover'],G4['Precipitation (%)'],color="purple",width=

plt.title("Winter_Cloudy Data")
plt.legend()
plt.show()
```



In [37]:
```
H=B.loc[B['Season']=="Spring"]#SPRING:Cloudy
H
```

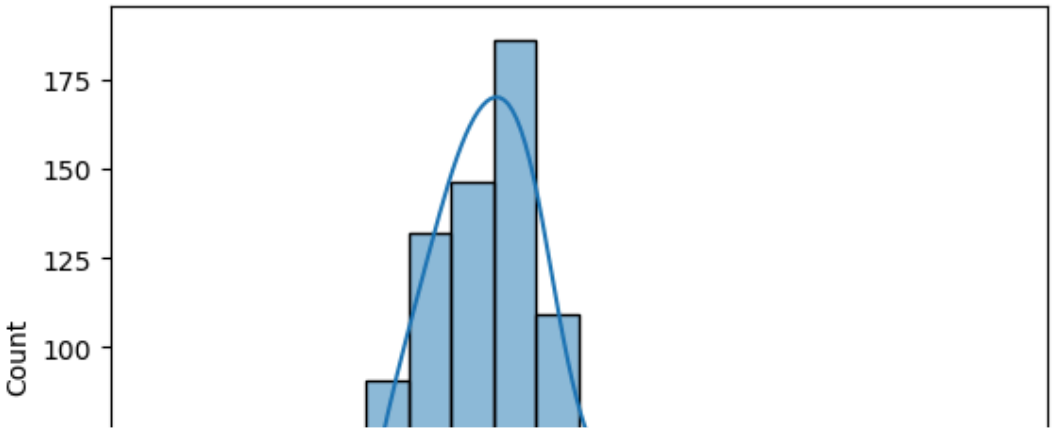
Out[37]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
2	30.0	64	7.0	16.0	clear	1018.72	5	Spring	
10	35.0	45	6.0	86.0	partly cloudy	879.88	2	Spring	
18	43.0	46	0.5	15.0	clear	1025.80	9	Spring	
24	38.0	83	7.0	101.0	partly cloudy	1017.94	4	Spring	
38	-9.0	49	1.5	58.0	partly cloudy	1132.20	8	Spring	
...	...	...	...	...	...	...	...	...	
13157	33.0	48	1.5	0.0	clear	1024.96	11	Spring	
13163	5.0	67	9.5	71.0	partly cloudy	1122.86	4	Spring	
13165	14.0	88	11.5	74.0	overcast	994.87	2	Spring	
13172	33.0	54	0.5	15.0	clear	1012.66	7	Spring	
13176	19.0	100	36.0	86.0	overcast	995.12	5	Spring	

851 rows × 12 columns

In [38]:
```
import seaborn as sns
for j in H.columns:
    if(H[j].dtype=="int64")|(H[j].dtype=="float64"):
        print(j.upper())
        sns.histplot(H[j],bins=20,kde=True)
        plt.show()
```

```
TEMPERATURE

C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [39]:
```
H['Weather Type'].value_counts()
```

Out[39]:
```
Weather Type
Cloudy    280
Sunny     276
Rainy     266
Snowy      29
Name: count, dtype: int64
```

In [40]:
```
H['Cloud Cover'].value_counts()
```

Out[40]:
```
Cloud Cover
partly cloudy    335
overcast         306
clear            174
cloudy            36
Name: count, dtype: int64
```

In [41]:
```
H1=H.loc[H['Weather Type']=='Cloudy']
H1
```

Out[41]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
10	35.0	45	6.0	86.0	partly cloudy	879.88	2	Spring	
24	38.0	83	7.0	101.0	partly cloudy	1017.94	4	Spring	
130	23.0	52	1.0	36.0	partly cloudy	1017.92	1	Spring	
283	19.0	72	11.0	25.0	overcast	1009.68	4	Spring	
352	13.0	75	0.5	33.0	overcast	1006.64	1	Spring	
...	...	...	...	...	...	...	...	...	
13038	30.0	71	14.5	20.0	partly cloudy	1014.07	4	Spring	
13046	18.0	79	8.0	32.0	partly cloudy	1007.36	1	Spring	
13115	40.0	75	26.5	78.0	partly cloudy	1011.36	0	Spring	
13135	27.0	63	1.5	31.0	partly cloudy	1011.62	4	Spring	
13163	5.0	67	9.5	71.0	partly cloudy	1122.86	4	Spring	

280 rows × 12 columns

In [42]:
```
H1['Cloud Cover'].value_counts()
```

Out[42]:
```
Cloud Cover
partly cloudy    153
overcast         115
cloudy            12
Name: count, dtype: int64
```

In [43]:

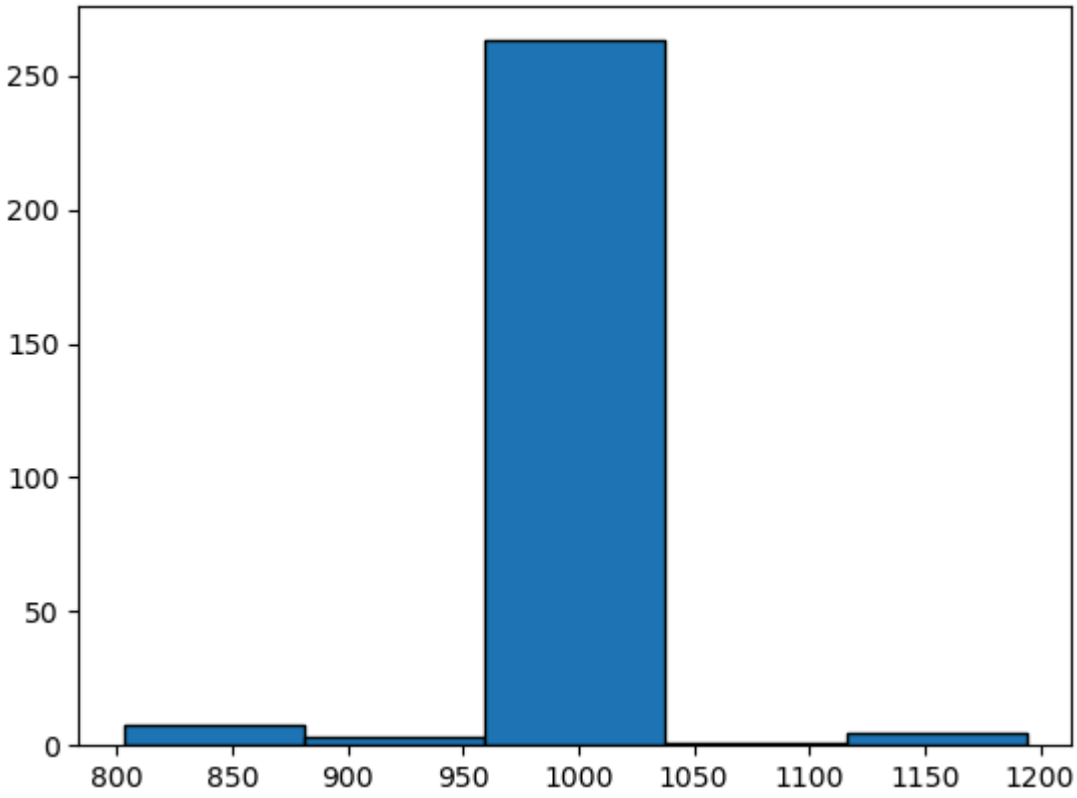
```
for i in H1.columns:
    if(H1[i].dtype=="int64")|(H1[i].dtype=="float64"):
        print(i.upper())
        print(H1[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
TEMPERATURE
   Temperature  count
0         30.0     14
1         27.0     14
2         25.0     13
3         26.0     13
4         32.0     13
5         31.0     12
-----
HUMIDITY
   Humidity  count
0        67     14
1        63     13
2        76     12
3        75     12
4        73     11
5        61     11
-----
WIND SPEED
   Wind Speed  count
0         8.0     17
1         9.5     13
2         9.0     13
3         8.5     13
4        13.0     12
5         1.5     12
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               45.0     11
1               20.0     11
2               25.0     10
3               23.0      9
4               39.0      9
5               19.0      9
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1000.60      3
1               1014.35      2
2               1003.09      2
3               1006.95      2
4               1001.36      2
5               1007.40      2
-----
UV INDEX
   UV Index  count
0         3     65
1         1     59
2         2     57
3         4     56
4        10      6
5         5      6
-----
VISIBILITY (KM)
   Visibility (km)  count
0              7.5     35
1              5.5     34
2              6.5     32
3              8.5     29
4              8.0     27
```

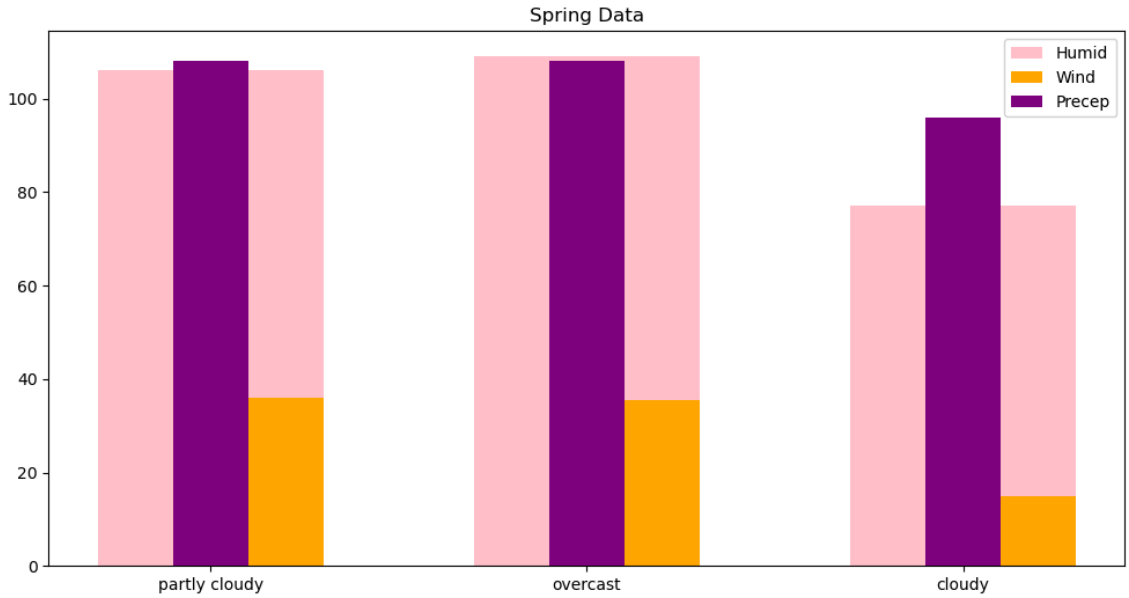
```
5              7.0     27
-----
TEMPERATURE C
   Temperature C  count
0      12.222222     14
1       9.222222     14
2       7.222222     13
3       8.222222     13
4      14.222222     13
5      13.222222     12
-----
```

In [44]:

```
plt.hist(H1['Atmospheric Pressure'],bins=5,edgecolor="black")
plt.show()
```



In [45]:
```
plt.figure(figsize=(12,6))
plt.bar(H1['Cloud Cover'],H1['Humidity'],color="pink",width=0.6,align='
plt.bar(H1['Cloud Cover'],H1['Wind Speed'],color="orange",width=0.3,ali
plt.bar(H1['Cloud Cover'],H1['Precipitation (%)'],color="purple",width=

plt.title("Spring Data")
plt.legend()
plt.show()
```



In [46]:
```
H2=H.loc[H['Weather Type']=='Rainy']
H2
```

Out[46]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
112	18.0	70	5.0	94.0	overcast	1012.87	2	Spring	
147	31.0	87	11.0	54.0	partly cloudy	995.87	0	Spring	
213	19.0	97	9.0	56.0	overcast	1015.74	3	Spring	
231	30.0	71	12.0	92.0	partly cloudy	992.51	2	Spring	
253	12.0	92	20.0	51.0	overcast	1010.65	3	Spring	
...	...	...	...	...	...	...	...	...	
13056	17.0	68	10.0	54.0	overcast	991.09	3	Spring	
13088	26.0	98	11.5	82.0	overcast	1019.64	0	Spring	
13113	42.0	106	32.5	103.0	partly cloudy	996.45	14	Spring	
13165	14.0	88	11.5	74.0	overcast	994.87	2	Spring	
13176	19.0	100	36.0	86.0	overcast	995.12	5	Spring	

266 rows × 12 columns

In [47]:
```
1 H2['Cloud Cover'].value_counts()
```

Out[47]:
```
Cloud Cover
overcast        175
partly cloudy    82
cloudy            9
Name: count, dtype: int64
```

In [48]:

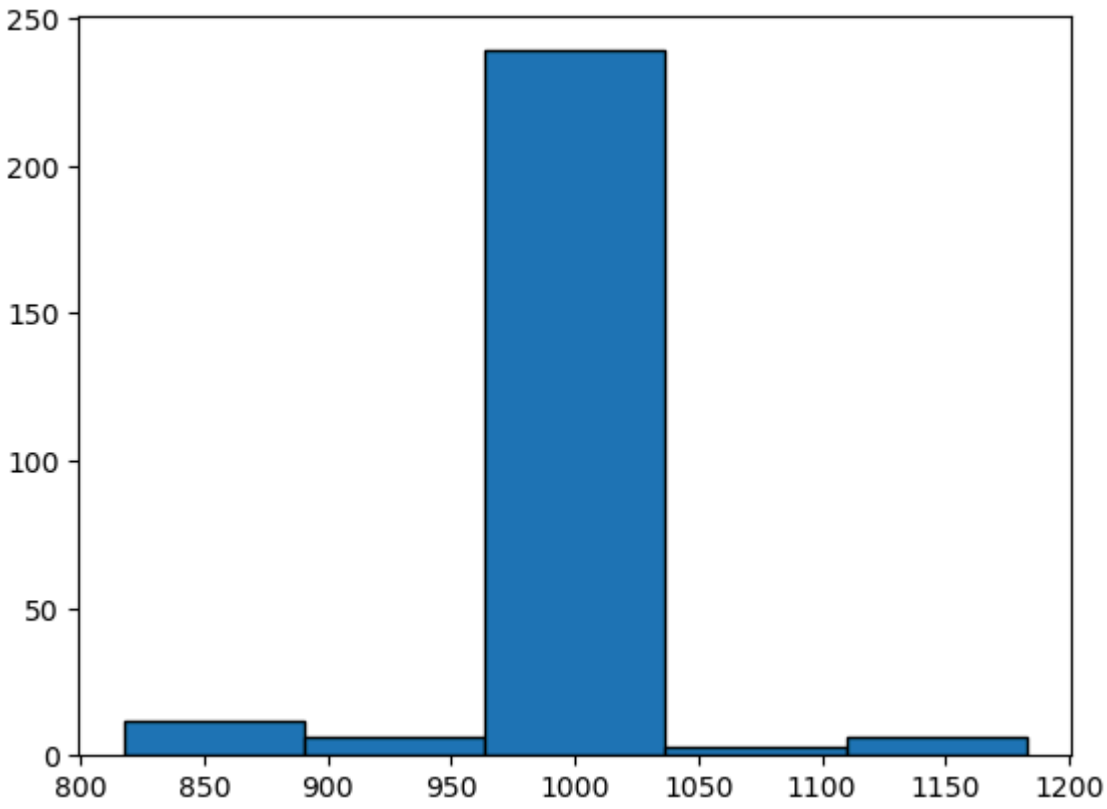
```
for i in H2.columns:
    if(H2[i].dtype=="int64")|(H2[i].dtype=="float64"):
        print(i.upper())
        print(H2[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
TEMPERATURE
   Temperature  count
0         19.0     19
1         22.0     14
2         17.0     13
3         11.0     13
4         26.0     12
5         21.0     11
-----
HUMIDITY
   Humidity  count
0        76     10
1        82     10
2        69     10
3        85      9
4        90      9
5        70      8
-----
WIND SPEED
   Wind Speed  count
0         9.5     14
1         8.0     12
2         7.5     12
3        11.0     11
4        18.5     10
5        16.5     10
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               94.0      9
1               66.0      9
2               99.0      9
3               53.0      9
4               93.0      9
5               92.0      8
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0                994.22      2
1               1013.75      2
2               1002.00      2
3                993.03      2
4               1004.05      2
5                990.55      2
-----
UV INDEX
   UV Index  count
0         1     64
1         2     54
2         0     51
3         3     48
4        11     12
5        10      6
-----
VISIBILITY (KM)
   Visibility (km)  count
0              2.5     35
1              3.0     34
2              1.5     32
3              2.0     27
4              4.5     27
```

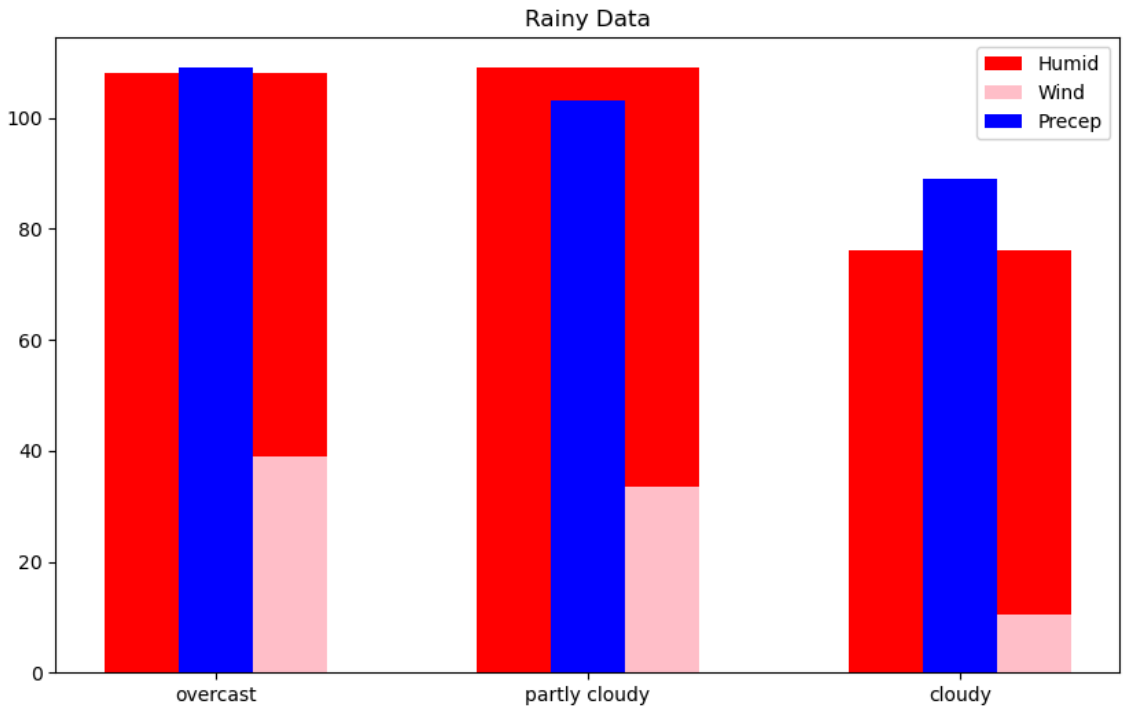
```
5              4.0     23
-----
TEMPERATURE C
   Temperature C  count
0       1.222222     19
1       4.222222     14
2      -0.777778     13
3      -6.777778     13
4       8.222222     12
5       3.222222     11
-----
```

In [49]:

```
plt.hist(H2['Atmospheric Pressure'],bins=5,edgecolor="black")
plt.show()
```



