1.Importing Necessary Libraries

In [84]:
```python
import pandas as pd
import numpy as np
import matplotlib as mlt
import sklearn
import scipy
import seaborn as sns
import missingno as msno
```

2.Importing the Dataset

In [85]:
```python
data=pd.read_csv("E:\IBM_Project\weatherAUS.csv")
```

3.Analyse the data

In [86]:
```python
data.describe()
```

Out[86]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pres
count	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	14219
mean	12.131807	23.174186	2.326738	3.129340	3.988338	37.377592	13.869248	18.292855	67.984915	50.175564	917.357684	91
std	6.440548	7.194768	8.426426	4.166674	4.688665	16.433198	8.954477	9.075870	20.416069	22.071424	303.403253	30
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	7.500000	17.900000	0.000000	0.000000	0.000000	30.000000	7.000000	11.000000	56.000000	35.000000	1011.000000	100
50%	12.000000	22.600000	0.000000	1.600000	0.200000	37.000000	13.000000	17.000000	70.000000	51.000000	1016.700000	101
75%	16.800000	28.200000	0.600000	5.400000	8.700000	46.000000	19.000000	24.000000	83.000000	65.000000	1021.800000	101
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.000000	1041.000000	103

In [87]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 24 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           142193 non-null  object 
 1   Location       142193 non-null  object 
 2   MinTemp        142193 non-null  float64
 3   MaxTemp        142193 non-null  float64
 4   Rainfall       142193 non-null  float64
 5   Evaporation    142193 non-null  float64
 6   Sunshine       142193 non-null  float64
 7   WindGustDir    142193 non-null  object 
 8   WindGustSpeed  142193 non-null  int64  
 9   WindDir9am     142193 non-null  object 
 10  WindDir3pm     142193 non-null  object 
 11  WindSpeed9am   142193 non-null  int64  
 12  WindSpeed3pm   142193 non-null  int64  
 13  Humidity9am    142193 non-null  int64  
 14  Humidity3pm    142193 non-null  int64  
 15  Pressure9am    142193 non-null  float64
 16  Pressure3pm    142193 non-null  float64
 17  Cloud9am       142193 non-null  int64  
 18  Cloud3pm       142193 non-null  int64  
 19  Temp9am        142193 non-null  float64
 20  Temp3pm        142193 non-null  float64
 21  RainToday      142193 non-null  object 
 22  RISK_MM        142193 non-null  float64
 23  RainTomorrow   142193 non-null  object 
dtypes: float64(10), int64(7), object(7)
memory usage: 26.0+ MB
```

In [88]:
```python
data.head()
```

Out[88]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm
0	01-12-2008	Albury	13.4	22.9	0.6	0.0	0.0	W	44	W	...	22	1007.7	1007.1	8	0
1	02-12-2008	Albury	7.4	25.1	0.0	0.0	0.0	WNW	44	NNW	...	25	1010.6	1007.8	0	0
2	03-12-2008	Albury	12.9	25.7	0.0	0.0	0.0	WSW	46	W	...	30	1007.6	1008.7	0	2
3	04-12-2008	Albury	9.2	28.0	0.0	0.0	0.0	NE	24	SE	...	16	1017.6	1012.8	0	0
4	05-12-2008	Albury	17.5	32.3	1.0	0.0	0.0	W	41	ENE	...	33	1010.8	1006.0	7	8

5 rows × 24 columns

In [89]:
```python
data.shape
```

Out[89]: (142193, 24)

4.Handling Missing values

Checking whether Null values exit

In [90]:
```python
data.isnull().sum()
```

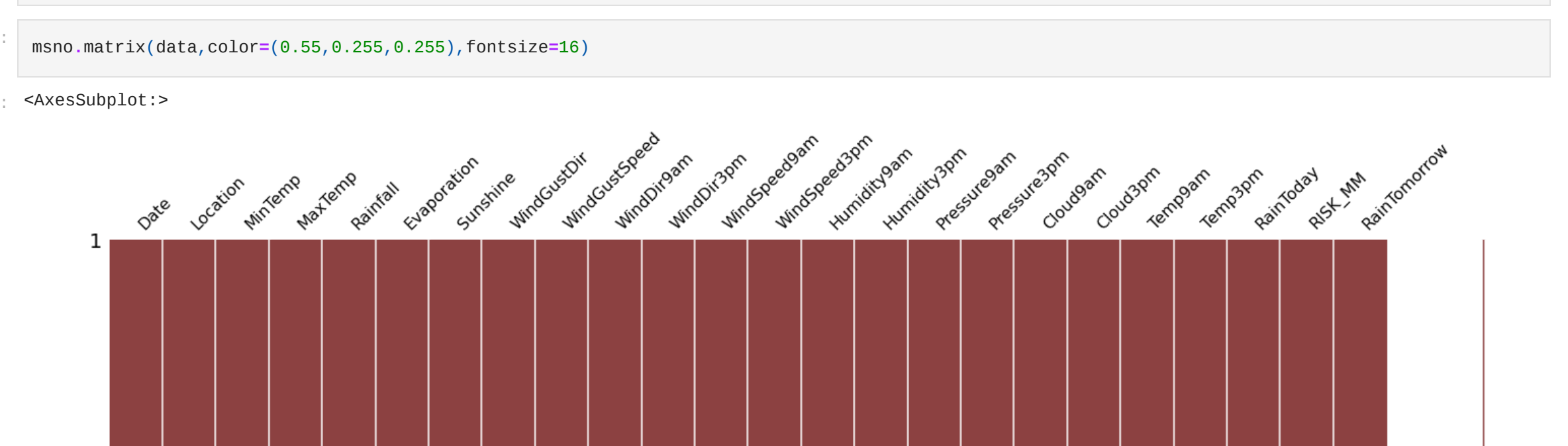
Out[90]:
```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RISK_MM          0
RainTomorrow     0
dtype: int64
```

Dealing with missing values

In [91]:
```python
import missingno as msno
```

In [92]:
```python
msno.matrix(data,color=(0.55,0.255,0.255),fontsize=16)
```

Out[92]: <AxesSubplot:>



Imputing Data

In [93]:
```python
data_c=data[["RainToday","WindGustDir","WindDir9am","WindDir3pm"]]
```

In [94]:
```python
data.drop(columns=["Evaporation","Sunshine","Cloud9am","Cloud3pm"],axis=1,inplace=True)
data.drop(columns=["RainToday","WindGustDir","WindDir9am","WindDir3pm"],axis=1,inplace=True)
```

In [95]:
```python
data['MinTemp'].fillna(data['MinTemp'].mean(),inplace=True)
data['MaxTemp'].fillna(data['MaxTemp'].mean(),inplace=True)
data['Rainfall'].fillna(data['Rainfall'].mean(),inplace=True)
data['WindGustSpeed'].fillna(data['WindGustSpeed'].mean(),inplace=True)
data['WindSpeed9am'].fillna(data['WindSpeed9am'].mean(),inplace=True)
data['WindSpeed3pm'].fillna(data['WindSpeed3pm'].mean(),inplace=True)
data['Humidity3pm'].fillna(data['Humidity3pm'].mean(),inplace=True)
data['Humidity9am'].fillna(data['Humidity9am'].mean(),inplace=True)
data['Temp9am'].fillna(data['Temp9am'].mean(),inplace=True)
data['Temp3pm'].fillna(data['Temp3pm'].mean(),inplace=True)
```

In [96]:
```python
c_names=data_c.columns
```

In [97]:
```python
from sklearn.impute import SimpleImputer
```

In [98]:
```python
imp_mode=SimpleImputer(missing_values=np.nan,strategy="most_frequency")
```

In [99]:
```python
data_c=pd.DataFrame(data_c,columns=c_names)
```

In [100]:
```python
data=pd.concat([data,data_c],axis=1)
```

5.Data Visualization

Data Correlation

In [101]:
```python
data.corr()
```

Out[101]:

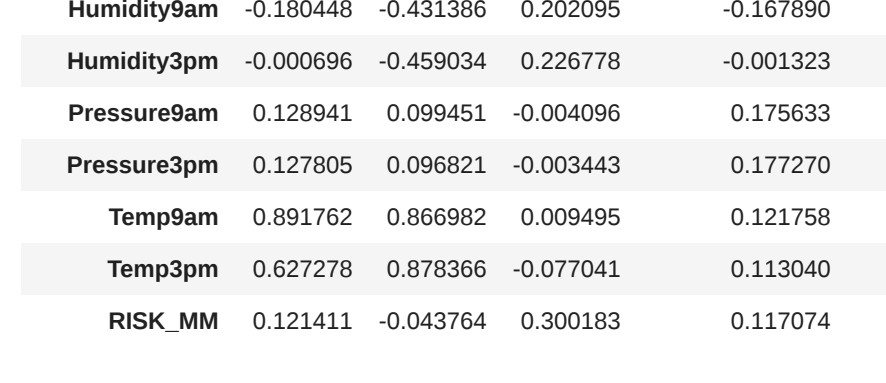
	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am	Temp3pm	RISK_MI
MinTemp	1.000000	0.722484	0.102244	0.130381	0.181565	0.173819	-0.180448	-0.000696	0.128941	0.127805	0.891762	0.627278	0.12141
MaxTemp	0.722484	1.000000	-0.073387	0.085229	0.021062	0.061310	-0.431386	-0.459034	0.099451	0.096821	0.866982	0.878366	-0.04376
Rainfall	0.102244	-0.073387	1.000000	0.094167	0.084768	0.052425	0.202095	0.226778	-0.004096	-0.003443	0.009495	-0.077041	0.30018
WindGustSpeed	0.130381	0.085229	0.094167	1.000000	0.527075	0.608283	-0.167890	-0.001323	0.175633	0.177270	0.121758	0.113040	0.11707
WindSpeed9am	0.181565	0.021062	0.084768	0.527075	1.000000	0.513931	-0.224094	-0.013411	0.132073	0.130537	0.137988	0.023547	0.06652
WindSpeed3pm	0.173819	0.061310	0.052425	0.608283	0.513931	1.000000	-0.116146	0.081109	0.197209	0.205051	0.164344	0.106350	0.04372
Humidity9am	-0.180448	-0.431386	0.202095	-0.167890	-0.224094	-0.116146	1.000000	0.613533	-0.007590	-0.014754	-0.366070	-0.389508	0.15255
Humidity3pm	-0.000696	-0.459034	0.226778	-0.001323	-0.013411	0.081109	0.613533	1.000000	0.015457	0.024935	-0.205338	-0.347502	0.27851
Pressure9am	0.128941	0.099451	-0.004096	0.175633	0.132073	0.197209	-0.007590	0.015457	1.000000	0.983630	0.129621	0.137550	-0.00545
Pressure3pm	0.127805	0.096821	-0.003443	0.177270	0.130537	0.205051	-0.014754	0.024935	0.983630	1.000000	0.121841	0.144317	-0.00563
Temp9am	0.891762	0.866982	0.009495	0.121758	0.137988	0.164344	-0.366070	-0.205338	0.129621	0.121841	1.000000	0.764786	0.04569
Temp3pm	0.627278	0.878366	-0.077041	0.113040	0.023547	0.106350	-0.389508	-0.347502	0.137550	0.144317	0.764786	1.000000	-0.06862
RISK_MM	0.121411	-0.043764	0.300183	0.117074	0.066521	0.043727	0.152556	0.278517	-0.005457	-0.005636	0.045691	-0.068622	1.00000

Heat map

In [102]:
```python
cor=data.corr()
```

In [103]:
```python
sns.heatmap(data=cor,xticklabels=cor.columns.values,yticklabels=cor.columns.values)
```

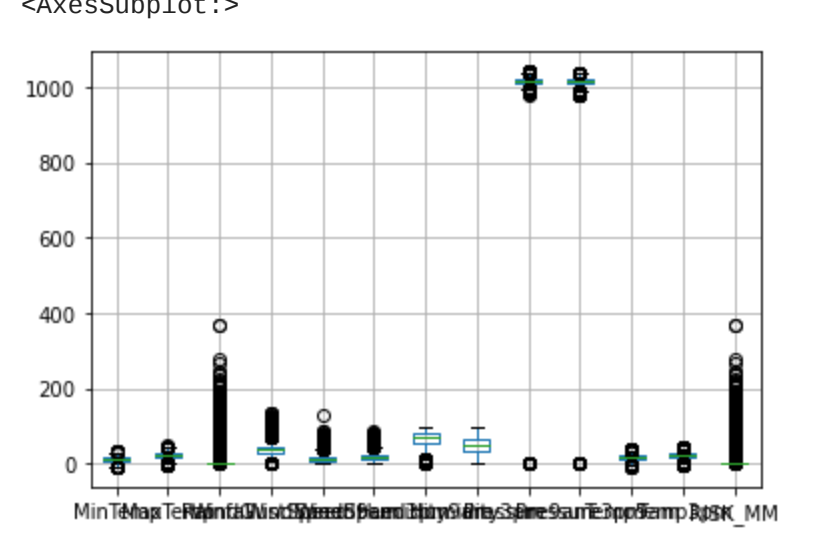
Out[103]: <AxesSubplot:>



Box plot

In [104]:
```python
data.boxplot()
```

Out[104]: <AxesSubplot:>

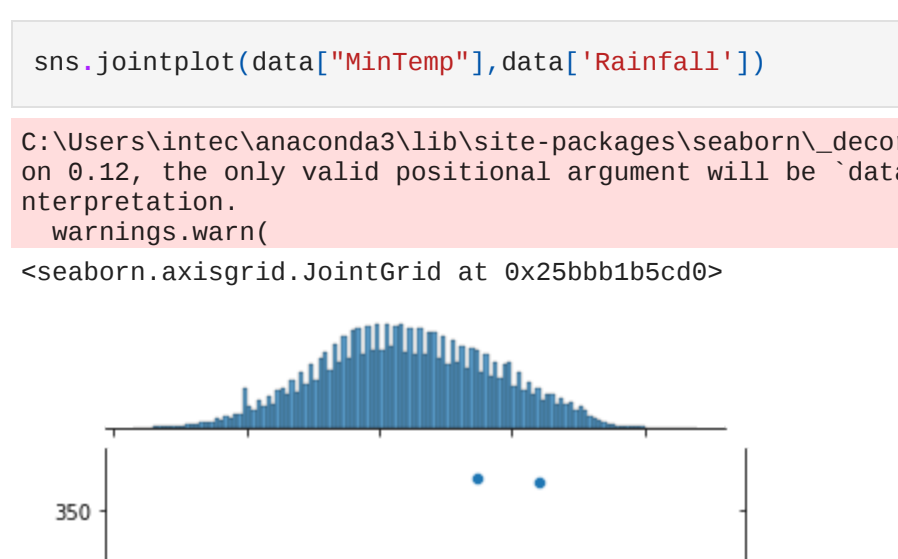


Joint plot

In [105]:
```python
sns.jointplot(data["MinTemp"],data['Rainfall'])
```