In [50]:
```
plt.figure(figsize=(10,6))
plt.bar(H2['Cloud Cover'],H2['Humidity'],color="red",width=0.6,align="
plt.bar(H2['Cloud Cover'],H2['Wind Speed'],color="pink",width=0.3,alig
plt.bar(H2['Cloud Cover'],H2['Precipitation (%)'],color="blue",width=0.

plt.title("Rainy Data")
plt.legend()
plt.show()
```



In [51]:
```
H3=H.loc[H['Weather Type']=='Sunny']
H3
```

Out[51]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Vis
2	30.0	64	7.0	16.0	clear	1018.72	5	Spring	
18	43.0	46	0.5	15.0	clear	1025.80	9	Spring	
44	21.0	43	10.0	11.0	clear	1019.41	11	Spring	
95	17.0	88	18.0	83.0	partly cloudy	1026.94	0	Spring	
137	46.0	94	4.0	90.0	clear	1022.88	10	Spring	
...	...	...	...	...	...	...	...	...	
13073	26.0	66	2.0	13.0	partly cloudy	1028.90	9	Spring	
13077	25.0	40	9.5	10.0	clear	1023.37	6	Spring	
13099	1.0	23	14.0	10.0	cloudy	1020.47	2	Spring	
13157	33.0	48	1.5	0.0	clear	1024.96	11	Spring	
13172	33.0	54	0.5	15.0	clear	1012.66	7	Spring	

276 rows × 12 columns

In [52]:
```
1 H3['Cloud Cover'].value_counts()
```

Out[52]:
```
Cloud Cover
clear            174
partly cloudy     88
overcast           7
cloudy             7
Name: count, dtype: int64
```

In [53]:

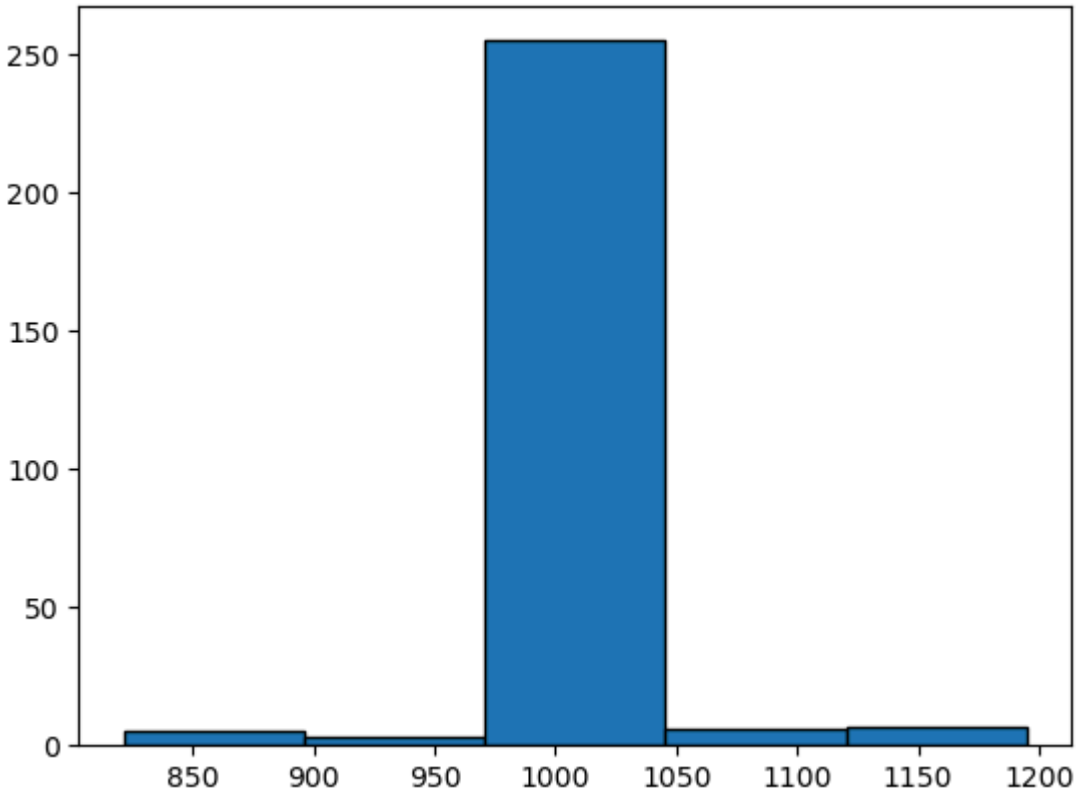
```
for i in H3.columns:
    if(H3[i].dtype=="int64")|(H3[i].dtype=="float64"):
        print(i.upper())
        print(H3[i].value_counts().head(6).reset_index().sort_values(by
        print("-"*5)
```

```
TEMPERATURE
   Temperature  count
0         31.0     14
1         22.0     14
2         40.0     14
3         44.0     14
4         28.0     13
5         29.0     12
-----
HUMIDITY
   Humidity  count
0        39     10
1        68      9
2        63      9
3        38      8
4        30      8
5        23      8
-----
WIND SPEED
   Wind Speed  count
0         1.5     20
1         8.0     17
2         5.5     14
3         2.0     14
4         4.0     14
5         3.0     13
-----
PRECIPITATION (%)
   Precipitation (%)  count
0               15.0     18
1                0.0     17
2                2.0     15
3               11.0     15
4                5.0     14
5                4.0     13
-----
ATMOSPHERIC PRESSURE
   Atmospheric Pressure  count
0               1021.59      3
1               1018.48      2
2               1022.23      2
3               1010.24      2
4               1024.35      2
5               1011.89      2
-----
UV INDEX
   UV Index  count
0        11     45
1         7     40
2         5     35
3         8     35
4        10     34
5         6     34
-----
VISIBILITY (KM)
   Visibility (km)  count
0              7.5     30
1              5.5     29
2              8.0     27
3              9.0     27
4              8.5     26
```

```
5              7.0     24
-----
TEMPERATURE C
   Temperature C  count
0      13.222222     14
1       4.222222     14
2      22.222222     14
3      26.222222     14
4      10.222222     13
5      11.222222     12
-----
```

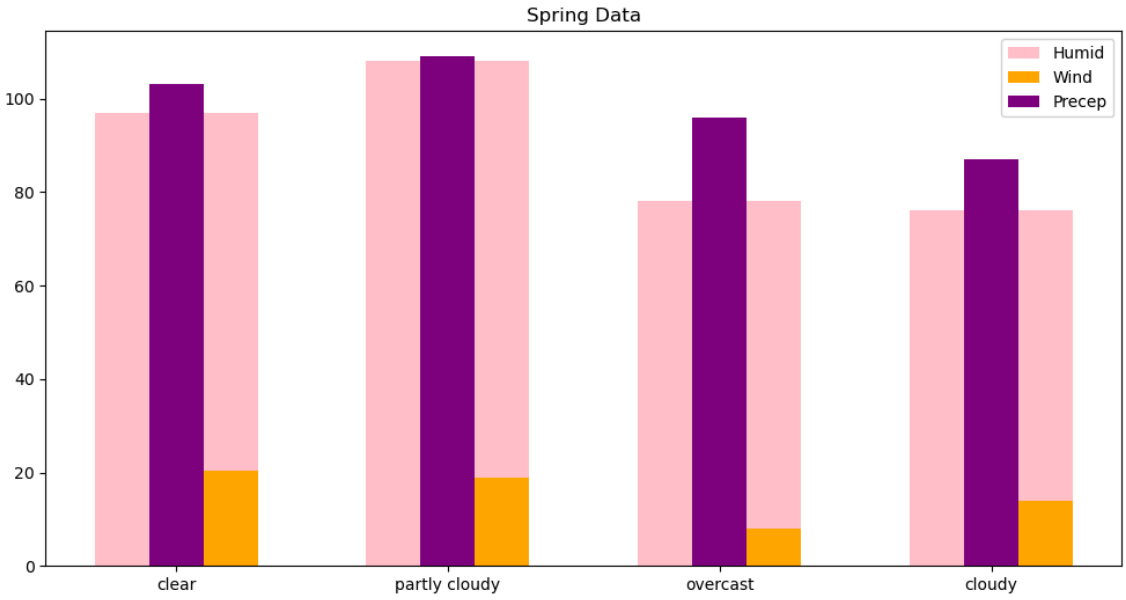
In [54]:

```
plt.hist(H3['Atmospheric Pressure'],bins=5,edgecolor="black")
plt.show()
```



In [55]:

```
plt.figure(figsize=(12,6))
plt.bar(H3['Cloud Cover'],H3['Humidity'],color="pink",width=0.6,align='
plt.bar(H3['Cloud Cover'],H3['Wind Speed'],color="orange",width=0.3,ali
plt.bar(H3['Cloud Cover'],H3['Precipitation (%)'],color="purple",width=

plt.title("Spring Data")
plt.legend()
plt.show()
```



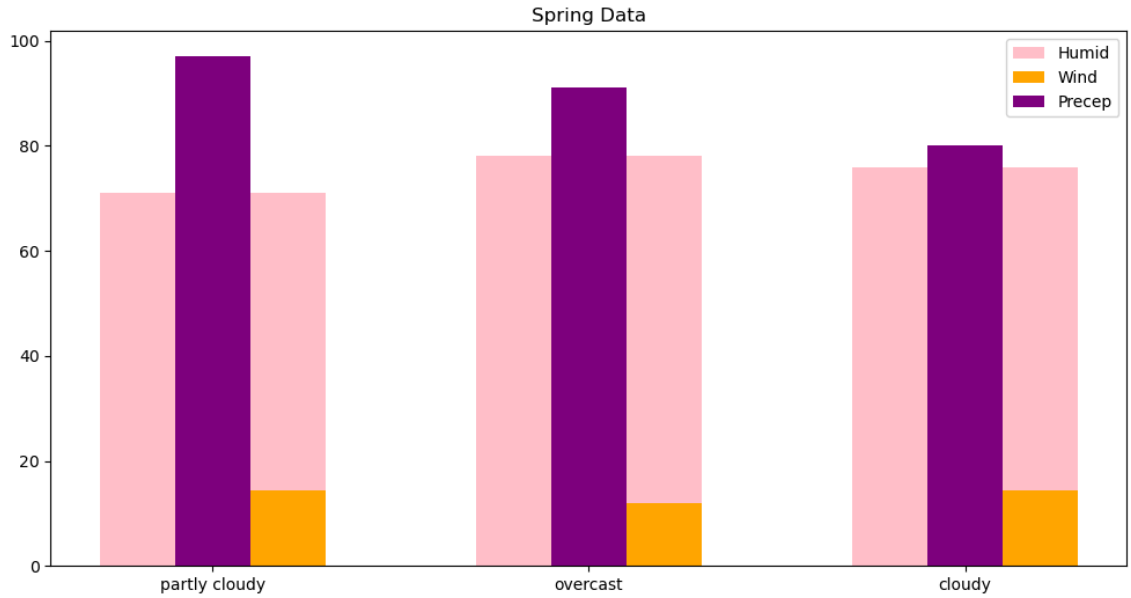
In [56]:

```
H4=H.loc[H['Weather Type']=='Snowy']
H4
```

Out[56]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	V
38	-9.0	49	1.5	58.0	partly cloudy	1132.20	8	Spring	
777	18.0	39	12.0	39.0	overcast	988.64	6	Spring	
947	10.0	76	11.5	71.0	cloudy	1095.61	12	Spring	
1057	-5.0	34	14.5	91.0	partly cloudy	1125.08	6	Spring	
2351	-14.0	60	14.0	68.0	partly cloudy	949.10	5	Spring	
2392	-14.0	78	5.5	16.0	overcast	1076.86	11	Spring	
2662	22.0	68	6.5	58.0	overcast	809.03	1	Spring	
3000	32.0	20	13.0	80.0	cloudy	1037.51	12	Spring	
3216	31.0	76	14.5	31.0	cloudy	1133.11	10	Spring	
4392	43.0	37	7.5	97.0	partly cloudy	1199.21	3	Spring	
4578	-3.0	52	10.0	55.0	overcast	817.52	3	Spring	
4940	-17.0	69	5.5	91.0	overcast	895.83	2	Spring	
4947	24.0	23	9.5	34.0	overcast	850.90	5	Spring	
5066	4.0	71	9.5	20.0	partly cloudy	842.79	3	Spring	
5302	38.0	43	2.5	17.0	partly cloudy	1010.58	10	Spring	
5508	-2.0	38	11.0	18.0	overcast	976.02	5	Spring	
5961	17.0	70	13.0	33.0	partly cloudy	1161.97	1	Spring	
6200	-11.0	71	7.5	14.0	partly cloudy	874.63	2	Spring	
6382	-6.0	40	7.5	79.0	overcast	1078.03	3	Spring	
8712	-8.0	71	11.5	14.0	partly cloudy	952.25	5	Spring	
8895	-9.0	42	6.5	11.0	overcast	1168.17	10	Spring	
10144	2.0	37	10.0	17.0	partly cloudy	1151.64	2	Spring	
10936	44.0	67	10.5	54.0	cloudy	1098.98	5	Spring	
10986	30.0	44	12.5	88.0	partly cloudy	969.76	10	Spring	
11439	21.0	57	2.0	67.0	cloudy	823.08	7	Spring	
11609	20.0	42	6.0	20.0	cloudy	1198.97	10	Spring	
11798	18.0	39	3.5	50.0	cloudy	1039.77	4	Spring	
12468	-14.0	38	1.5	52.0	partly cloudy	1111.93	9	Spring	
12885	-14.0	55	4.5	22.0	cloudy	1050.48	10	Spring	

In [57]:
```
plt.figure(figsize=(12,6))
plt.bar(H4['Cloud Cover'],H4['Humidity'],color="pink",width=0.6,align='
plt.bar(H4['Cloud Cover'],H4['Wind Speed'],color="orange",width=0.3,ali
plt.bar(H4['Cloud Cover'],H4['Precipitation (%)'],color="purple",width=

plt.title("Spring Data")
plt.legend()
plt.show()
```



In [58]:
```
I=B.loc[B['Season']=='Summer']#SUMMER:SUNNY
I
```

Out[58]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season
15	10.0	50	6.5	46.0	partly cloudy	1000.44	2	Summer
22	33.0	36	7.5	18.0	clear	1014.29	8	Summer
26	17.0	72	7.5	62.0	overcast	1010.16	2	Summer
41	36.0	22	1.0	0.0	clear	1028.63	10	Summer
48	26.0	91	9.5	82.0	overcast	1019.42	3	Summer
...	...	...	...	...	...	...	...	...
13153	20.0	87	17.5	57.0	partly cloudy	1007.44	0	Summer
13180	10.0	62	17.0	83.0	partly cloudy	1013.09	3	Summer
13184	3.0	62	7.5	14.0	overcast	1128.35	3	Summer
13190	30.0	24	3.5	16.0	partly cloudy	1017.54	11	Summer
13195	10.0	74	14.5	71.0	overcast	1003.15	1	Summer

848 rows × 12 columns

In [59]:
```
1 I['Cloud Cover'].value_counts()
```

Out[59]:
```
Cloud Cover
partly cloudy    321
overcast         313
clear            182
cloudy            32
Name: count, dtype: int64
```

In [60]:
```
1 I['Weather Type'].value_counts()
```

Out[60]:
```
Weather Type
Sunny     306
Cloudy    267
Rainy     258
Snowy      17
Name: count, dtype: int64
```

In [62]:
```
1 I1=I.loc[I['Weather Type']=='Sunny']
2 I1
```

Out[62]:

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Vis
22	33.0	36	7.5	18.0	clear	1014.29	8	Summer	
41	36.0	22	1.0	0.0	clear	1028.63	10	Summer	
66	25.0	89	1.5	87.0	partly cloudy	1017.16	11	Summer	
67	24.0	47	8.0	17.0	clear	1025.22	5	Summer	
69	21.0	54	3.0	17.0	clear	1026.15	7	Summer	
...	...	...	...	...	...	...	...	...	
12991	17.0	95	6.5	107.0	clear	1028.12	14	Summer	
13081	66.0	81	10.0	97.0	clear	1015.03	11	Summer	
13107	29.0	24	8.5	5.0	clear	1019.57	9	Summer	
13125	29.0	57	5.0	9.0	clear	1018.93	10	Summer	
13190	30.0	24	3.5	16.0	partly cloudy	1017.54	11	Summer	

306 rows × 12 columns

In [63]:
```
1 I1['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[63]:

	UV Index	count
0	8	45
1	11	40
2	10	38
3	5	36
4	9	35
5	7	31
6	6	30
7	12	10
8	4	9
9	0	8
10	14	7
11	2	5
12	3	5
13	13	4
14	1	3

In [64]:
```
1 I1['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[64]:

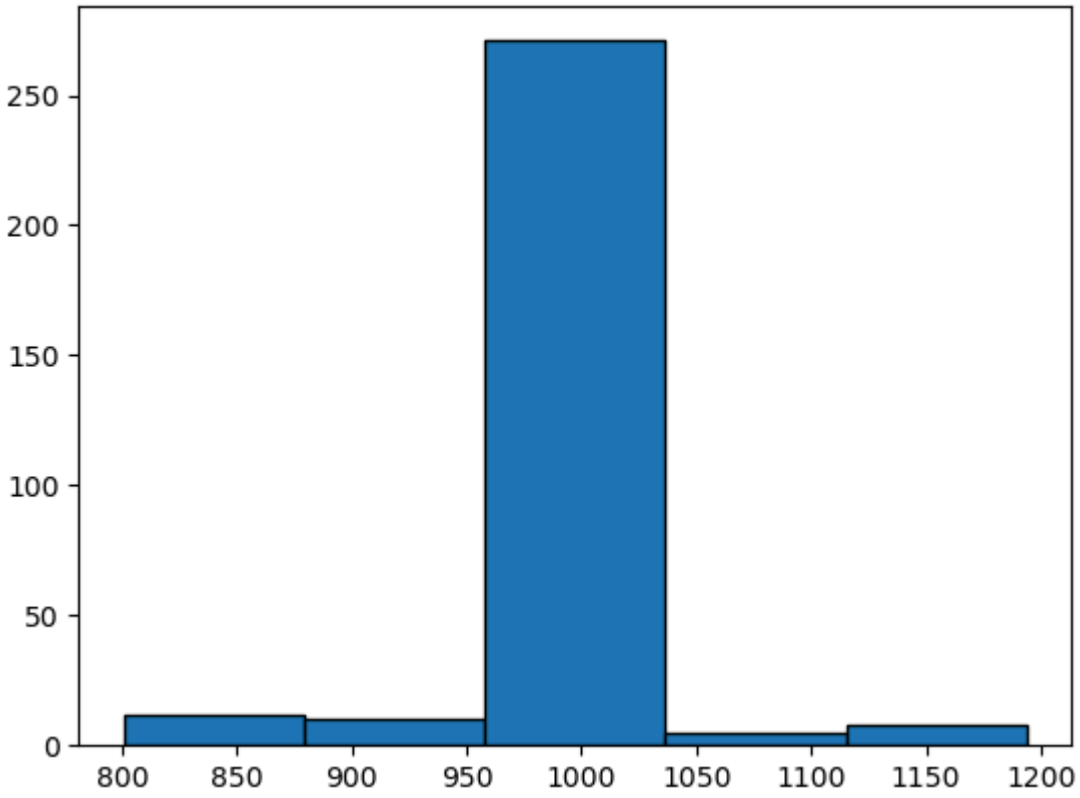
	Visibility (km)	count
0	8.0	33
1	6.5	32
2	9.5	30
3	6.0	29
4	7.5	28
5	9.0	26
6	5.5	22
7	7.0	19
8	8.5	19
9	5.0	13