```
C:\Users\intec\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From versi
on 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misi
nterpretation.
  warnings.warn(
```

Out[105]: <seaborn.axisgrid.JointGrid at 0x25bbb1b5cd0>



In [106]:
```python
# Joint plot with hue function
```

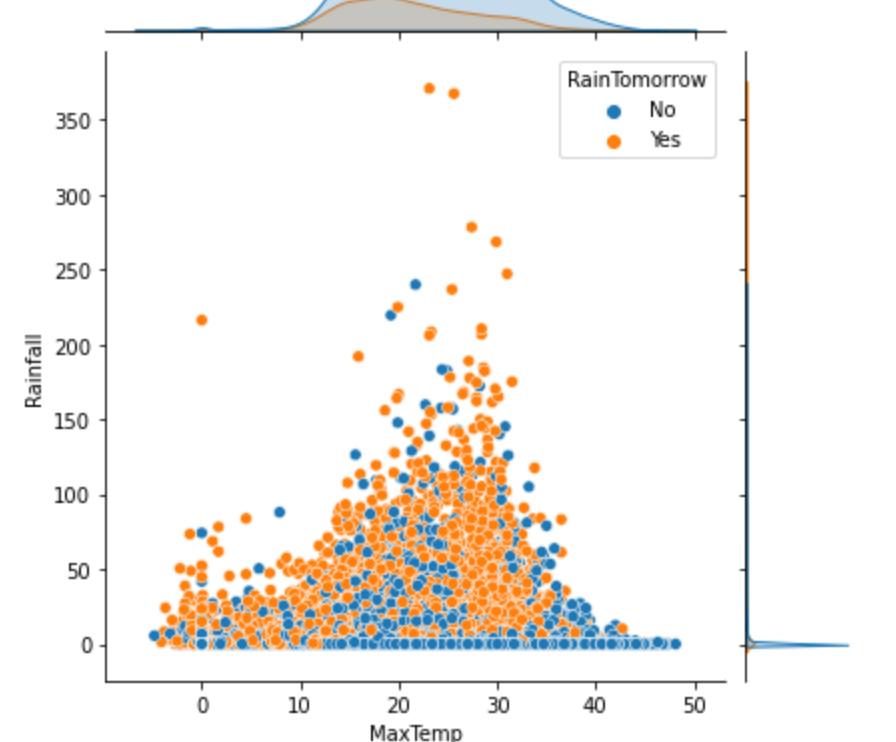
In [107]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [108]:
```python
data=pd.read_csv("E:\IBM_Project\weatherAUS.csv")
```

In [109]:
```python
sns.jointplot(data['MaxTemp'],data['Rainfall'],hue=data['RainTomorrow'])
```

```
C:\Users\intec\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From versi
on 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misi
nterpretation.
  warnings.warn(
```

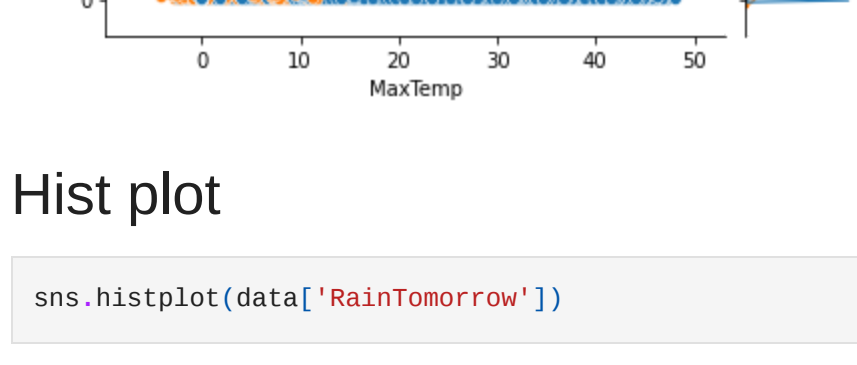
Out[109]: <seaborn.axisgrid.JointGrid at 0x25bd8506f10>



Hist plot

In [110]:
```python
sns.histplot(data['RainTomorrow'])
```

Out[110]: <AxesSubplot:xlabel='RainTomorrow', ylabel='Count'>

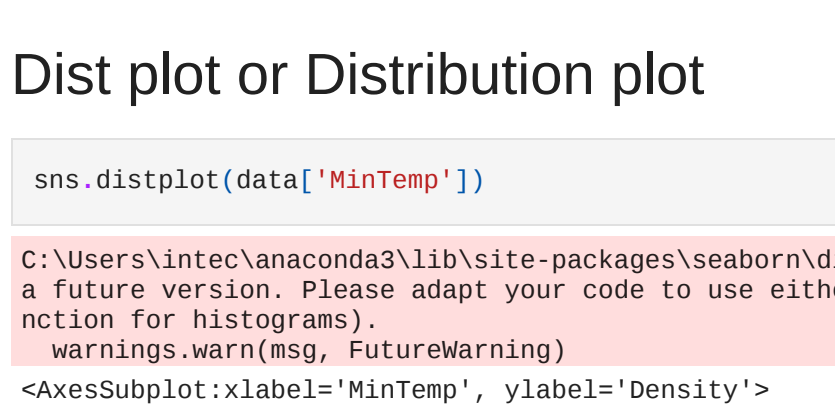


6.Scatter plot

In [111]:
```python
sns.scatterplot(data['MaxTemp'],data['Rainfall'])
```