In [65]:
```
I1.groupby(["Weather Type","Cloud Cover"]).agg({'Atmospheric Pressure'
```

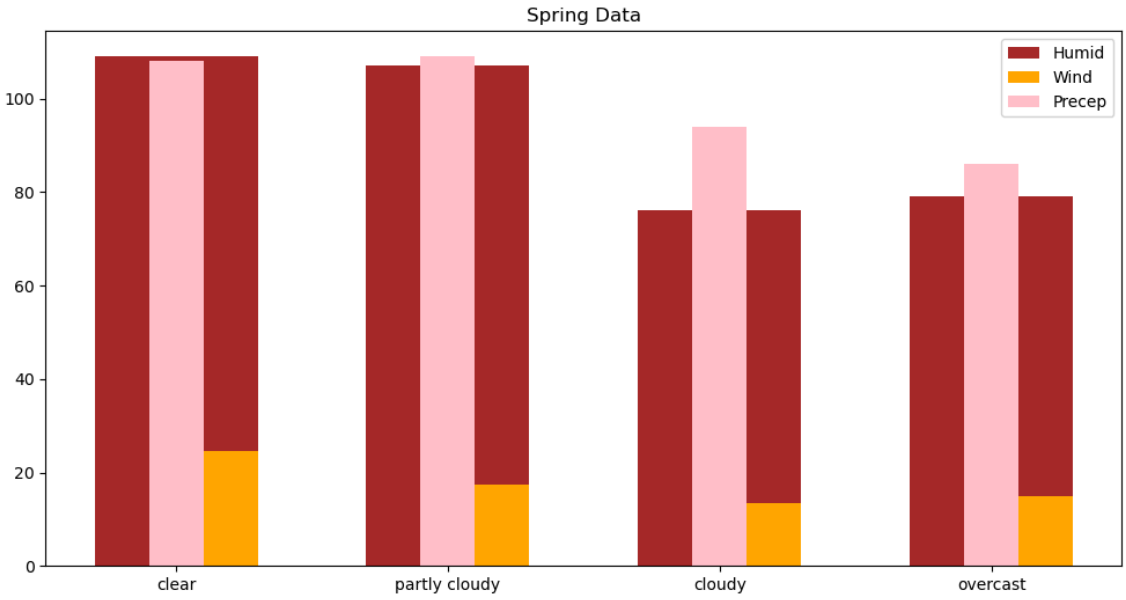
Out[65]:

	Weather Type	Cloud Cover	Atmospheric Pressure	Humidity	Temperature C		Wind Speed	Precipitation (%)
			mean	mean	min	max	mean	mean
0	Sunny	clear	1020.163132	55.472527	-6.777778	82.222222	6.071429	26.043956
1	Sunny	cloudy	925.359286	48.142857	-29.777778	14.222222	8.642857	60.785714
2	Sunny	overcast	934.719231	50.230769	-35.777778	29.222222	8.769231	52.307692
3	Sunny	partly cloudy	1024.548866	50.824742	-25.777778	89.222222	6.350515	23.979381

In [457]:
```
plt.hist(I1['Atmospheric Pressure'],bins=5,edgecolor="black")
plt.show()
```



In [458]:
```
plt.figure(figsize=(12,6))
plt.bar(I1['Cloud Cover'],I1['Humidity'],color="brown",width=0.6,align=
plt.bar(I1['Cloud Cover'],I1['Wind Speed'],color="orange",width=0.3,ali
plt.bar(I1['Cloud Cover'],I1['Precipitation (%)'],color="pink",width=0.
plt.title("Spring Data")
plt.legend()
plt.show()
```



In [459]:
```
I2=I.loc[I['Weather Type']=='Cloudy']
I2
```

Out[459]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
15	50	6.5	46.0	partly cloudy	1000.44	2	Summer	8.5	mou
56	85	9.0	101.0	overcast	1017.89	3	Summer	7.5	mou
162	50	3.5	10.0	overcast	1017.18	4	Summer	6.5	mou
167	59	11.5	37.0	partly cloudy	1004.12	4	Summer	6.5	mou
195	68	4.0	39.0	partly cloudy	1016.39	4	Summer	5.5	mou
...	...	...	...	...	...	...	...	...	
12551	75	3.0	17.0	overcast	1019.39	1	Summer	8.5	mou
12843	79	14.0	44.0	partly cloudy	1000.39	3	Summer	6.5	mou
12951	105	12.5	107.0	overcast	1019.08	12	Summer	9.0	mou
13103	71	1.5	43.0	partly cloudy	1006.75	2	Summer	8.0	mou
13110	54	6.0	37.0	partly cloudy	1000.23	2	Summer	8.5	mou

267 rows × 11 columns

In [460]:
```
1 I2['Cloud Cover'].value_counts()
```

Out[460]:
```
Cloud Cover
partly cloudy    150
overcast         114
cloudy             3
Name: count, dtype: int64
```

In [461]:
```
1 I2['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[461]:

	UV Index	count
0	4	67
1	1	59
2	2	54
3	3	54
4	12	8
5	14	5
6	9	4
7	10	4
8	7	3
9	6	2
10	11	2
11	13	2
12	8	1
13	0	1
14	5	1

In [462]:
```
1 I2['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[462]:

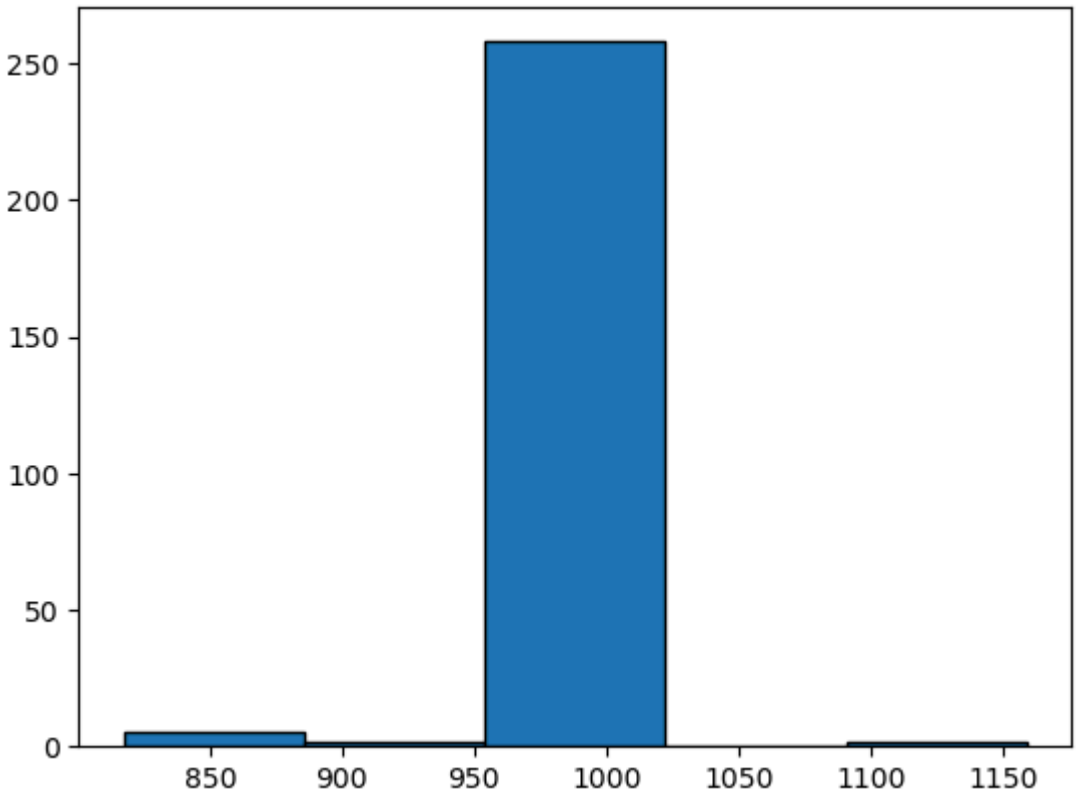
	Visibility (km)	count
0	8.5	39
1	6.5	35
2	7.5	34
3	7.0	29
4	5.5	28
5	8.0	26
6	6.0	20
7	9.0	14
8	5.0	8
9	10.0	4

In [463]:
```
1 I2['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[463]:

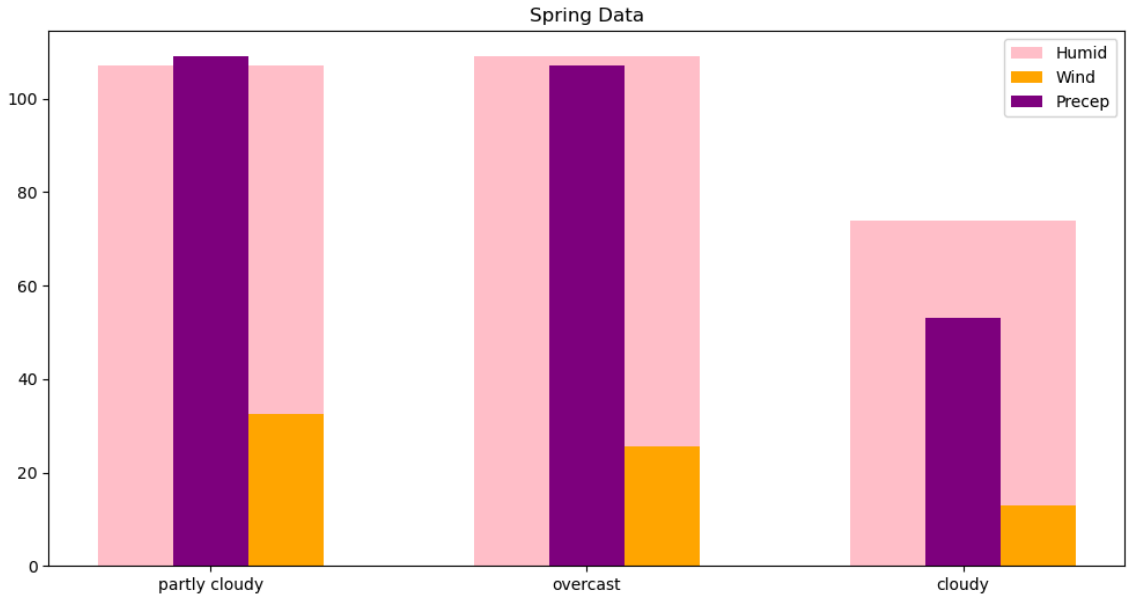
	Temperature C	count
0	16.222222	15
2	12.222222	13
1	-3.777778	13
3	-7.777778	12
4	9.222222	12
5	-1.777778	12
6	8.222222	12
7	11.222222	10
8	-2.777778	10
9	1.222222	10

In [464]:
```
1 plt.hist(I2['Atmospheric Pressure'],bins=5,edgecolor="black")
2 plt.show()
```



In [465]:

```
plt.figure(figsize=(12,6))
plt.bar(I2['Cloud Cover'],I2['Humidity'],color="pink",width=0.6,align='
plt.bar(I2['Cloud Cover'],I2['Wind Speed'],color="orange",width=0.3,ali
plt.bar(I2['Cloud Cover'],I2['Precipitation (%)'],color="purple",width=

plt.title("Spring Data")
plt.legend()
plt.show()
```



In [466]:

```

I3=I.loc[I['Weather Type']=="Rainy"]
I3
```

Out[466]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
26	72	7.5	62.0	overcast	1010.16	2	Summer	3.0	mou
48	91	9.5	82.0	overcast	1019.42	3	Summer	4.0	mou
83	62	5.5	99.0	overcast	1019.92	0	Summer	4.5	mou
113	60	12.5	42.0	overcast	822.66	2	Summer	17.5	mou
129	96	14.0	94.0	overcast	997.77	0	Summer	4.5	mou
...	...	...	...	...	...	...	...	...	
13134	89	10.5	89.0	overcast	994.68	2	Summer	4.5	mou
13153	87	17.5	57.0	partly cloudy	1007.44	0	Summer	4.5	mou
13180	62	17.0	83.0	partly cloudy	1013.09	3	Summer	3.0	mou
13184	62	7.5	14.0	overcast	1128.35	3	Summer	7.5	mou
13195	74	14.5	71.0	overcast	1003.15	1	Summer	1.0	mou

258 rows × 11 columns

In [467]:
```
1 I3['Cloud Cover'].value_counts()
```

Out[467]:
```
Cloud Cover
overcast         181
partly cloudy     67
cloudy            10
Name: count, dtype: int64
```

In [468]:
```
1 I3['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[468]:

	Visibility (km)	count
0	4.5	47
1	2.0	39
2	3.0	30
3	3.5	26
4	2.5	25
5	4.0	21
6	1.0	21
7	1.5	19
8	5.0	9
9	17.5	3
10	0.5	3
12	14.5	2
11	6.5	2
13	11.5	1
14	8.5	1
15	5.5	1
16	12.0	1
17	11.0	1
18	16.5	1
19	7.0	1
20	6.0	1
21	16.0	1
22	13.0	1
23	7.5	1

In [469]:
```
1 I3['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[469]:

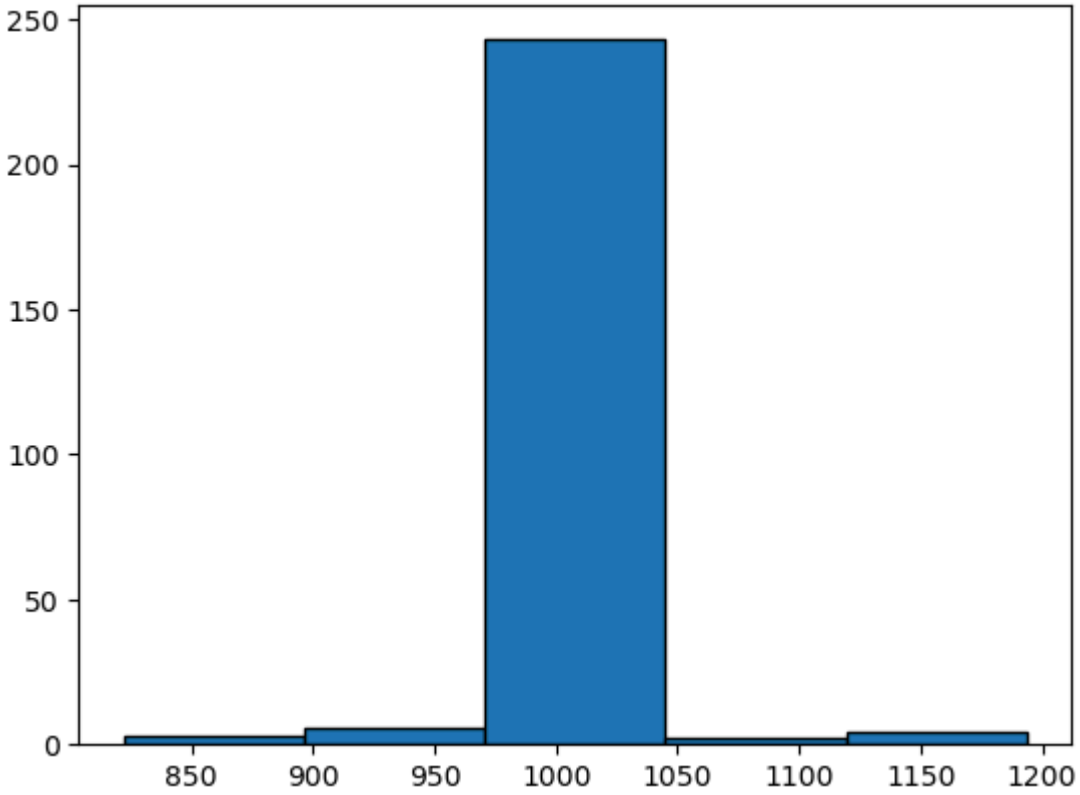
	UV Index	count
0	0	67
1	3	55
2	2	50
3	1	47
4	6	6
5	12	6
6	11	5
7	9	5
8	8	5
9	4	4
10	14	3
11	10	3
12	7	2

In [470]:
```
1 I3['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

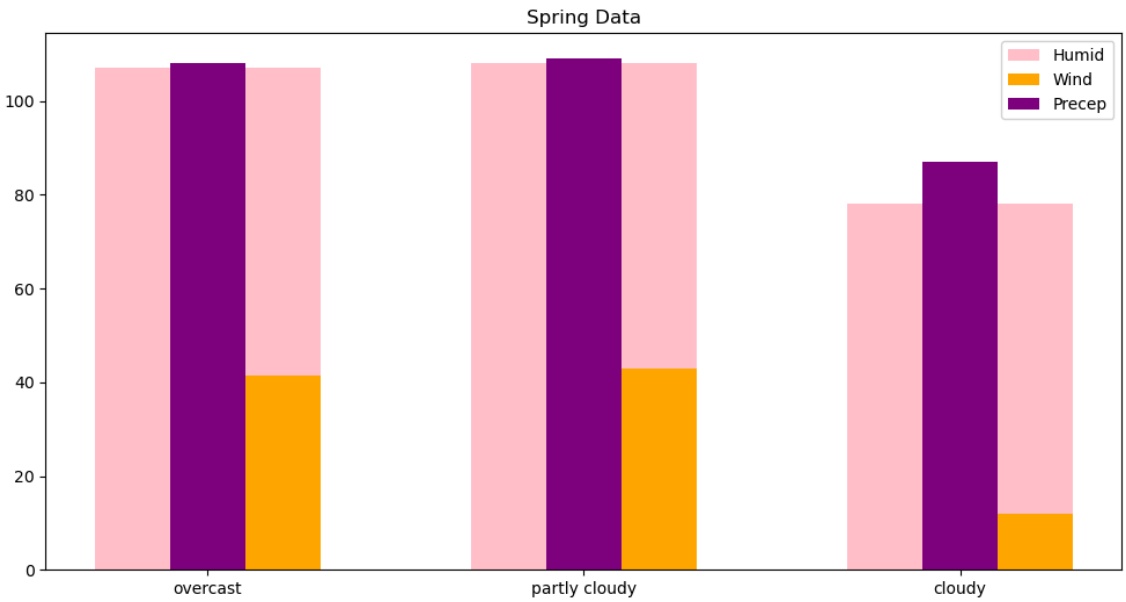
Out[470]:

	Temperature C	count
0	12.222222	16
1	14.222222	15
2	4.222222	14
3	5.222222	14
4	7.222222	12
5	-4.777778	11
6	1.222222	10
7	16.222222	10
8	15.222222	10
9	-6.777778	10

In [471]:
```
plt.hist(I3['Atmospheric Pressure'],bins=5,edgecolor="black")
plt.show()
```



In [472]:
```
plt.figure(figsize=(12,6))
plt.bar(I3['Cloud Cover'],I3['Humidity'],color="pink",width=0.6,align='
plt.bar(I3['Cloud Cover'],I3['Wind Speed'],color="orange",width=0.3,ali
plt.bar(I3['Cloud Cover'],I3['Precipitation (%)'],color="purple",width=

plt.title("Spring Data")
plt.legend()
plt.show()
```