```
C:\Users\intec\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From versi
on 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misi
nterpretation.
  warnings.warn(
```

Out[111]: <AxesSubplot:xlabel='MaxTemp', ylabel='Rainfall'>

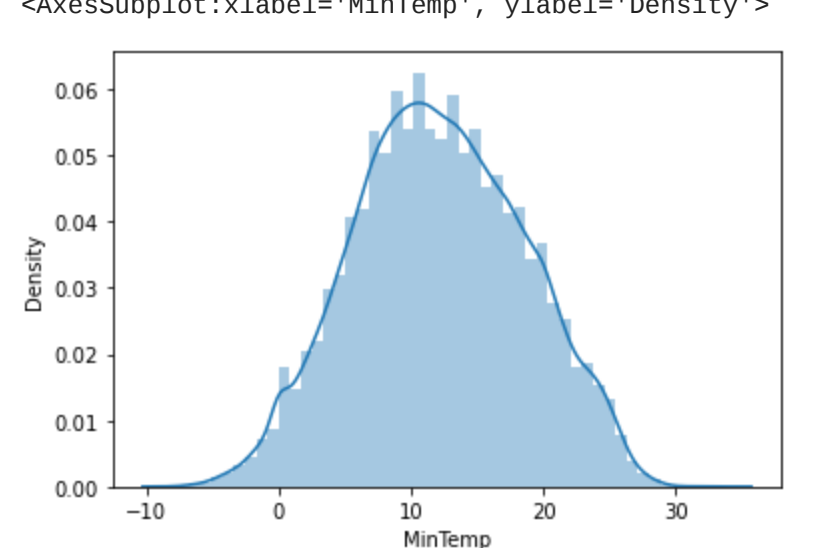


Dist plot or Distribution plot

In [112]:
```python
sns.distplot(data['MinTemp'])
```

```
C:\Users\intec\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level fu
nction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[112]: <AxesSubplot:xlabel='MinTemp', ylabel='Density'>



6.Splitting the dataset into Dependent and Independent Variable

In [113]:
```python
from sklearn.preprocessing import StandardScaler
```

In [114]:
```python
data=data[data['RainTomorrow'].notnull()]
```

In [115]:
```python
data['Pressure9am'].fillna(data['Pressure9am'].mean(),inplace=True)
data['Pressure3pm'].fillna(data['Pressure3pm'].mean(),inplace=True)
```

In [116]:
```python
y=data['RainTomorrow']
x=data.drop('RainTomorrow',axis=1)
```

In [117]:
```python
set(y)
```

Out[117]: {'No', 'Yes'}

In [118]:
```python
x=x.drop('Date',axis=1)
```

In [119]:
```python
names=x.columns
```

In [120]:
```python
names
```

Out[120]:
```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RISK_MM'],
      dtype='object')
```

In [121]:
```python
sc=StandardScaler()
```

In [122]:
```python
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
```

In [123]:
```python
print(len(x),len(y))
```

142193 142193

7.Label Encoding

In [144]:
```python
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
```

In [145]:
```python
print(len(x),len(y))
```

142193 142193

In [146]:
```python
## RainToday     WindGustDir    WindDir9am    WindDir3pm

LE = LabelEncoder()
x['Location'] = LE.fit_transform(x['Location'])
x.head()

LE = LabelEncoder()
x['RainToday'] = LE.fit_transform(x['RainToday'])
x.head()

LE = LabelEncoder()
x['WindGustDir'] = LE.fit_transform(x['WindGustDir'])
x.head()

LE = LabelEncoder()
x['WindDir9am'] = LE.fit_transform(x['WindDir9am'])
x.head()

LE = LabelEncoder()
x['WindDir3pm'] = LE.fit_transform(x['WindDir3pm'])
x.head()
```

Out[146]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm
0	2	0.196908	-0.038109	-0.204920	-0.751043	-0.850637	14	0.402991	14	15	...	0.147682	-1.276568	0.297764	0.303160
1	2	-0.734693	0.267670	-0.276125	-0.751043	-0.850637	15	0.402991	7	16	...	-1.174810	-1.140645	0.307322	0.305475
2	2	0.119275	0.351064	-0.276125	-0.751043	-0.850637	16	0.524696	14	16	...	-1.468697	-0.914107	0.297435	0.308451
3	2	-0.455212	0.670742	-0.276125	-0.751043	-0.850637	5	-0.814062	10	1	...	-1.125829	-1.548413	0.330394	0.322011
4	2	0.833502	1.268400	-0.157450	-0.751043	-0.850637	14	0.220433	2	8	...	0.686476	-0.778184	0.307982	0.299522

5 rows × 22 columns

In [147]:
```python
LE = LabelEncoder()
y=pd.DataFrame(y)
y = LE.fit_transform(y)
```

```
C:\Users\intec\anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:115: 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)
```