In [473]:
```
I4=I.loc[I['Weather Type']=='Snowy']
I4
```

Out[473]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
686	48	7.5	34.0	overcast	1196.03	6	Summer	15.5	mou
1019	61	13.0	69.0	overcast	1004.98	0	Summer	11.0	mou
1211	22	14.0	51.0	partly cloudy	1050.12	2	Summer	1.0	mou
1475	77	8.5	21.0	overcast	887.74	5	Summer	14.0	mou
1769	40	3.0	29.0	partly cloudy	1035.69	9	Summer	8.0	mou
2448	42	8.0	83.0	partly cloudy	1069.74	7	Summer	0.5	mou
3461	69	7.5	84.0	cloudy	837.34	13	Summer	3.0	mou
3693	23	4.5	96.0	partly cloudy	854.45	1	Summer	9.0	mou
4594	63	13.0	35.0	partly cloudy	997.65	9	Summer	9.0	mou
6526	27	8.0	13.0	cloudy	804.00	14	Summer	19.5	mou
6893	64	9.0	41.0	cloudy	837.98	0	Summer	2.5	mou
8637	35	5.0	96.0	cloudy	988.10	13	Summer	10.0	mou
8696	54	10.0	99.0	partly cloudy	1080.07	0	Summer	1.0	mou
8765	53	5.5	30.0	partly cloudy	1126.78	7	Summer	16.0	mou
9239	68	4.0	66.0	cloudy	1066.11	11	Summer	11.0	mou
11206	32	11.5	55.0	overcast	1024.35	3	Summer	19.5	mou
11300	20	9.0	19.0	overcast	987.05	7	Summer	18.5	mou

In [109]:
```
J=B.loc[B['Season']=='Autumn']
J
```

Out[109]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
12	59	10.5	25.0	partly cloudy	1016.08	3	Autumn	5.5	mour
28	61	6.5	3.0	partly cloudy	1018.15	7	Autumn	5.5	mour
43	64	2.0	18.0	partly cloudy	1028.55	8	Autumn	8.0	mour
46	66	9.0	18.0	partly cloudy	1014.34	3	Autumn	6.0	mour
64	52	9.0	29.0	partly cloudy	1007.84	1	Autumn	8.0	mour
...	...	...	...	...	...	...	...	...	
13095	89	16.0	50.0	partly cloudy	991.07	2	Autumn	4.0	mour
13106	78	11.5	21.0	partly cloudy	1014.60	2	Autumn	5.0	mour
13128	69	7.5	14.0	clear	1018.30	6	Autumn	6.5	mour
13146	59	3.5	13.0	clear	1016.91	11	Autumn	5.5	mour
13199	38	0.0	92.0	overcast	1015.37	5	Autumn	10.0	mour

806 rows × 11 columns

In [474]:
```
J['Weather Type'].value_counts()
```

Out[474]:
```
Weather Type
Sunny     285
Rainy     252
Cloudy    244
Snowy      25
Name: count, dtype: int64
```

In [475]:
```
J1=J.loc[J['Weather Type']=="Sunny"]
J1
```

Out[475]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Locati
28	61	6.5	3.0	partly cloudy	1018.15	7	Autumn	5.5	mounta
43	64	2.0	18.0	partly cloudy	1028.55	8	Autumn	8.0	mounta
100	66	8.0	5.0	clear	1019.82	5	Autumn	5.5	mounta
108	39	5.0	16.0	clear	1021.61	5	Autumn	5.0	mounta
110	48	9.5	6.0	clear	1015.16	6	Autumn	8.5	mounta
...	...	...	...	...	...	...	...	...	
12947	57	8.0	2.0	clear	1029.39	8	Autumn	7.5	mounta
13025	100	7.0	109.0	clear	1018.30	12	Autumn	14.5	mounta
13052	41	1.5	10.0	clear	1019.55	11	Autumn	7.0	mounta
13128	69	7.5	14.0	clear	1018.30	6	Autumn	6.5	mounta
13146	59	3.5	13.0	clear	1016.91	11	Autumn	5.5	mounta

285 rows × 11 columns

In [476]:
```
J1['Cloud Cover'].value_counts()
```

Out[476]:
```
Cloud Cover
clear            187
partly cloudy     81
cloudy            11
overcast           6
Name: count, dtype: int64
```

In [477]:
```
J1['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[477]:

	Temperature C	count
0	19.222222	14
2	13.222222	13
1	7.222222	13
3	11.222222	12
4	9.222222	12
5	18.222222	11
6	20.222222	11
7	4.222222	11
8	12.222222	11
9	24.222222	11

In [478]:
```
1 J1['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[478]:

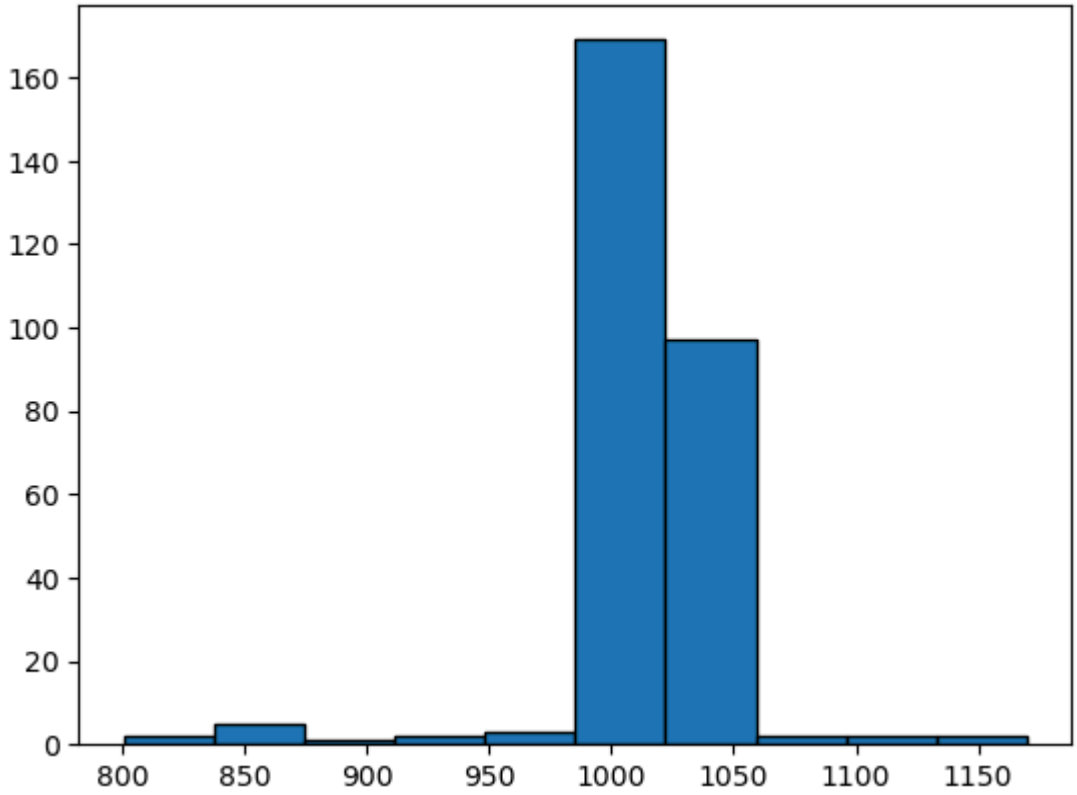
	UV Index	count
0	5	44
1	6	44
2	10	32
3	9	32
4	7	31
5	11	31
6	8	30
7	12	9
8	14	7
9	1	6

In [479]:
```
1 J1['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

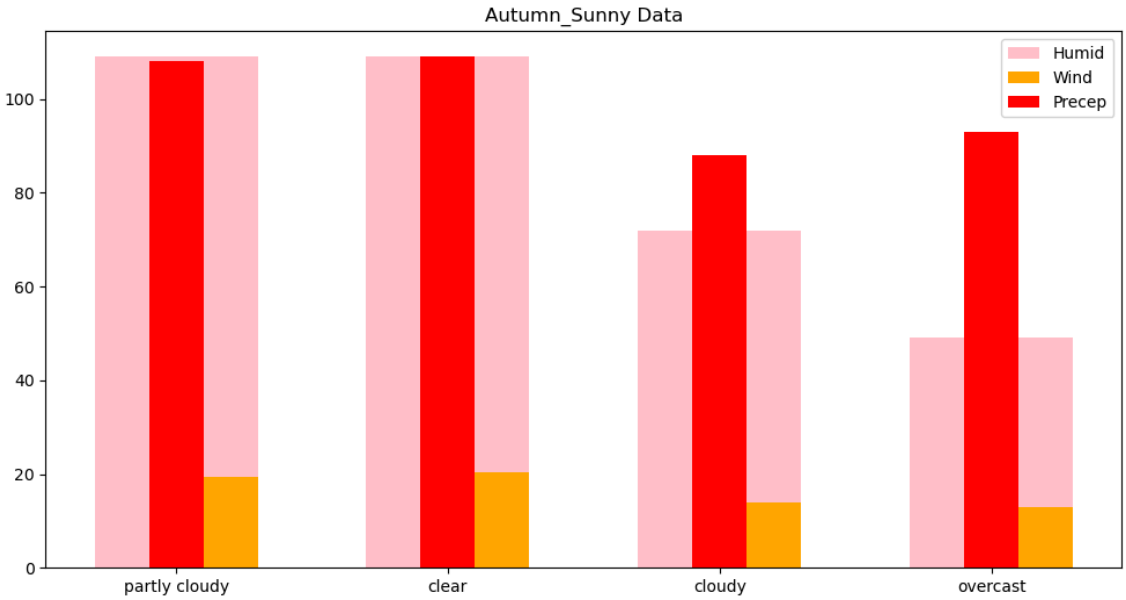
Out[479]:

	Visibility (km)	count
0	8.5	32
1	6.5	31
2	5.5	28
3	6.0	25
4	9.0	24
5	7.0	23
6	8.0	22
7	7.5	21
8	9.5	19
9	10.0	12

In [480]:
```
plt.hist(J1['Atmospheric Pressure'],bins=10,edgecolor="black")
plt.show()
```



In [481]:
```
plt.figure(figsize=(12,6))
plt.bar(J1['Cloud Cover'],J1['Humidity'],color="pink",width=0.6,align='
plt.bar(J1['Cloud Cover'],J1['Wind Speed'],color="orange",width=0.3,ali
plt.bar(J1['Cloud Cover'],J1['Precipitation (%)'],color="red",width=0.2

plt.title("Autumn_Sunny Data")
plt.legend()
plt.show()
```



In [482]:
```
J2=J.loc[J['Weather Type']=="Rainy"]
J2
```

Out[482]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
79	96	6.0	90.0	overcast	1006.67	0	Autumn	4.5	mour
107	80	15.5	59.0	overcast	1012.06	0	Autumn	3.5	mour
168	73	2.5	79.0	overcast	1034.10	3	Autumn	13.0	mour
170	88	19.0	98.0	overcast	998.80	2	Autumn	3.0	mour
196	99	16.0	58.0	overcast	995.85	2	Autumn	1.5	mour
...	...	...	...	...	...	...	...	...	
12945	65	15.0	81.0	overcast	1012.43	1	Autumn	5.0	mour
12949	83	19.0	95.0	partly cloudy	1002.21	2	Autumn	3.0	mour
13023	76	12.5	74.0	overcast	993.05	0	Autumn	4.0	mour
13095	89	16.0	50.0	partly cloudy	991.07	2	Autumn	4.0	mour
13199	38	0.0	92.0	overcast	1015.37	5	Autumn	10.0	mour

252 rows × 11 columns

In [483]:
```
J2['Cloud Cover'].value_counts()
```

Out[483]:
```
Cloud Cover
overcast         171
partly cloudy     75
cloudy             6
Name: count, dtype: int64
```

In [484]:
```
J2['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[484]:

	UV Index	count
0	2	57
1	1	54
2	3	52
3	0	45
4	5	7
5	7	6
6	9	5
7	4	5
8	14	4
9	8	4

In [485]:
```
1 J2['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[485]:

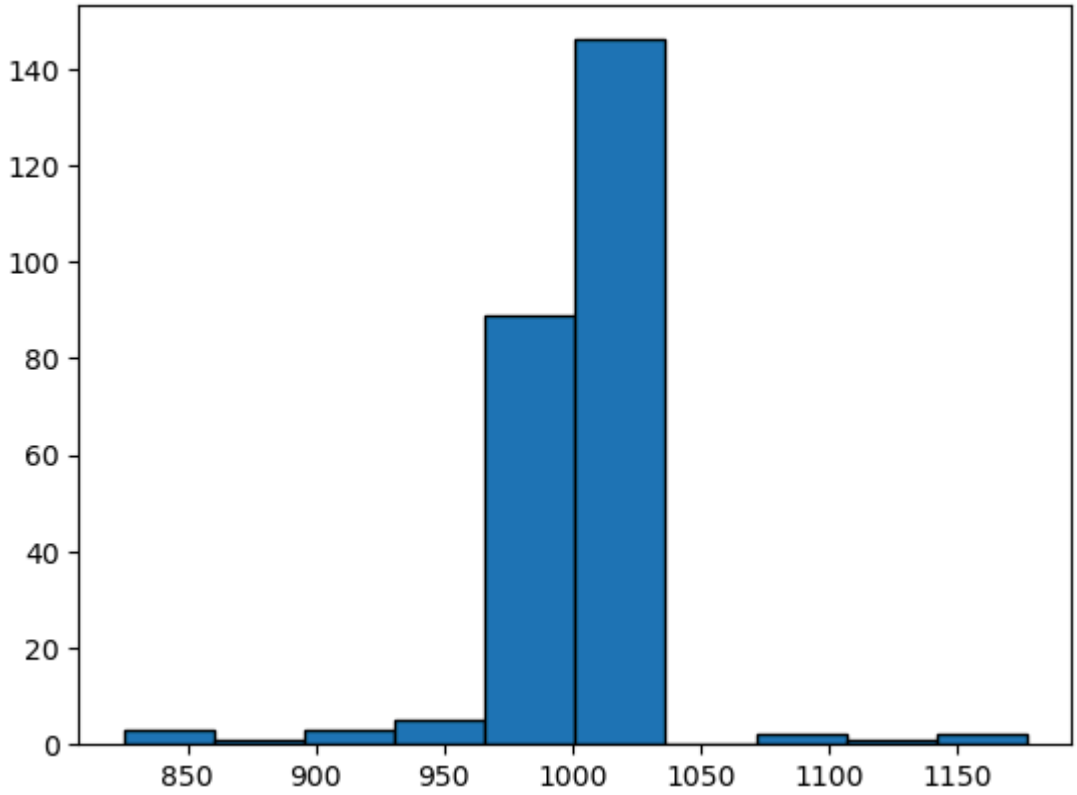
	Visibility (km)	count
0	4.0	34
1	3.0	32
2	1.5	30
3	3.5	27
4	2.5	25
5	2.0	21
6	4.5	20
7	1.0	19
8	5.0	18
9	0.5	4

In [486]:
```
1 J2['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

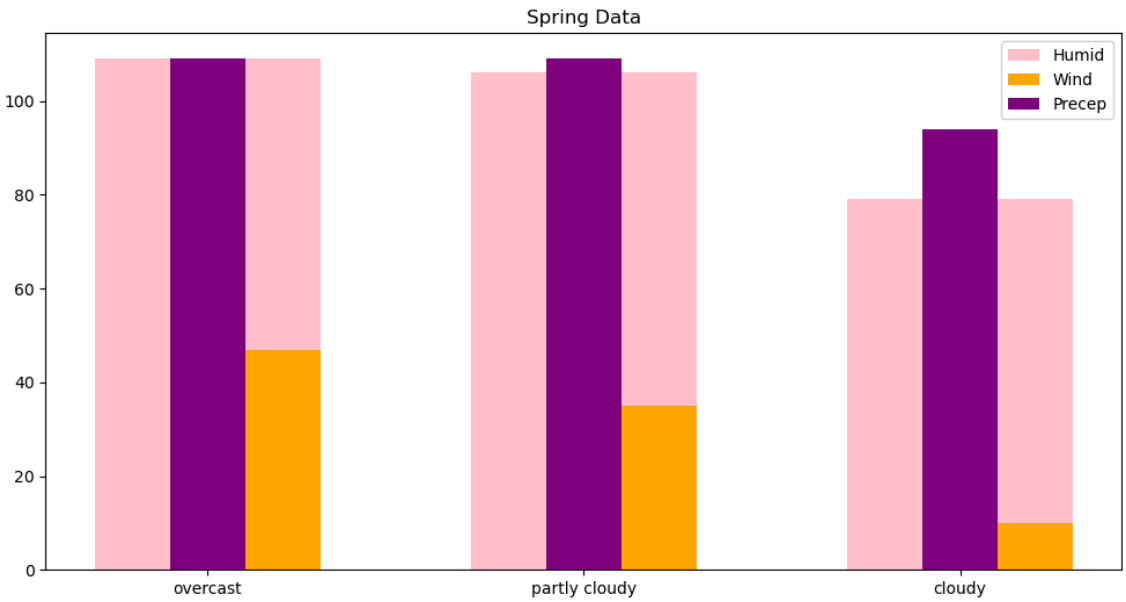
Out[486]:

	Temperature C	count
0	-0.777778	16
2	-2.777778	16
1	13.222222	16
3	-3.777778	15
4	15.222222	14
5	6.222222	10
6	-6.777778	10
7	4.222222	9
8	-7.777778	9
9	-4.777778	9

In [487]:
```
plt.hist(J2['Atmospheric Pressure'],bins=10,edgecolor="black")
plt.show()
```



In [488]:
```
plt.figure(figsize=(12,6))
plt.bar(J2['Cloud Cover'],J2['Humidity'],color="pink",width=0.6,align='
plt.bar(J2['Cloud Cover'],J2['Wind Speed'],color="orange",width=0.3,ali
plt.bar(J2['Cloud Cover'],J2['Precipitation (%)'],color="purple",width=

plt.title("Spring Data")
plt.legend()
plt.show()
```



In [489]:
```
J3=J.loc[J['Weather Type']=='Cloudy']
J3
```

Out[489]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
12	59	10.5	25.0	partly cloudy	1016.08	3	Autumn	5.5	mour
46	66	9.0	18.0	partly cloudy	1014.34	3	Autumn	6.0	mour
64	52	9.0	29.0	partly cloudy	1007.84	1	Autumn	8.0	mour
76	66	13.5	29.0	partly cloudy	1018.98	3	Autumn	8.5	mour
116	69	1.0	19.0	partly cloudy	1011.50	2	Autumn	6.0	mour
...	...	...	...	...	...	...	...	...	
12919	70	4.5	47.0	partly cloudy	1003.79	1	Autumn	5.5	mour
12953	58	9.5	43.0	overcast	1006.37	2	Autumn	7.5	mour
12971	55	9.0	32.0	overcast	1004.62	2	Autumn	6.5	mour
12977	59	1.0	41.0	cloudy	1099.43	2	Autumn	10.5	mour
13106	78	11.5	21.0	partly cloudy	1014.60	2	Autumn	5.0	mour

244 rows × 11 columns

In [490]:
```
J3['Cloud Cover'].value_counts()
```

Out[490]:
```
Cloud Cover
partly cloudy    144
overcast          90
cloudy            10
Name: count, dtype: int64
```

In [491]:
```
1 J3['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[491]:

	UV Index	count
0	3	52
1	1	52
2	2	48
3	4	47
4	11	8
5	6	5
6	14	5
7	0	5
8	8	4
9	7	4

In [492]:
```
1 J3['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[492]:

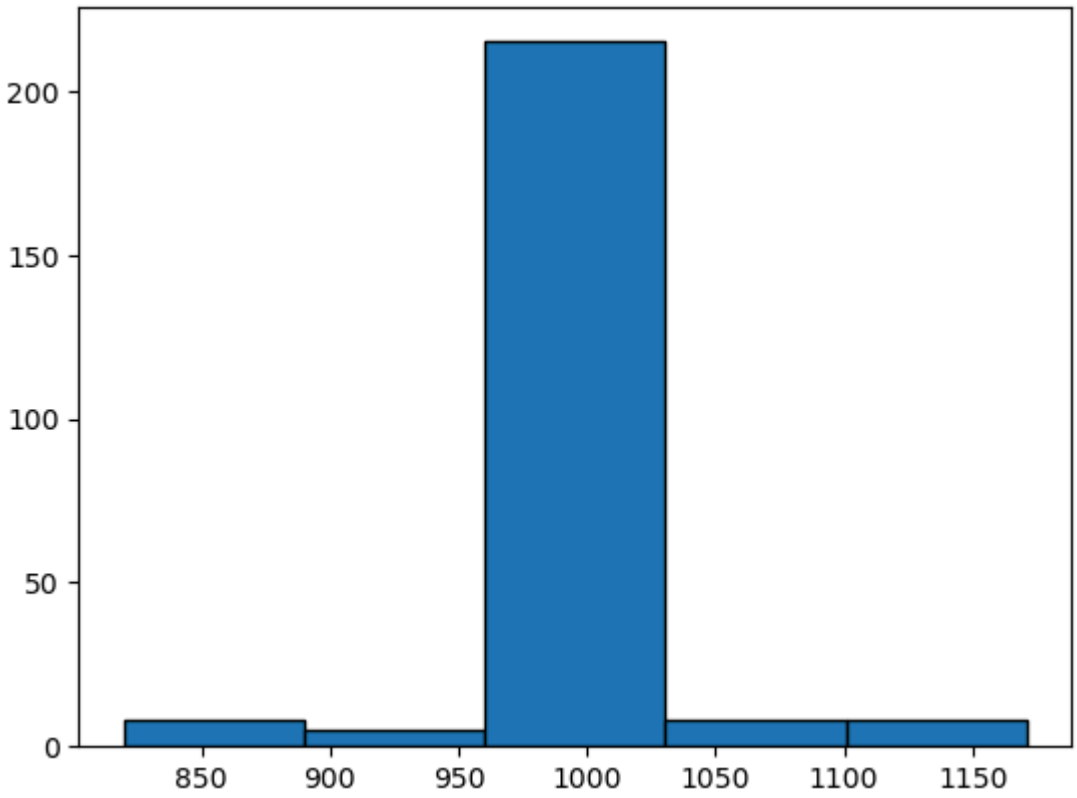
	Visibility (km)	count
0	6.5	34
2	6.0	30
1	5.5	30
3	8.0	28
4	7.5	28
5	7.0	25
6	8.5	20
7	5.0	12
8	9.0	8
13	9.5	2

In [493]:
```
1 J3['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[493]:

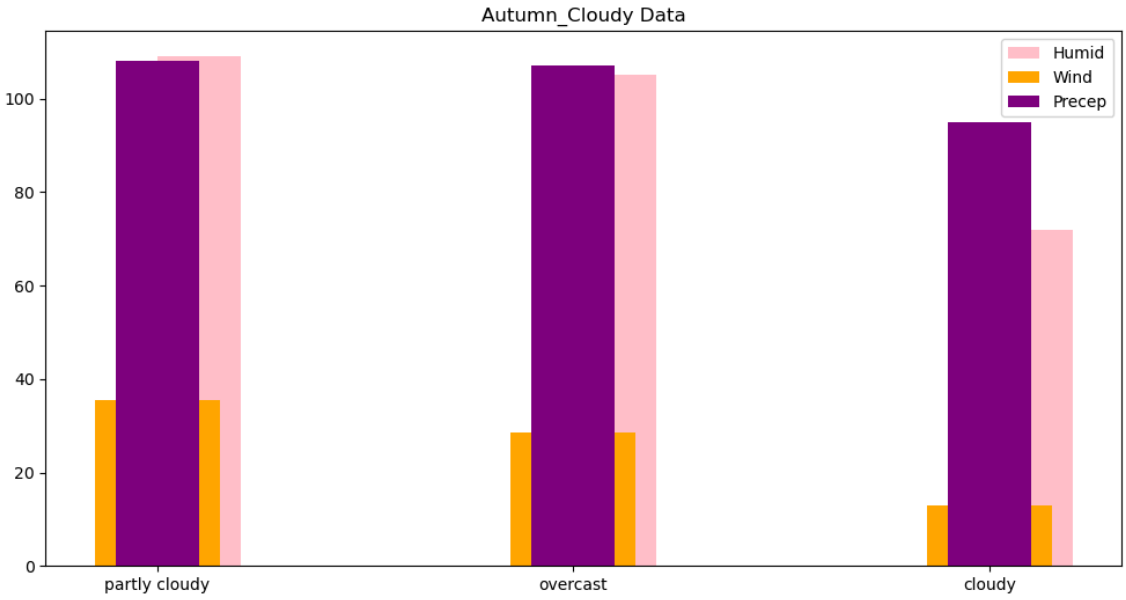
	Temperature C	count
0	15.222222	15
1	3.222222	13
2	12.222222	13
3	-6.777778	13
4	2.222222	12
5	-3.777778	10
6	7.222222	10
7	-2.777778	9
8	8.222222	9
9	10.222222	9

In [494]:
```
1 plt.hist(J3['Atmospheric Pressure'],bins=5,edgecolor="black")
2 plt.show()
```



In [495]:

```
plt.figure(figsize=(12,6))
plt.bar(J3['Cloud Cover'],J3['Humidity'],color="pink",width=0.2,align='
plt.bar(J3['Cloud Cover'],J3['Wind Speed'],color="orange",width=0.3,ali
plt.bar(J3['Cloud Cover'],J3['Precipitation (%)'],color="purple",width=

plt.title("Autumn_Cloudy Data")
plt.legend()
plt.show()
```



In [496]:
```
J4=J.loc[J['Weather Type']=="Snowy"]
J4
```

Out[496]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
190	21	0.0	14.0	cloudy	960.81	2	Autumn	6.0	mour
300	79	12.0	88.0	partly cloudy	1140.24	4	Autumn	17.0	mour
931	59	4.0	11.0	overcast	1022.41	1	Autumn	19.0	mour
1638	44	5.5	83.0	overcast	925.06	12	Autumn	8.0	mour
2118	44	5.0	29.0	cloudy	875.87	13	Autumn	16.5	mour
2180	48	13.5	60.0	partly cloudy	1074.00	12	Autumn	19.0	mour
2414	35	10.0	34.0	cloudy	1100.82	12	Autumn	0.5	mour
2715	36	13.5	12.0	overcast	805.92	5	Autumn	5.0	mour
3211	62	11.5	93.0	partly cloudy	921.50	14	Autumn	18.5	mour
3874	65	1.0	61.0	cloudy	1135.33	1	Autumn	13.5	mour
5095	75	6.5	92.0	overcast	1026.83	1	Autumn	16.5	mour
5789	76	13.5	21.0	partly cloudy	1138.17	1	Autumn	2.0	mour
6340	38	11.0	20.0	overcast	892.79	7	Autumn	1.5	mour
6553	52	3.5	87.0	partly cloudy	922.51	14	Autumn	16.0	mour
6754	58	2.0	35.0	cloudy	1180.39	11	Autumn	2.0	mour
7038	23	3.0	73.0	overcast	999.13	0	Autumn	7.0	mour
8485	45	11.0	47.0	overcast	933.99	4	Autumn	19.5	mour
9338	35	4.5	85.0	overcast	1134.21	13	Autumn	9.0	mour
10004	29	9.0	58.0	overcast	845.51	1	Autumn	16.5	mour
10239	50	3.0	23.0	cloudy	816.14	8	Autumn	19.5	mour
10334	21	10.0	84.0	overcast	1064.61	0	Autumn	1.5	mour
10596	40	6.0	13.0	cloudy	803.48	7	Autumn	15.5	mour
10627	47	7.5	24.0	partly cloudy	1016.26	4	Autumn	3.5	mour
11747	47	14.5	71.0	overcast	819.46	6	Autumn	15.0	mour
12281	51	3.0	59.0	overcast	932.02	8	Autumn	5.5	mour

In [497]:
```
# FOR COASTAL AREA DATA
```

In [102]:
```
C=X.loc[X['Location']=='coastal']
C
```

Out[102]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
3	83	1.5	82.0	clear	1026.25	7	Spring	1.0	co
9	74	8.5	107.0	clear	1012.13	8	Winter	7.5	co
14	21	3.5	8.0	clear	1018.88	8	Winter	5.5	co
17	51	0.5	27.0	overcast	1009.18	3	Autumn	5.5	co
27	57	4.0	12.0	partly cloudy	1020.73	11	Winter	9.5	co
...	...	...	...	...	...	...	...	...	
13185	106	23.5	104.0	partly cloudy	998.71	14	Summer	3.0	co
13186	66	5.5	39.0	overcast	1018.86	4	Spring	8.0	co
13194	62	13.0	17.0	overcast	1002.81	2	Spring	5.0	co
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	co
13197	77	5.5	28.0	overcast	1012.69	3	Autumn	9.0	co

3571 rows × 11 columns

In [498]:
```
C['Season'].value_counts()
```

Out[498]:
```
Season
Winter    930
Autumn    900
Spring    888
Summer    853
Name: count, dtype: int64
```

In [499]:
```
C['Weather Type'].value_counts()
```

Out[499]:
```
Weather Type
Rainy     1216
Sunny     1129
Cloudy    1106
Snowy      120
Name: count, dtype: int64
```

In [500]:
```
1 C['Atmospheric Pressure'].value_counts()
```

Out[500]:
```
Atmospheric Pressure
1010.50    7
1017.97    6
1004.04    6
1019.24    6
1010.23    6
          ..
1011.31    1
999.84     1
1081.43    1
1012.83    1
1067.23    1
Name: count, Length: 2455, dtype: int64
```

In [501]:
```
1 K=C.loc[C['Season']=='Winter']
2 K
```

Out[501]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
9	74	8.5	107.0	clear	1012.13	8	Winter	7.5	coa
14	21	3.5	8.0	clear	1018.88	8	Winter	5.5	coa
27	57	4.0	12.0	partly cloudy	1020.73	11	Winter	9.5	coa
57	41	0.0	4.0	clear	1018.56	7	Winter	9.5	coa
85	35	2.5	63.0	partly cloudy	1160.48	2	Winter	0.0	coa
...	...	...	...	...	...	...	...	...	
13126	61	2.0	2.0	clear	1013.04	7	Winter	7.5	coa
13156	64	15.0	23.0	overcast	1016.81	3	Winter	6.5	coa
13161	78	3.5	61.0	cloudy	898.87	10	Winter	1.5	coa
13170	60	17.0	86.0	overcast	992.76	1	Winter	2.0	coa
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	coa

930 rows × 11 columns

In [502]:
```
1 K['Cloud Cover'].value_counts()
```

Out[502]:
```
Cloud Cover
partly cloudy    377
overcast         333
clear            175
cloudy            45
Name: count, dtype: int64
```

In [503]:
```
K['Weather Type'].value_counts()
```

Out[503]:
```
Weather Type
Rainy     327
Cloudy    286
Sunny     281
Snowy      36
Name: count, dtype: int64
```

In [504]:
```
K1=K.loc[K['Weather Type']=='Rainy']
K1
```

Out[504]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
93	44	9.0	11.0	partly cloudy	853.01	14	Winter	3.5	coa
128	92	19.5	81.0	overcast	1017.76	0	Winter	4.5	coa
178	90	21.5	103.0	overcast	1002.15	13	Winter	4.0	coa
185	64	19.0	96.0	partly cloudy	1006.43	0	Winter	4.5	coa
191	97	12.5	62.0	overcast	1013.38	2	Winter	3.0	coa
...	...	...	...	...	...	...	...	...	
13060	82	8.0	60.0	partly cloudy	992.83	1	Winter	4.5	coa
13074	70	10.5	61.0	partly cloudy	1015.92	0	Winter	1.0	coa
13101	65	11.0	82.0	overcast	1014.96	3	Winter	4.0	coa
13161	78	3.5	61.0	cloudy	898.87	10	Winter	1.5	coa
13170	60	17.0	86.0	overcast	992.76	1	Winter	2.0	coa

327 rows × 11 columns

In [505]:
```
K1["Cloud Cover"].value_counts()
```

Out[505]:
```
Cloud Cover
overcast         199
partly cloudy    118
cloudy            10
Name: count, dtype: int64
```

In [506]:
```
1 K1['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[506]:

	UV Index	count
0	0	78
1	2	69
2	3	67
3	1	63
4	14	7
5	4	7
6	6	5
7	10	5
8	8	5
9	7	5

In [507]:
```
1 K1['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[507]:

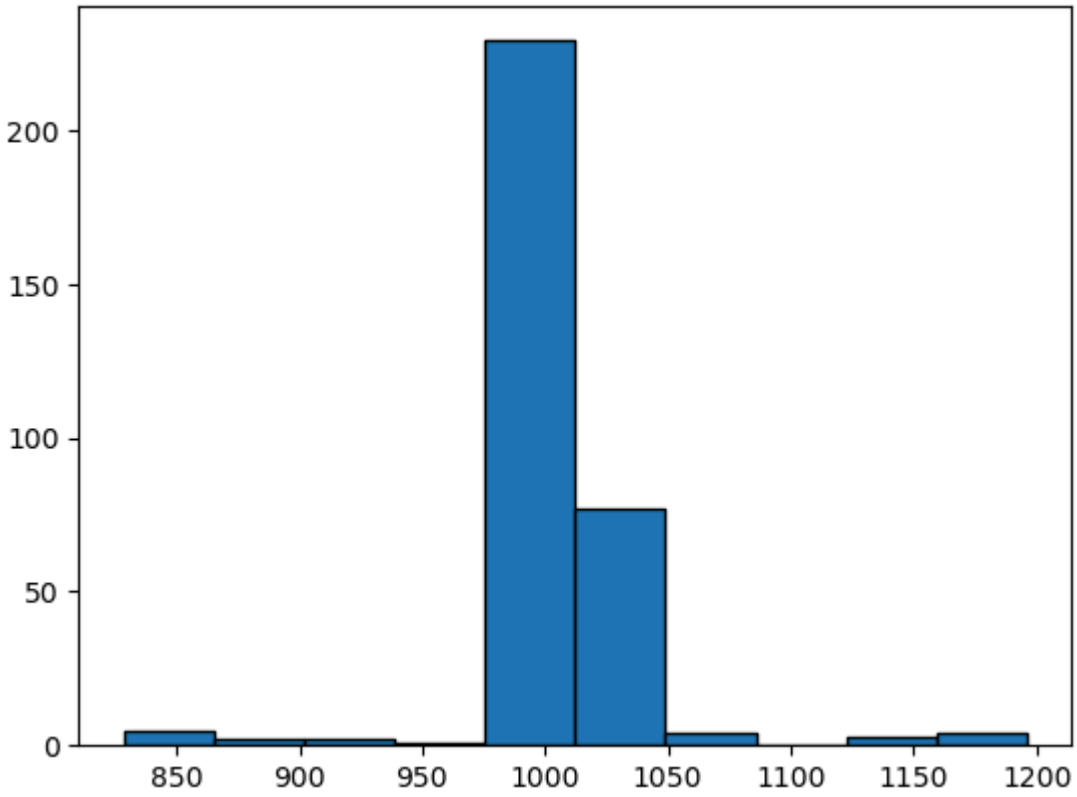
	Temperature C	count
0	6.222222	17
2	-5.777778	16
1	3.222222	16
3	0.222222	15
4	10.222222	14
5	-7.777778	13
6	-0.777778	13
7	-3.777778	13
8	16.222222	12
9	13.222222	12

In [508]:
```
1 K1['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[508]:

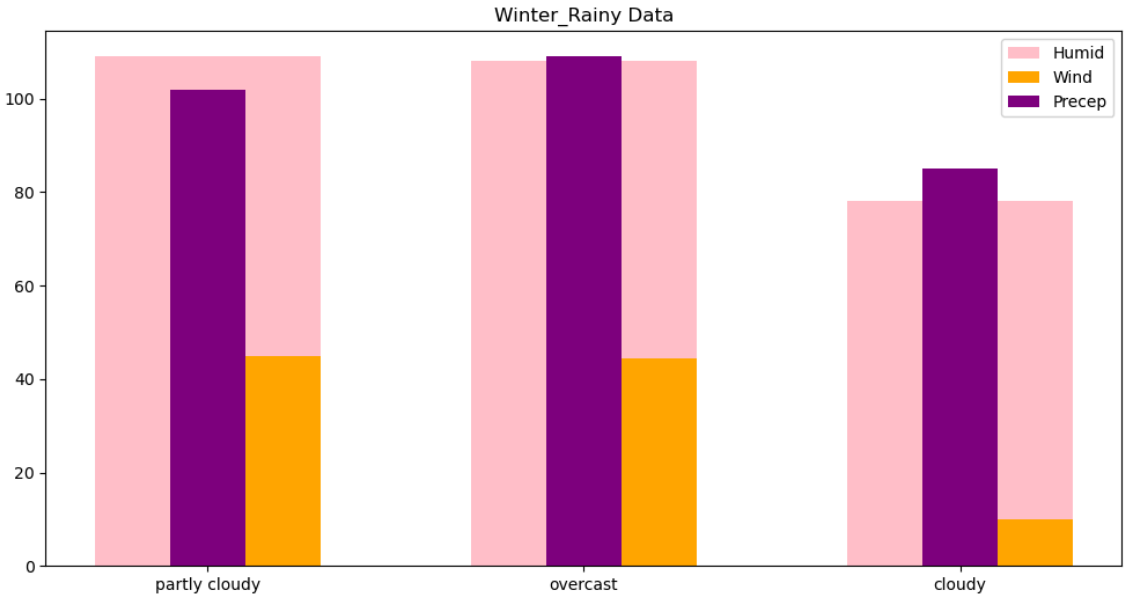
	Visibility (km)	count
0	2.0	48
1	3.5	43
2	1.5	42
3	4.5	34
4	4.0	31
5	2.5	31
6	1.0	26
7	3.0	24
8	5.0	23
9	0.5	5

In [509]:
```
1 plt.hist(K1['Atmospheric Pressure'],bins=10,edgecolor="black")
2 plt.show()
```



In [510]:

```
plt.figure(figsize=(12,6))
plt.bar(K1['Cloud Cover'],K1['Humidity'],color="pink",width=0.6,align='
plt.bar(K1['Cloud Cover'],K1['Wind Speed'],color="orange",width=0.3,ali
plt.bar(K1['Cloud Cover'],K1['Precipitation (%)'],color="purple",width=

plt.title("Winter_Rainy Data")
plt.legend()
plt.show()
```



In [511]:

```
K2=K.loc[K['Weather Type']=="Cloudy"]
K2
```

Out[511]:

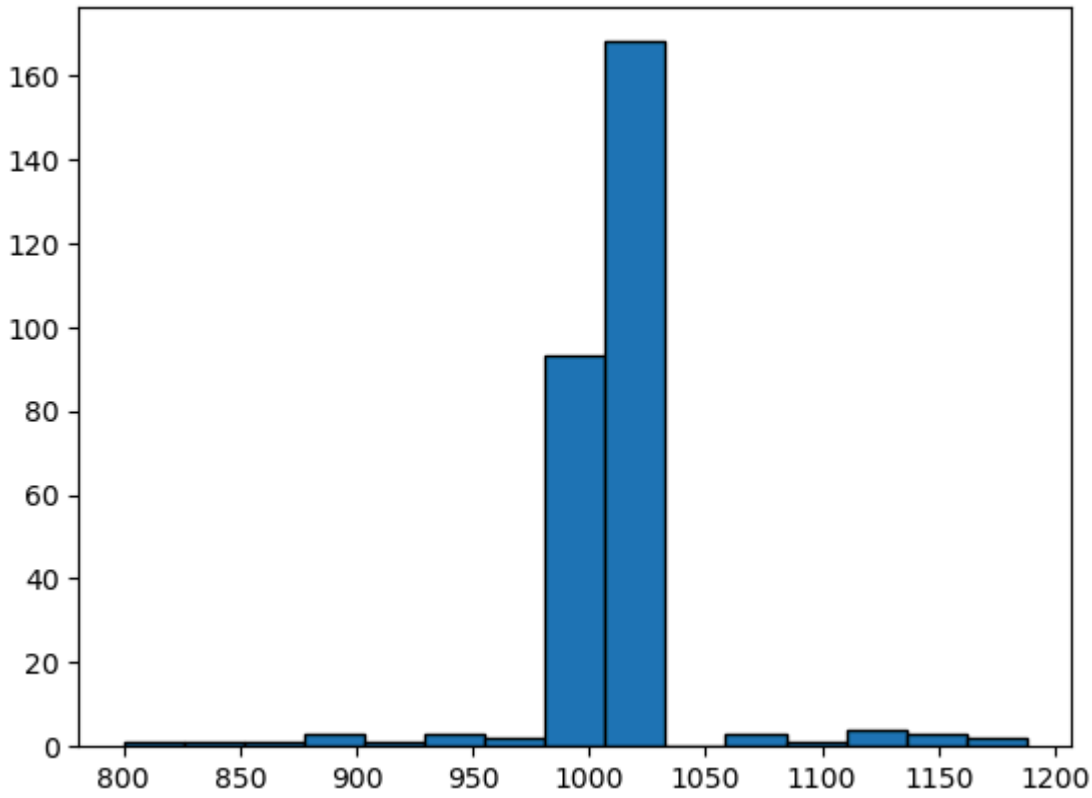
	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
176	60	9.5	18.0	partly cloudy	931.79	7	Winter	5.5	coa
205	56	8.5	12.0	partly cloudy	1013.53	1	Winter	7.5	coa
206	64	4.0	61.0	cloudy	1130.72	14	Winter	0.0	coa
229	89	12.0	85.0	partly cloudy	1017.54	11	Winter	2.0	coa
233	68	11.5	11.0	overcast	1012.85	2	Winter	6.5	coa
...	...	...	...	...	...	...	...	...	
13024	76	8.5	40.0	partly cloudy	1011.80	3	Winter	6.0	coa
13030	60	7.0	39.0	partly cloudy	1016.08	3	Winter	5.5	coa
13068	63	15.0	15.0	overcast	1012.77	2	Winter	7.5	coa
13121	86	6.0	89.0	overcast	1013.47	13	Winter	1.5	coa
13156	64	15.0	23.0	overcast	1016.81	3	Winter	6.5	coa

286 rows × 11 columns

In [512]:
```
1 K2['Atmospheric Pressure'].value_counts().head(10)
```

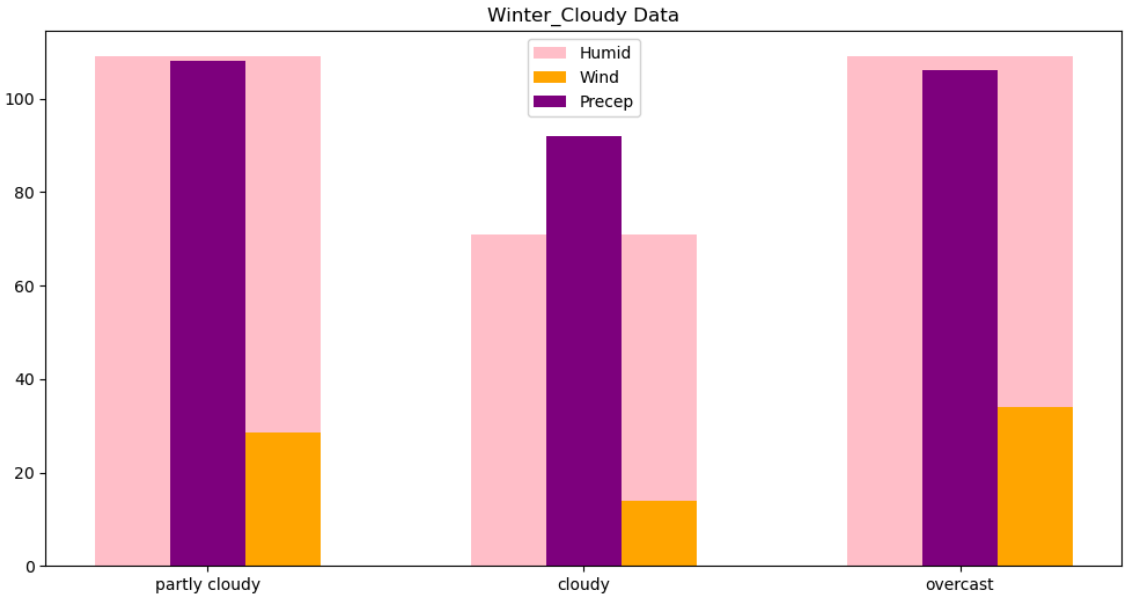
Out[512]:
```
Atmospheric Pressure
1019.47    3
1003.80    2
1012.14    2
1018.34    2
1006.67    2
1010.43    2
1004.17    2
1000.41    2
1019.24    2
1016.79    2
Name: count, dtype: int64
```

In [513]:
```
1 plt.hist(K2['Atmospheric Pressure'],bins=15,edgecolor="black")
2 plt.show()
```



In [514]:

```
plt.figure(figsize=(12,6))
plt.bar(K2['Cloud Cover'],K2['Humidity'],color="pink",width=0.6,align='
plt.bar(K2['Cloud Cover'],K2['Wind Speed'],color="orange",width=0.3,ali
plt.bar(K2['Cloud Cover'],K2['Precipitation (%)'],color="purple",width=

plt.title("Winter_Cloudy Data")
plt.legend()
plt.show()
```



In [515]:

```
K2['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[515]:

	Temperature C	count
0	-2.777778	20
1	9.222222	15
2	16.222222	14
3	14.222222	13
4	8.222222	12
5	2.222222	11
6	4.222222	11
7	-1.777778	11
8	12.222222	11
9	-5.777778	11

In [516]:
```
1 K2['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[516]:

	UV Index	count
0	1	63
1	2	57
2	4	52
3	3	51
4	12	13
5	10	10
6	8	6
7	9	6
8	11	5
9	6	5

In [517]:
```
1 K2['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[517]:

	Visibility (km)	count
0	5.5	37
1	6.5	34
2	6.0	31
3	8.5	30
4	7.5	28
5	7.0	28
6	8.0	25
7	5.0	18
8	9.0	14
9	4.0	5

In [518]:
```
K3=K.loc[K['Weather Type']=="Sunny"]
K3
```

Out[518]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
9	74	8.5	107.0	clear	1012.13	8	Winter	7.5	coa
14	21	3.5	8.0	clear	1018.88	8	Winter	5.5	coa
27	57	4.0	12.0	partly cloudy	1020.73	11	Winter	9.5	coa
57	41	0.0	4.0	clear	1018.56	7	Winter	9.5	coa
85	35	2.5	63.0	partly cloudy	1160.48	2	Winter	0.0	coa
...	...	...	...	...	...	...	...	...	
12875	50	4.5	17.0	clear	1012.65	7	Winter	6.5	coa
12916	33	7.5	42.0	overcast	908.17	4	Winter	17.5	coa
13016	28	9.0	14.0	clear	1016.96	5	Winter	10.0	coa
13055	60	5.5	9.0	clear	1026.37	9	Winter	8.5	coa
13126	61	2.0	2.0	clear	1013.04	7	Winter	7.5	coa

281 rows × 11 columns

In [519]:
```
K3['Cloud Cover'].value_counts()
```

Out[519]:
```
Cloud Cover
clear            175
partly cloudy     85
overcast          11
cloudy            10
Name: count, dtype: int64
```

In [520]:
```
K3['UV Index'].mean()
```

Out[520]:
```
7.5088967971530245
```

In [521]:
```
K3['Visibility (km)'].value_counts().reset_index().sort_values(by="cour
```

Out[521]:

	Visibility (km)	count
0	5.5	31
1	6.5	29
2	8.0	28
3	7.5	27
4	9.0	22
5	6.0	22
6	7.0	20
7	8.5	20
8	9.5	19
9	5.0	13

In [522]:
```
K3['UV Index'].value_counts().reset_index().sort_values(by="count",asce
```

Out[522]:

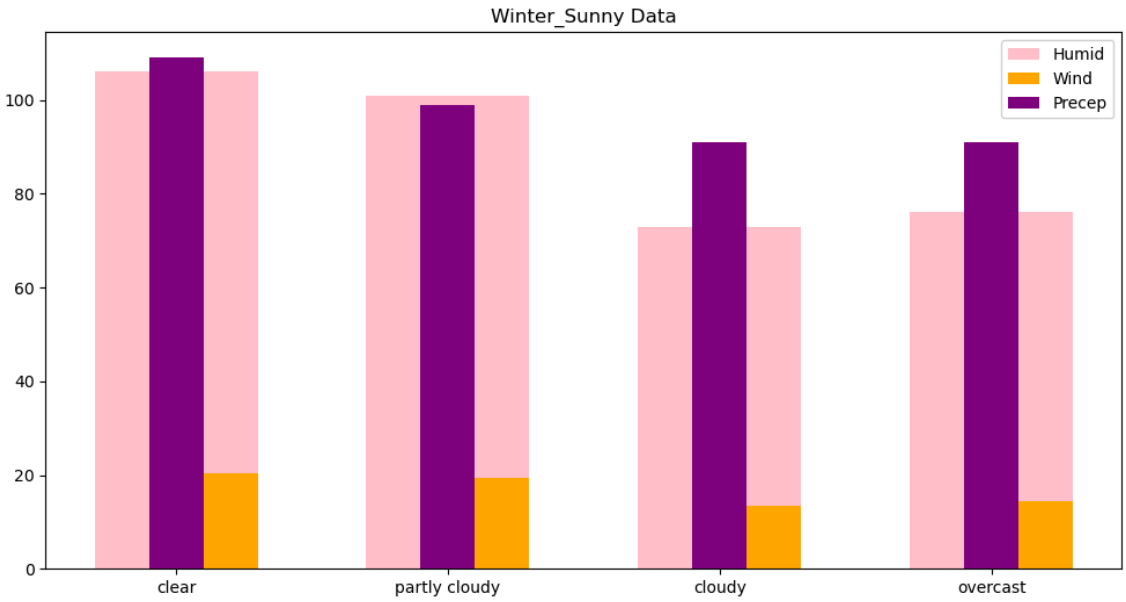
	UV Index	count
0	6	45
1	5	40
2	9	36
3	11	35
4	10	32
5	8	27
6	7	23
7	0	9
8	2	6
9	12	6

In [523]:
```
1 K3['Temperature C'].value_counts().reset_index().sort_values(by="count'
```

Out[523]:

	Temperature C	count
0	5.222222	14
2	18.222222	14
1	26.222222	14
3	11.222222	12
4	10.222222	11
5	19.222222	11
6	16.222222	11
7	22.222222	11
8	17.222222	10
9	14.222222	10

In [524]:
```
1 plt.figure(figsize=(12,6))
2 plt.bar(K3['Cloud Cover'],K3['Humidity'],color="pink",width=0.6,align='
3 plt.bar(K3['Cloud Cover'],K3['Wind Speed'],color="orange",width=0.3,ali
4 plt.bar(K3['Cloud Cover'],K3['Precipitation (%)'],color="purple",width=
5
6 plt.title("Winter_Sunny Data")
7 plt.legend()
8 plt.show()
```



In [525]:
```
1 K3['Atmospheric Pressure'].value_counts().head(6)
```

Out[525]:
```
Atmospheric Pressure
1023.32    3
1029.12    2
1010.03    2
1014.72    2
1015.29    2
1014.78    2
Name: count, dtype: int64
```

In [526]:

```
K4=K.loc[K['Weather Type']=='Snowy']
K4
```

Out[526]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
376	58	8.5	40.0	cloudy	1101.70	5	Winter	18.0	coa
862	21	6.5	17.0	partly cloudy	1054.24	7	Winter	7.5	coa
1193	78	3.5	85.0	overcast	1187.42	0	Winter	5.0	coa
1337	23	9.5	58.0	partly cloudy	997.17	0	Winter	3.0	coa
1528	39	5.5	55.0	partly cloudy	1073.46	6	Winter	17.0	coa
2714	34	2.0	34.0	cloudy	1061.53	11	Winter	17.0	coa
2891	37	15.0	60.0	overcast	1033.20	1	Winter	8.0	coa
3261	69	3.5	26.0	overcast	930.59	2	Winter	2.0	coa
4129	65	0.5	82.0	cloudy	1051.73	13	Winter	3.5	coa
4318	30	10.5	39.0	cloudy	1080.38	14	Winter	2.0	coa
4479	68	8.0	38.0	overcast	1112.41	14	Winter	16.0	coa
4726	40	4.0	27.0	overcast	810.06	3	Winter	4.0	coa
4744	27	12.0	40.0	overcast	1041.93	12	Winter	17.5	coa
5434	54	2.0	47.0	partly cloudy	830.65	1	Winter	8.5	coa
5560	25	7.5	78.0	cloudy	927.22	7	Winter	15.0	coa
5653	26	1.5	46.0	partly cloudy	1156.95	7	Winter	1.5	coa
5922	37	12.0	53.0	cloudy	865.93	13	Winter	20.0	coa
6082	33	1.0	76.0	cloudy	1084.82	8	Winter	11.0	coa
6225	67	8.0	94.0	overcast	803.08	12	Winter	5.5	coa
6605	33	2.5	86.0	cloudy	1146.40	3	Winter	16.5	coa
6752	69	14.0	52.0	cloudy	1094.52	4	Winter	9.0	coa
7081	70	8.5	28.0	cloudy	1130.56	10	Winter	14.5	coa
7679	20	3.0	93.0	partly cloudy	904.86	3	Winter	7.5	coa
7783	35	14.5	54.0	overcast	962.88	6	Winter	0.5	coa
7889	55	11.5	54.0	overcast	1174.96	14	Winter	11.5	coa
7907	56	8.5	59.0	overcast	879.52	11	Winter	9.0	coa
8083	49	3.0	57.0	cloudy	932.93	3	Winter	12.5	coa
8318	49	6.5	22.0	overcast	1091.99	6	Winter	15.0	coa
9195	25	9.5	64.0	partly cloudy	1022.32	3	Winter	12.0	coa
9789	72	11.5	11.0	cloudy	1112.53	14	Winter	5.5	coa
9972	72	3.0	12.0	cloudy	949.13	10	Winter	19.5	coa
10000	38	10.0	98.0	overcast	972.02	12	Winter	20.0	coa
10042	55	4.0	72.0	cloudy	1123.67	8	Winter	4.5	coa
10799	34	4.0	89.0	partly cloudy	933.62	8	Winter	5.5	coa

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
13061	42	3.0	57.0	cloudy	860.95	6	Winter	1.0	coa
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	coa

In [527]:
```
L=C.loc[C['Season']=='Spring']
L
```

Out[527]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
3	83	1.5	82.0	clear	1026.25	7	Spring	1.0	coa
30	85	7.0	97.0	overcast	991.07	1	Spring	2.0	coa
59	84	18.5	54.0	overcast	996.12	0	Spring	3.0	coa
60	64	14.0	18.0	overcast	1011.96	1	Spring	8.5	coa
105	79	4.5	43.0	overcast	1007.94	1	Spring	7.0	coa
...	...	...	...	...	...	...	...	...	
13167	73	5.0	31.0	partly cloudy	840.06	9	Spring	10.5	coa
13178	50	11.5	82.0	cloudy	822.40	8	Spring	15.5	coa
13183	26	6.5	10.0	clear	1016.53	10	Spring	7.5	coa
13186	66	5.5	39.0	overcast	1018.86	4	Spring	8.0	coa
13194	62	13.0	17.0	overcast	1002.81	2	Spring	5.0	coa

888 rows × 11 columns

In [528]:
```
L['Weather Type'].value_counts()
```

Out[528]:
```
Weather Type
Rainy     301
Cloudy    287
Sunny     270
Snowy      30
Name: count, dtype: int64
```

In [529]:
```
L['Cloud Cover'].value_counts()
```

Out[529]:
```
Cloud Cover
overcast         350
partly cloudy    326
clear            173
cloudy            39
Name: count, dtype: int64
```

In [530]:
```
1 L['UV Index'].value_counts().reset_index().sort_values(by="count",ascer
```

Out[530]:

	UV Index	count
0	2	131
1	1	123
2	3	119
3	4	82
4	0	66
5	9	53
6	7	52
7	10	46
8	8	46
9	11	43

In [531]:
```
1 L['Visibility (km)'].value_counts().reset_index().sort_values(by="count
```

Out[531]:

	Visibility (km)	count
0	6.0	67
1	5.5	59
2	7.0	58
3	7.5	55
4	3.0	53
5	8.5	50
6	8.0	49
7	6.5	48
8	5.0	47
9	2.5	43

In [532]:
```
1 L.groupby(["Weather Type","Cloud Cover"]).agg({'Atmospheric Pressure':
```

Out[532]:

	Weather Type	Cloud Cover	Atmospheric Pressure	Humidity	Temperature C		Wind Speed	Precipitatio (%
			mean	mean	min	max	mean	mea
0	Cloudy	cloudy	1043.947857	45.500000	-37.777778	30.222222	8.000000	51.07142
1	Cloudy	overcast	1009.124710	67.869565	-36.777778	48.222222	7.804348	39.18115
2	Cloudy	partly cloudy	1010.068593	66.674074	-18.777778	39.222222	9.137037	40.95555
3	Rainy	cloudy	956.098333	50.166667	-37.777778	14.222222	7.250000	69.83333
4	Rainy	overcast	1005.418308	78.422886	-34.777778	57.222222	13.664179	75.06965
5	Rainy	partly cloudy	1006.776277	77.702128	-34.777778	31.222222	13.643617	75.57446
6	Snowy	cloudy	980.723571	47.071429	-33.777778	31.222222	6.464286	72.07142
7	Snowy	overcast	997.474000	51.600000	-35.777778	25.222222	10.500000	42.60000
8	Snowy	partly cloudy	1009.736364	43.727273	-31.777778	16.222222	7.545455	58.00000
9	Sunny	clear	1020.000058	53.907514	-6.777778	81.222222	5.760116	22.41618
10	Sunny	cloudy	919.640000	60.000000	-34.777778	29.222222	8.900000	40.40000
11	Sunny	overcast	1091.341667	48.166667	-36.777778	30.222222	7.500000	74.66666
12	Sunny	partly cloudy	1018.428721	51.372093	-34.777778	74.222222	5.959302	23.84883

In [533]:
```
M=C.loc[C['Season']=="Autumn"]
M
```

Out[533]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
17	51	0.5	27.0	overcast	1009.18	3	Autumn	5.5	coa
35	21	10.0	6.0	clear	1026.81	5	Autumn	7.5	coa
39	64	11.5	37.0	partly cloudy	1017.27	3	Autumn	8.0	coa
55	95	13.0	90.0	overcast	997.97	3	Autumn	2.5	coa
104	65	6.0	14.0	partly cloudy	1010.17	8	Autumn	7.0	coa
...	...	...	...	...	...	...	...	...	
13072	71	4.5	23.0	partly cloudy	1015.10	1	Autumn	9.0	coa
13094	65	1.0	30.0	overcast	1018.25	3	Autumn	6.5	coa
13098	95	17.5	62.0	overcast	996.69	3	Autumn	3.0	coa
13105	75	15.0	80.0	partly cloudy	1019.83	5	Autumn	3.0	coa
13197	77	5.5	28.0	overcast	1012.69	3	Autumn	9.0	coa

900 rows × 11 columns

In [534]:
```
1 M.groupby(["Weather Type","Cloud Cover"]).agg({'Atmospheric Pressure':
```

Out[534]:

	Weather Type	Cloud Cover	Atmospheric Pressure	Humidity	Temperature C		Wind Speed	Precipitatio (%
			mean	mean	min	max	mean	mea
0	Cloudy	cloudy	1101.655556	42.555556	-30.777778	7.222222	8.666667	51.66666
1	Cloudy	overcast	1012.304312	67.293578	-33.777778	48.222222	8.339450	38.42201
2	Cloudy	partly cloudy	1008.945000	66.487654	-34.777778	52.222222	8.608025	41.25925
3	Rainy	cloudy	1016.217059	49.411765	-34.777778	27.222222	7.911765	54.58823
4	Rainy	overcast	1006.319596	79.434343	-25.777778	54.222222	13.878788	75.35858
5	Rainy	partly cloudy	999.542247	80.359551	-35.777778	49.222222	13.887640	73.40449
6	Snowy	cloudy	982.751667	49.416667	-37.777778	28.222222	9.000000	49.75000
7	Snowy	overcast	1027.983333	57.888889	-34.777778	9.222222	6.555556	68.66666
8	Snowy	partly cloudy	1035.892857	49.428571	-36.777778	30.222222	6.000000	60.71428
9	Sunny	clear	1019.991055	54.969849	-4.777778	76.222222	6.321608	25.98492
10	Sunny	cloudy	968.970000	55.428571	-35.777778	22.222222	9.714286	63.28571
11	Sunny	overcast	1011.490000	46.555556	-35.777778	31.222222	9.222222	51.11111
12	Sunny	partly cloudy	1023.132192	52.082192	-34.777778	55.222222	6.397260	23.28767

In [535]:
```
1 M['Cloud Cover'].value_counts()
```

Out[535]:
```
Cloud Cover
partly cloudy    331
overcast         325
clear            199
cloudy            45
Name: count, dtype: int64
```

In [536]:
```
1 M['Weather Type'].value_counts()
```

Out[536]:
```
Weather Type
Rainy     304
Sunny     288
Cloudy    280
Snowy      28
Name: count, dtype: int64
```

In [537]:
```
M['UV Index'].value_counts().reset_index().sort_values(by="count",ascer
```

Out[537]:

	UV Index	count
0	1	136
1	2	127
2	3	126
3	4	77
4	0	70
5	7	58

In [538]:
```
M['Visibility (km)'].value_counts().reset_index().sort_values(by="count
```

Out[538]:

	Visibility (km)	count
0	5.0	59
1	8.0	59
2	8.5	56
3	5.5	56
4	6.0	52
5	7.5	52

In [539]:
```
N=C.loc[C['Season']=="Summer"]
N
```

Out[539]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
45	50	0.0	18.0	clear	1016.50	5	Summer	8.0	co
74	31	6.5	18.0	clear	1022.53	7	Summer	8.5	co
80	73	9.5	47.0	partly cloudy	1017.82	1	Summer	7.0	co
81	75	11.5	82.0	partly cloudy	1029.09	10	Summer	7.0	co
91	64	6.0	67.0	partly cloudy	1009.12	3	Summer	4.0	co
...	...	...	...	...	...	...	...	...	
13132	88	12.0	73.0	overcast	994.11	1	Summer	3.0	co
13144	69	11.0	32.0	partly cloudy	1003.45	1	Summer	7.0	co
13145	69	6.5	31.0	overcast	1019.26	2	Summer	9.0	co
13155	70	18.5	84.0	partly cloudy	1003.58	3	Summer	4.0	co
13185	106	23.5	104.0	partly cloudy	998.71	14	Summer	3.0	co

853 rows × 11 columns

In [540]:
```
1 N.groupby(["Weather Type","Cloud Cover"]).agg({'Atmospheric Pressure':
```

Out[540]:

	Weather Type	Cloud Cover	Atmospheric Pressure	Humidity	Temperature C		Wind Speed	Precipitatio (%
			mean	mean	min	max	mean	mea
0	Cloudy	cloudy	1118.923333	71.666667	-35.777778	1.222222	8.500000	36.00000
1	Cloudy	overcast	1013.010882	66.715686	-25.777778	50.222222	9.279412	38.34313
2	Cloudy	partly cloudy	1010.735608	68.560811	-35.777778	49.222222	9.175676	41.00675
3	Rainy	cloudy	1022.770000	48.875000	-27.777778	25.222222	8.875000	46.50000
4	Rainy	overcast	1001.264309	79.287234	-29.777778	58.222222	13.957447	75.80851
5	Rainy	partly cloudy	1004.956932	76.784091	-36.777778	39.222222	13.454545	75.93181
6	Snowy	cloudy	919.552222	51.888889	-37.777778	31.222222	7.833333	47.11111
7	Snowy	overcast	992.466000	50.100000	-20.777778	18.222222	8.700000	61.30000
8	Snowy	partly cloudy	964.928571	61.285714	-33.777778	27.222222	7.928571	35.42857
9	Sunny	clear	1019.811026	51.594872	-3.777778	80.222222	6.320513	20.25641
10	Sunny	cloudy	980.946667	47.333333	-24.777778	29.222222	10.000000	65.00000
11	Sunny	overcast	1056.657143	39.428571	-33.777778	19.222222	7.857143	48.14285
12	Sunny	partly cloudy	1020.346341	51.231707	-14.777778	74.222222	6.219512	27.32926

In [541]:
```
1 N['Cloud Cover'].value_counts()
```

Out[541]:
```
Cloud Cover
partly cloudy    325
overcast         307
clear            195
cloudy            26
Name: count, dtype: int64
```

In [542]:
```
1 N['Weather Type'].value_counts()
```

Out[542]:
```
Weather Type
Sunny     290
Rainy     284
Cloudy    253
Snowy      26
Name: count, dtype: int64
```

In [543]:
```
1 N['UV Index'].value_counts().reset_index().sort_values(by="count",ascer
```

Out[543]:

	UV Index	count
0	1	143
1	2	111
2	3	101
3	0	72
4	10	63
5	4	58

In [544]:
```
1 N['Visibility (km)'].value_counts().reset_index().sort_values(by="count
```

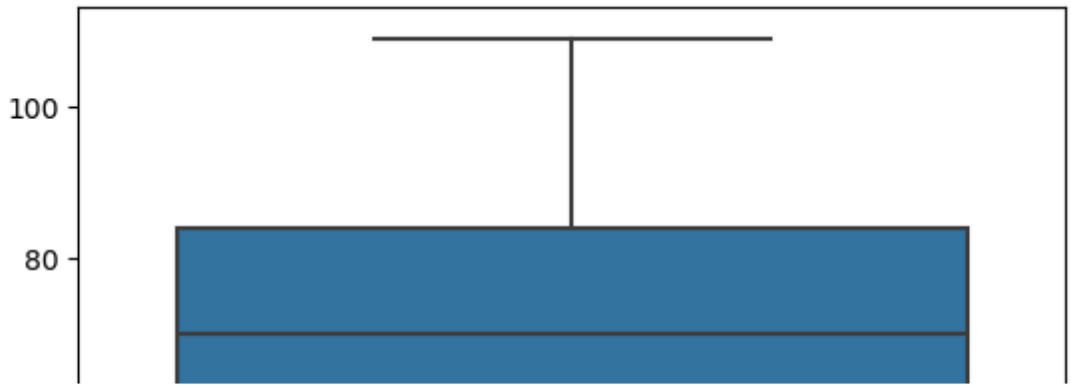
Out[544]:

	Visibility (km)	count
0	8.0	61
1	5.5	60
2	7.5	59
3	6.5	55
4	7.0	51
5	8.5	51

In [545]:

```
import seaborn as sns
for i in X.columns:
    if(X[i].dtype=="int64")|(X[i].dtype=="float64"):
        q1=X[i].quantile(0.25)
        q2=X[i].quantile(0.5)
        q3=X[i].quantile(0.75)
        iqr=q3-q1
        low=q1-(1.5*iqr)
        high=q3+(1.5*iqr)
        print(i.upper())
        print("Q1",q1)
        print("Q2",q2)
        print("Q3",q3)
        print("IQR",iqr)
        print("LOW",low)
        print("High",high)
        Y=X.loc[(X[i]>=low)&(X[i]<=high)]
        sns.boxplot(Y[i])
        plt.show()
```

```
HUMIDITY
Q1 57.0
Q2 70.0
Q3 84.0
IQR 27.0
LOW 16.5
High 124.5
```



In [546]:
```
1 Y
```

Out[546]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	ir
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	ir
2	64	7.0	16.0	clear	1018.72	5	Spring	5.5	mou
3	83	1.5	82.0	clear	1026.25	7	Spring	1.0	co
4	74	17.0	66.0	overcast	990.67	1	Winter	2.5	mou
...	...	...	...	...	...	...	...	...	
13195	74	14.5	71.0	overcast	1003.15	1	Summer	1.0	mou
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	co
13197	77	5.5	28.0	overcast	1012.69	3	Autumn	9.0	co
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	ir
13199	38	0.0	92.0	overcast	1015.37	5	Autumn	10.0	mou

13108 rows × 11 columns

In [547]:
```
1 q1=Y['Temperature C'].quantile(0.25)
2 print('Q1',q1)
3 q2=Y['Temperature C'].quantile(0.5)
4 print('Q2',q2)
5 q3=Y['Temperature C'].quantile(0.75)
6 print('Q3',q3)
7 iqr=q3-q1
8 print('IQR',iqr)
9 low=q1-(1.5*iqr)
10
11 high=q3+(1.5*iqr)
12
```

```
Q1 -13.777777777777779
Q2 3.2222222222222214
Q3 12.222222222222221
IQR 26.0
```

In [548]:
```
1 low
```

Out[548]: -52.77777777777778

In [549]:
```
1 high
```

Out[549]: 51.22222222222222

In [550]:
```
y=pd.get_dummies(X[['Cloud Cover','Season','Location']],drop_first=True
y
```

Out[550]:

	Cloud Cover_cloudy	Cloud Cover_overcast	Cloud Cover_partly cloudy	Season_Spring	Season_Summer	Seasor
0	0	0	1	0	0	
1	0	0	1	1	0	
2	0	0	0	1	0	
3	0	0	0	1	0	
4	0	1	0	0	0	
...	...	...	...	...	...	
13195	0	1	0	0	1	
13196	1	0	0	0	0	
13197	0	1	0	0	0	
13198	0	1	0	0	0	
13199	0	1	0	0	0	

13200 rows × 8 columns

In [551]:
```
A=pd.concat([X,y],axis=1)
A
```

Out[551]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	ir
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	ir
2	64	7.0	16.0	clear	1018.72	5	Spring	5.5	mou
3	83	1.5	82.0	clear	1026.25	7	Spring	1.0	co
4	74	17.0	66.0	overcast	990.67	1	Winter	2.5	mou
...	...	...	...	...	...	...	...	...	
13195	74	14.5	71.0	overcast	1003.15	1	Summer	1.0	mou
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	co
13197	77	5.5	28.0	overcast	1012.69	3	Autumn	9.0	co
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	ir
13199	38	0.0	92.0	overcast	1015.37	5	Autumn	10.0	mou

13200 rows × 19 columns

In [552]:
```
A['Weather Type']=A['Weather Type'].map({"Snowy":4,"Cloudy":3,"Rainy":2
A
```

Out[552]:

	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Loca
0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	ir
1	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	ir
2	64	7.0	16.0	clear	1018.72	5	Spring	5.5	mou
3	83	1.5	82.0	clear	1026.25	7	Spring	1.0	co
4	74	17.0	66.0	overcast	990.67	1	Winter	2.5	mou
...	...	...	...	...	...	...	...	...	
13195	74	14.5	71.0	overcast	1003.15	1	Summer	1.0	mou
13196	76	3.5	23.0	cloudy	1067.23	1	Winter	6.0	co
13197	77	5.5	28.0	overcast	1012.69	3	Autumn	9.0	co
13198	76	10.0	94.0	overcast	984.27	0	Winter	2.0	ir
13199	38	0.0	92.0	overcast	1015.37	5	Autumn	10.0	mou

13200 rows × 19 columns

In [553]:
```
d=A.drop(columns=['Cloud Cover','Season','Location'],axis=1)
d
```

Out[553]:

	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	Visibility (km)	Weather Type	Temperature C
0	73	9.5	82.0	1010.82	2	3.5	2	-3.777778
1	96	8.5	71.0	1011.43	7	10.0	3	21.222222
2	64	7.0	16.0	1018.72	5	5.5	1	12.222222
3	83	1.5	82.0	1026.25	7	1.0	1	20.222222
4	74	17.0	66.0	990.67	1	2.5	2	9.222222
...	...	...	...	...	...	...	...	...
13195	74	14.5	71.0	1003.15	1	1.0	2	-7.777778
13196	76	3.5	23.0	1067.23	1	6.0	4	-18.777778
13197	77	5.5	28.0	1012.69	3	9.0	3	12.222222
13198	76	10.0	94.0	984.27	0	2.0	4	-14.777778
13199	38	0.0	92.0	1015.37	5	10.0	2	-22.777778

13200 rows × 16 columns

In [554]:
```
d['Wind Speed'].unique()
```

Out[554]:
```
array([ 9.5,  8.5,  7. ,  1.5, 17. ,  3.5,  8. ,  6. ,  2. , 10.5, 15. ,
        6.5,  0.5, 12. , 12.5,  7.5, 13.5,  1. ,  4. , 16. , 16.5,  2.5,
       23. ,  3. , 10. , 25.5, 19. , 11.5,  0. ,  9. , 18.5, 11. , 20. ,
       14. ,  5.5, 13. , 46.5,  5. , 18. , 28.5, 14.5,  4.5, 15.5, 28. ,
       19.5, 21.5, 34. , 17.5, 47. , 34.5, 35.5, 23.5, 42.5, 33. , 31.5,
       26. , 22. , 36.5, 27.5, 20.5, 35. , 30. , 26.5, 21. , 32.5, 32. ,
       24. , 27. , 22.5, 31. , 30.5, 24.5, 29.5, 37. , 44.5, 41. , 41.5,
       40.5, 37.5, 46. , 25. , 39. , 29. , 45. , 43.5, 45.5, 36. , 38. ,
       44. , 38.5, 33.5, 40. , 42. , 47.5, 39.5, 43. , 48.5])
```

In [555]:
```
d['Visibility (km)'].unique()
```

Out[555]:
```
array([ 3.5, 10. ,  5.5,  1. ,  2.5,  5. ,  4. ,  7.5,  1.5,  8.5,  6. ,
        8. ,  3. ,  9.5,  9. ,  4.5,  2. , 16.5, 12.5,  6.5,  7. ,  0. ,
       17.5, 17. , 13. , 11. ,  0.5, 16. , 18. , 10.5, 11.5, 19. , 18.5,
       13.5, 15.5, 15. , 14.5, 14. , 12. , 20. , 19.5])
```

In [556]:
```
F=d.drop(columns="Weather Type",axis=1)
T=d[['Weather Type']]
```

In [557]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(F,T,train_size=0.80,rand
```

In [558]:
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler
M=MinMaxScaler()
Std=StandardScaler()
```

In [559]:
```
X_train[['Atmospheric Pressure']]=M.fit_transform(X_train[['Atmospheric
X_train
```

Out[559]:

	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	Visibility (km)	Temperature C	Cl Cover_clc
10330	50	8.5	25.0	0.550803	2	8.5	13.222222	
4687	77	12.0	46.0	0.527450	3	6.5	11.222222	
3396	73	6.5	92.0	0.480618	1	5.0	16.222222	
9196	70	18.0	96.0	0.501867	0	3.5	12.222222	
7312	78	1.5	59.0	0.327295	9	16.5	2.222222	
...	...	...	...	...	...	...	...	
9175	71	5.5	84.0	0.561728	0	14.0	34.222222	
1251	76	11.5	45.0	0.548999	2	6.5	12.222222	
6357	41	3.5	1.0	0.554361	10	6.0	6.222222	
12188	77	10.5	19.0	0.507680	2	6.0	3.222222	
4274	96	15.0	72.0	0.537247	14	2.0	74.222222	

10560 rows × 15 columns

In [560]:
```
X_test[['Atmospheric Pressure']]=M.transform(X_test[['Atmospheric Press
X_test
```

Out[560]:

	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	Visibility (km)	Temperature C	Cl Cover_clc
142	50	5.5	13.0	0.505951	4	8.0	11.222222	
12799	62	18.5	82.0	0.479215	0	4.5	-16.777778	
8613	72	2.5	28.0	0.533288	1	7.5	14.222222	
5628	36	2.0	4.0	0.544063	10	9.5	15.222222	
1858	78	11.0	71.0	0.492946	3	4.5	16.222222	
...	...	...	...	...	...	...	...	
10324	77	24.5	73.0	0.519933	12	4.5	2.222222	
12945	65	15.0	81.0	0.531985	1	5.0	-2.777778	
12213	96	16.0	96.0	0.539477	11	8.5	21.222222	
12518	28	0.5	11.0	0.988674	14	17.5	-11.777778	
11063	70	18.5	85.0	0.464908	1	1.5	-21.777778	

2640 rows × 15 columns

In [561]:
```
X_train[['Temperature C']]=M.fit_transform(X_train[['Temperature C']])
X_train
```

Out[561]:

	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	Visibility (km)	Temperature C	Cl Cover_clc
10330	50	8.5	25.0	0.550803	2	8.5	0.413534	
4687	77	12.0	46.0	0.527450	3	6.5	0.398496	
3396	73	6.5	92.0	0.480618	1	5.0	0.436090	
9196	70	18.0	96.0	0.501867	0	3.5	0.406015	
7312	78	1.5	59.0	0.327295	9	16.5	0.330827	
...	...	...	...	...	...	...	...	
9175	71	5.5	84.0	0.561728	0	14.0	0.571429	
1251	76	11.5	45.0	0.548999	2	6.5	0.406015	
6357	41	3.5	1.0	0.554361	10	6.0	0.360902	
12188	77	10.5	19.0	0.507680	2	6.0	0.338346	
4274	96	15.0	72.0	0.537247	14	2.0	0.872180	

10560 rows × 15 columns

In [562]:
```
X_test[['Temperature C']]=M.transform(X_test[['Temperature C']])
X_test
```

Out[562]:

	Humidity	Wind Speed	Precipitation (%)	Atmospheric Pressure	UV Index	Visibility (km)	Temperature C	Cl Cover_clc
142	50	5.5	13.0	0.505951	4	8.0	0.398496	
12799	62	18.5	82.0	0.479215	0	4.5	0.187970	
8613	72	2.5	28.0	0.533288	1	7.5	0.421053	
5628	36	2.0	4.0	0.544063	10	9.5	0.428571	
1858	78	11.0	71.0	0.492946	3	4.5	0.436090	
...	...	...	...	...	...	...	...	
10324	77	24.5	73.0	0.519933	12	4.5	0.330827	
12945	65	15.0	81.0	0.531985	1	5.0	0.293233	
12213	96	16.0	96.0	0.539477	11	8.5	0.473684	
12518	28	0.5	11.0	0.988674	14	17.5	0.225564	
11063	70	18.5	85.0	0.464908	1	1.5	0.150376	

2640 rows × 15 columns

In [563]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression


L=LogisticRegression()

params={'C':[0.1,0.2,0.3,0.06],"penalty":["l1","l2"]}

G=GridSearchCV(L, param_grid=params,scoring="accuracy",cv=6)
```

In [564]:
```
G.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:
1143: DataConversionWarning: A column-vector y was passed when a 1d arr
ay was expected. Please change the shape of y to (n_samples, ), for exa
mple using ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression (https://scikit-learn.org/stable/modules/linear_model.html#l
ogistic-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:
```

In [565]:
```
G.best_params_
```

Out[565]:
```
{'C': 0.06, 'penalty': 'l2'}
```

In [566]:
```
model=G.best_estimator_
model
```

Out[566]:
```
LogisticRegression(C=0.06)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [567]:
```
pred=model.predict(X_test)
pred
```

Out[567]:
```
array([3, 4, 3, ..., 1, 1, 4], dtype=int64)
```

In [568]:
```
model.score(X_train,y_train)
```

Out[568]:
```
0.7971590909090909
```

In [569]:
```
model.score(X_test,y_test)
```

Out[569]: 0.7916666666666666

In [570]:
```
from sklearn.metrics import accuracy_score,classification_report,confus
```

In [571]:
```
accuracy_score(y_test,pred)
```

Out[571]: 0.7916666666666666

In [572]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           1       0.82      0.83      0.83       641
           2       0.77      0.68      0.72       631
           3       0.83      0.76      0.79       683
           4       0.75      0.89      0.82       685

    accuracy                           0.79      2640
   macro avg       0.79      0.79      0.79      2640
weighted avg       0.79      0.79      0.79      2640
```

In [573]:
```
confusion_matrix(y_test,pred)
```

Out[573]:
```
array([[533,  48,  39,  21],
       [ 33, 426,  42, 130],
       [ 46,  67, 519,  51],
       [ 36,  12,  25, 612]], dtype=int64)
```

In [574]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
K=KNeighborsClassifier()
params={'n_neighbors':[3,8,9,10,12,15]}
G1=GridSearchCV(K, param_grid=params,scoring="accuracy",cv=6)
```

In [575]:
```
G1.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neighbors\_classific
ation.py:215: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  return self._fit(X, y)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neighbors\_classific
ation.py:215: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  return self._fit(X, y)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neighbors\_classific
ation.py:215: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  return self._fit(X, y)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neighbors\_classific
ation.py:215: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
```

In [576]:
```
G1.best_params_
```

Out[576]:
```
{'n_neighbors': 10}
```

In [577]:
```
models=G1.best_estimator_
models
```

Out[577]:
```
KNeighborsClassifier(n_neighbors=10)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [578]:
```
preds=models.predict(X_test)
preds
```

Out[578]:
```
array([1, 2, 3, ..., 1, 1, 2], dtype=int64)
```

In [579]:
```
models.score(X_train,y_train)
```

Out[579]:
```
0.7949810606060606
```

In [580]:
```
models.score(X_test,y_test)
```

Out[580]:
```
0.7287878787878788
```

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
S=SVC()
params={"gamma":[0.1,0.3,0.4,0.5],"kernel":["rbf"]}

G2=GridSearchCV(S, param_grid=params,scoring="accuracy",cv=6)
```

In [ ]:
```
G2.fit(X_train,y_train)
```

In [ ]:
```
G2.best_params_
```

In [584]:
```
Mod=G2.best_estimator_
Mod
```

Out[584]:
```
SVC(gamma=0.1)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [585]:
```
P=Mod.predict(X_test)
P
```

Out[585]:
```
array([3, 2, 3, ..., 3, 2, 2], dtype=int64)
```

In [586]:
```
Mod.score(X_train,y_train)
```

Out[586]:
```
0.9820075757575758
```

In [587]:
```
Mod.score(X_test,y_test)
```

Out[587]:
```
0.7621212121212121
```

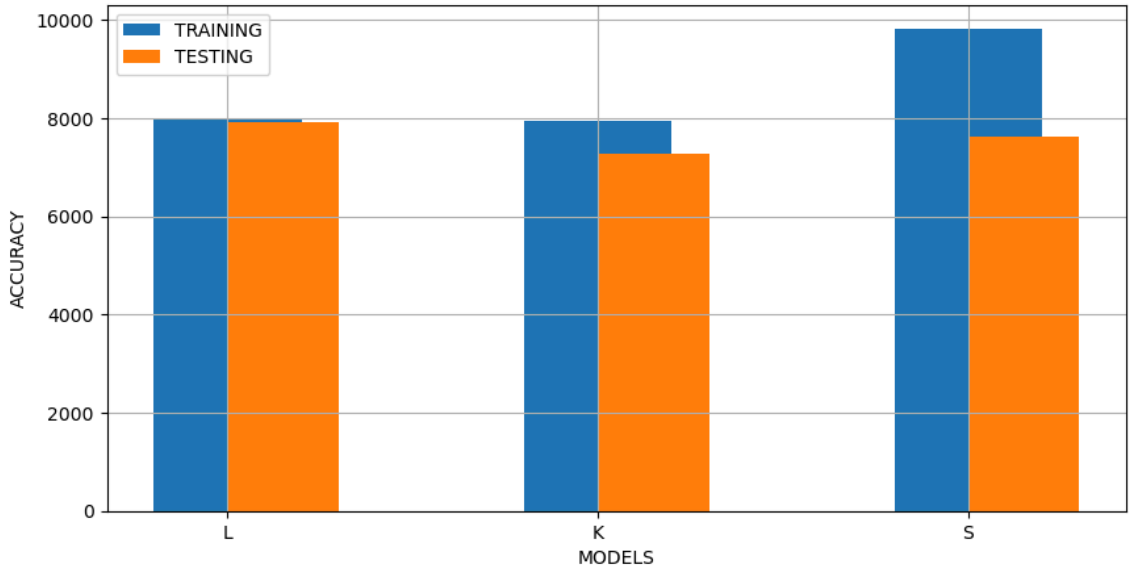
In [588]:
```
C={"method":["L","K","S"],"Train":[79.71,79.49,98.20],"Test":[79.16,72.
```

In [589]:
```
C=pd.DataFrame(C)
C
```

Out[589]:

	method	Train	Test
0	L	79.71	79.16
1	K	79.49	72.87
2	S	98.20	76.21

In [590]:
```
plt.figure(figsize=(10,5))
plt.bar(C['method'],C['Train']*100,align='center',width=0.4,label='TRAI
plt.bar(C['method'],C['Test']*100,align='edge',width=0.3,label='TESTIN
plt.grid()
plt.legend()
plt.xlabel('MODELS')
plt.ylabel('ACCURACY')
plt.show()
```



In [591]:
```
from sklearn.naive_bayes import GaussianNB,BernoulliNB,ComplementNB,Cat
```

In [592]:
```
Models={'Gaussian':GaussianNB(),
        'Bernoulian':BernoulliNB(),
        'Complement':ComplementNB(),
        'Categorical':CategoricalNB(),
        'Multi':MultinomialNB()}
```

In [660]:

```
df=[]

for k,l in Models.items():
    l.fit(X_train,y_train)
    TR=l.score(X_train,y_train)
    TE=l.score(X_test,y_test)
    pred=l.predict(X_test)


    df.append([k,TR,TE])
    print(k.upper())
    print(classification_report(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print('__'*40)
```

```
GAUSSIAN
              precision    recall  f1-score   support

           1       0.83      0.89      0.86       641
           2       0.78      0.73      0.76       631
           3       0.89      0.77      0.82       683
           4       0.82      0.91      0.86       685

    accuracy                           0.83      2640
   macro avg       0.83      0.83      0.82      2640
weighted avg       0.83      0.83      0.83      2640

[[573  40  16  12]
 [ 35 460  35 101]
 [ 46  83 526  28]
 [ 40   3  17 625]]
```

```
BERNOULIAN
              precision    recall  f1-score   support

           1       0.87      0.59      0.70       641
           2       0.59      0.54      0.56       631
           3       0.49      0.57      0.53       683
           4       0.67      0.81      0.73       685

    accuracy                           0.63      2640
   macro avg       0.65      0.63      0.63      2640
weighted avg       0.65      0.63      0.63      2640

[[376  27 162  76]
 [ 21 339 146 125]
 [ 22 193 392  76]
 [ 14  16  98 557]]
```

```
COMPLEMENT
              precision    recall  f1-score   support

           1       0.49      0.87      0.62       641
           2       0.69      0.54      0.60       631
           3       0.89      0.16      0.27       683
           4       0.65      0.83      0.73       685

    accuracy                           0.60      2640
   macro avg       0.68      0.60      0.56      2640
weighted avg       0.68      0.60      0.56      2640

[[560  38   3  40]
 [ 61 339   9 222]
 [467  63 109  44]
 [ 64  52   1 568]]
```

```
CATEGORICAL
              precision    recall  f1-score   support

           1       0.86      0.86      0.86       641
           2       0.82      0.80      0.81       631
           3       0.87      0.83      0.85       683
           4       0.85      0.90      0.87       685
```

```
    accuracy                           0.85      2640
   macro avg       0.85      0.85      0.85      2640
weighted avg       0.85      0.85      0.85      2640

[[550  47  31  13]
 [ 29 504  23  75]
 [ 38  54 569  22]
 [ 26  13  32 614]]
```

```
MULTI
              precision    recall  f1-score   support

           1       0.79      0.81      0.80       641
           2       0.63      0.66      0.64       631
           3       0.84      0.70      0.76       683
           4       0.75      0.81      0.78       685

    accuracy                           0.75      2640
   macro avg       0.75      0.75      0.75      2640
weighted avg       0.75      0.75      0.75      2640

[[520  74  30  17]
 [ 21 417  44 149]
 [ 86  94 481  22]
 [ 29  82  20 554]]
```

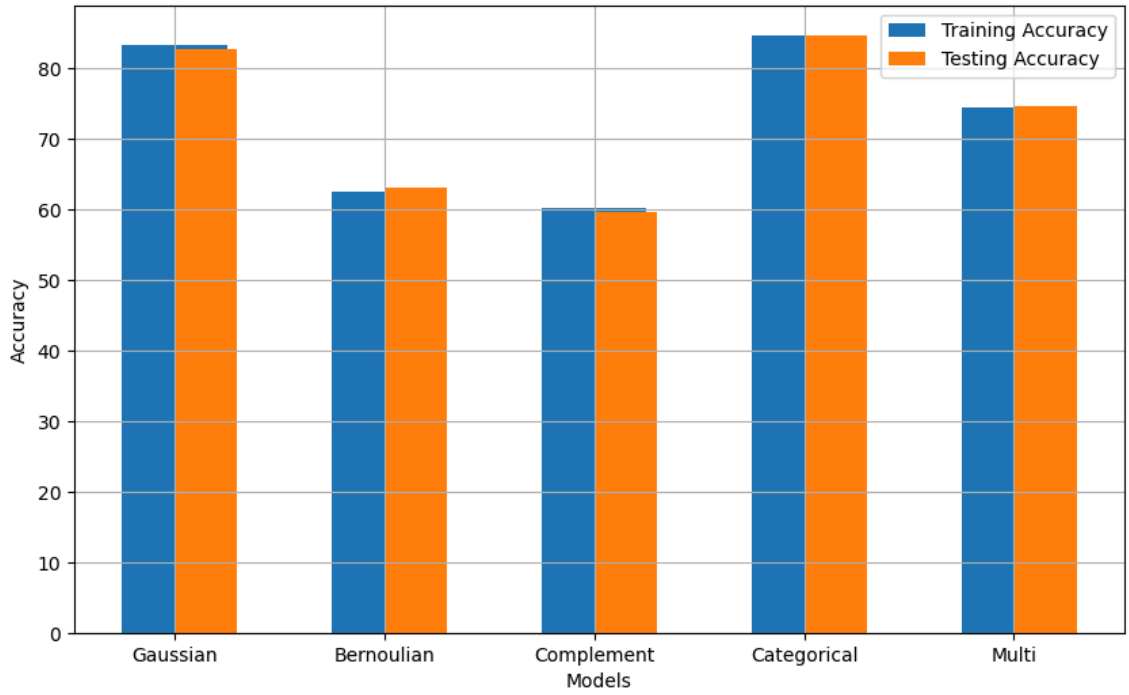
In [594]:

```
R=pd.DataFrame(df,columns=('model name','Train','Test'))
R
```

Out[594]:

	model name	Train	Test
0	Gaussian	0.832670	0.827273
1	Bernoulian	0.625663	0.630303
2	Complement	0.602746	0.596970
3	Categorical	0.846686	0.847348
4	Multi	0.745455	0.746970

In [595]:
```
plt.figure(figsize=(10,6))
plt.bar(R['model name'],R['Train']*100,align='center',width=0.5,label='
plt.bar(R['model name'],R['Test']*100,align='edge',width=0.3,label='Tes
plt.grid()
plt.legend()
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.show()
```



In [596]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
D=DecisionTreeClassifier()
params={'max_depth':[6,7,10,12],'criterion':['entropy'],'min_samples_sp
g=GridSearchCV(D,param_grid=params,scoring='accuracy',cv=6)

```

In [597]:
```
g.fit(X_train,y_train)
```

Out[597]:
```
GridSearchCV(cv=6, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['entropy'], 'max_depth': [6, 7, 10,
12],
                         'min_samples_split': [2, 8, 10]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [598]:
```
g.best_params_
```

Out[598]:
```
{'criterion': 'entropy', 'max_depth': 12, 'min_samples_split': 10}
```

In [599]:
```
DTCE=g.best_estimator_
DTCE
```

Out[599]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=12, min_samples_spli
t=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [600]:
```
prd=DTCE.predict(X_test)
prd
```

Out[600]:
```
array([3, 4, 3, ..., 1, 3, 4], dtype=int64)
```

In [601]:
```
DTCE.score(X_train,y_train)
```

Out[601]:
```
0.953219696969697
```

In [602]:
```
DTCE.score(X_test,y_test)
```

Out[602]:
```
0.9030303030303031
```

In [603]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
D1=DecisionTreeClassifier()
params={'max_depth':[5,7,10,12],'criterion':['gini'],'min_samples_split
Gd=GridSearchCV(D1,param_grid=params,scoring='accuracy',cv=6)

```

In [604]:
```
Gd.fit(X_train,y_train)
```

Out[604]:
```
GridSearchCV(cv=6, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini'], 'max_depth': [5, 7, 10, 1
2],
                         'min_samples_split': [2, 5, 8, 9]},
             scoring='accuracy')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [605]:
```
Gd.best_params_
```

Out[605]:
```
{'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 9}
```

In [606]:
```
G_M=Gd.best_estimator_
G_M
```

Out[606]:
```
DecisionTreeClassifier(max_depth=10, min_samples_split=9)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [607]:
```
pdt=G_M.predict(X_test)
pdt
```

Out[607]:
```
array([3, 4, 3, ..., 1, 1, 4], dtype=int64)
```

In [608]:
```
G_M.score(X_train,y_train)
```

Out[608]:
```
0.9551136363636363
```

In [609]:
```
G_M.score(X_test,y_test)
```

Out[609]:
```
0.9026515151515152
```

In [610]:
```
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier,
R1= RandomForestClassifier(n_estimators=25)
Bag=BaggingClassifier(estimator=K,n_estimators=75)
V=VotingClassifier(estimators=[('log',LogisticRegression()),('KNN',KNei
ST=StackingClassifier(estimators=[('NB',GaussianNB()),('Svc',SVC()),('K
```

In [611]:
```
R1.fit(X_train,y_train)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_6700\2357190787.py:1: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  R1.fit(X_train,y_train)
```

Out[611]:
```
RandomForestClassifier(n_estimators=25)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [612]:
```
R1.score(X_train,y_train)
```

Out[612]:
```
0.999905303030303
```

In [613]:
```
R1.score(X_test,y_test)
```

Out[613]:
```
0.9071969696969697
```

In [654]:
```
R2= RandomForestClassifier(n_estimators=55)
R2.fit(X_train,y_train)
```

Out[654]:
```
RandomForestClassifier(n_estimators=55)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [657]:
```
print (R2.score(X_train,y_train))
print (R2.score(X_test,y_test))
```

```
1.0
0.9098484848484848
```

In [614]:
```
V.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.p
y:99: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.p
y:134: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[614]:
```
VotingClassifier(estimators=[('log', LogisticRegression()),
                             ('KNN', KNeighborsClassifier()),
                             ('NB', GaussianNB())])
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [615]:
```
V.score(X_train,y_train)
```

Out[615]:
```
0.8445075757575757
```

In [616]:
```
1 V.score(X_test,y_test)
```

Out[616]:
```
0.8223484848484849
```

In [617]:
```
1 Bag.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_bagging.py:802: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[617]:
```
BaggingClassifier(estimator=KNeighborsClassifier(), n_estimators=75)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [618]:
```
1 Bag.score(X_train,y_train)
```

Out[618]:
```
0.8434659090909091
```

In [619]:
```
1 Bag.score(X_test,y_test)
```

Out[619]:
```
0.728030303030303
```

In [623]:
```
1 ST.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.p
y:99: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.p
y:134: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[623]:
```
StackingClassifier(estimators=[('NB', GaussianNB()), ('Svc', SVC()),
                               ('KNN', KNeighborsClassifier())],
                   final_estimator=LogisticRegression())
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [624]:
```
1 ST.score(X_train,y_train)
```

Out[624]:
```
0.8836174242424243
```

In [625]:
```
1 ST.score(X_test,y_test)
```

Out[625]:
```
0.8549242424242425
```

In [633]:
```
1 V1=VotingClassifier(estimators=[('S',SVC()),('MUlt',MultinomialNB()),(
2 ST1=StackingClassifier(estimators=[('NB',ComplementNB()),("s",SVC()),('
```

In [629]:
```
1 V1.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.p
y:99: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.p
y:134: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
```

Out[629]:
```
VotingClassifier(estimators=[('S', SVC()), ('MUlt', MultinomialNB()),
                             ('NB', BernoulliNB()),
                             ('DT', DecisionTreeClassifier())])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [630]:
```
1 V1.score(X_train,y_train)
```

Out[630]:
```
0.8699810606060606
```

In [631]:
```
1 V1.score(X_test,y_test)
```

Out[631]:
```
0.8397727272727272
```

In [643]:
```
1 ST1.fit(X_train,y_train)
2
```

Out[643]:
```
StackingClassifier(estimators=[('NB', ComplementNB()), ('s', SVC()),
                               ('Log', LogisticRegression()),
                               ('nb', ComplementNB()),
                               ('DT', DecisionTreeClassifier())],
                   final_estimator=KNeighborsClassifier())
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [646]:
```
1 ST1.score(X_train,y_train)
```

Out[646]:
```
0.9479166666666666
```

In [637]:
```
1 ST1.score(X_test,y_test)
```

Out[637]:
```
0.8958333333333334
```

In [649]:
```
from sklearn.ensemble import AdaBoostClassifier
AD=AdaBoostClassifier(n_estimators=50)
AD.fit(X_train,y_train)
```

Out[649]:
```
AdaBoostClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [651]:
```
AD.score(X_train,y_train)
```

Out[651]:
```
0.862310606060606
```

In [652]:
```
AD.score(X_test,y_test)
```

Out[652]:
```
0.8587121212121213
```

```
                          #REPORT
  # Here we are going to discuss about Classification of weather in areas like inland,Coastal,Mountain.
   # We have information about the parameters of weather such as Temperature,Humidity,Wind speed,UV index etc.

  #APPROACH:
          As started to find variation in parameters according to season but could'nt conclude any impact thus by going through each column and comparing it and started finding impact of each on others It helps to understand whole data.

   ##CONCLUSION:-
               As per data FOR Inland: as Humidity (60,100),temp lies between(-25,16) & precipitation increases from 50 to above with low atmospheric pressure Causes majorly Cloudy or Overcast with moderate visibility(1-10) which causes Snowy weather mostly.
               F
```