In [148]:
```python
print(len(x),len(y))
```

142193 142193

In [149]:
```python
sc=StandardScaler()
```

In [150]:
```python
x=sc.fit_transform(x)
```

In [151]:
```python
x[:5]
```

Out[151]:
```
array([[-1.5270045 ,  0.19690834, -0.03810929, -0.20492006, -0.75104289,
        -0.85063691,  1.17249935,  0.40299101,  1.29072649,  1.36214594,
         0.68466012,  0.62882836,  0.14768248, -1.27656756,  0.29776422,
         0.30315951,  1.67163053, -0.88360606,  0.00309923,  0.07059605,
        -0.48941675, -0.27844993],
       [-1.5270045 , -0.73469264,  0.26766956, -0.27612488, -0.75104289,
        -0.85063691,  1.37259921,  0.40299101, -0.14767056,  1.57244417,
        -1.10216181,  0.40846304, -1.17480986, -1.14064472,  0.30732249,
         0.30547456, -0.88184222, -0.88360606,  0.04847488,  0.40452007,
        -0.48941675, -0.27844993],
       [-1.5270045 ,  0.11927493,  0.35106379, -0.27612488, -0.75104289,
        -0.85063691,  1.57269906,  0.52469629,  1.29072649,  1.57244417,
         0.57298375,  0.84919368, -1.46869705, -0.91410665,  0.29743463,
         0.30845106, -0.88184222, -0.22787705,  0.62323311,  0.2575935 ,
        -0.48941675, -0.27844993],
       [-1.5270045 , -0.45521235,  0.67074166, -0.27612488, -0.75104289,
        -0.85063691, -0.62839936, -0.81406186,  0.46878532, -1.58202933,
        -0.32042722, -1.02391153, -1.12582866, -1.54841324,  0.33039418,
         0.32201063, -0.88184222, -0.88360606,  0.18460183,  0.69837321,
        -0.48941675, -0.16049674],
       [-1.5270045 ,  0.83350235,  1.26840031, -0.15745018, -0.75104289,
        -0.85063691,  1.17249935,  0.22043308, -1.17509703, -0.1099417 ,
        -0.7671327 ,  0.18809772,  0.68647566, -0.77818381,  0.30798168,
         0.29952158,  1.35244644,  1.73930998,  0.13922618,  1.12579597,
        -0.48941675, -0.25485929]])
```

In [152]:
```python
x=pd.DataFrame(x,columns=names)
```

8.Splitting the data into Train and Test

In [153]:
```python
from sklearn import model_selection
```

In [154]:
```python
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.2,random_state=0)
```

9.Training and Testing the model

In [166]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [167]:
```python
RFC=RandomForestClassifier()
```

In [168]:
```python
GBC=GradientBoostingClassifier()
```

In [169]:
```python
np.any(np.isnan(x))
```

Out[169]: False

In [170]:
```python
GBC.fit(x_train,y_train)
```

Out[170]:
▾ GradientBoostingClassifier
GradientBoostingClassifier()

In [171]:
```python
RFC.fit(x_train,y_train)
```

Out[171]:
▾ RandomForestClassifier
RandomForestClassifier()

In [172]:
```python
data.isnull().any()
```

Out[172]:
```
Date             False
Location         False
MinTemp          False
MaxTemp          False
Rainfall         False
Evaporation      False
Sunshine         False
WindGustDir      False
WindGustSpeed    False
WindDir9am       False
WindDir3pm       False
WindSpeed9am     False
WindSpeed3pm     False
Humidity9am      False
Humidity3pm      False
Pressure9am      False
Pressure3pm      False
Cloud9am         False
Cloud3pm         False
Temp9am          False
Temp3pm          False
RainToday        False
RISK_MM          False
RainTomorrow     False
dtype: bool
```

In [173]:
```python
x.isnull().any()
```

Out[173]:
```
Location         False
MinTemp          False
MaxTemp          False
Rainfall         False
Evaporation      False
Sunshine         False
WindGustDir      False
WindGustSpeed    False
WindDir9am       False
WindDir3pm       False
WindSpeed9am     False
WindSpeed3pm     False
Humidity9am      False
Humidity3pm      False
Pressure9am      False
Pressure3pm      False
Cloud9am         False
Cloud3pm         False
Temp9am          False
Temp3pm          False
RainToday        False
RISK_MM          False
dtype: bool
```

In [184]:
```python
p1=RFC.predict(x_train)
```

In [185]:
```python
p2=RFC.predict(x_test)
```

10.Model Evaluation

In [186]:
```python
import sklearn.metrics as metrics
```

In [189]:
```python
print(metrics.accuracy_score(y_train,p1))
```

1.0

In [191]:
```python
print(metrics.accuracy_score(y_test,p2))
```

1.0

11.Save The Model

In [192]:
```python
import pickle
```

In [194]:
```python
pickle.dump(RFC,open('rainfall.pkl','wb'))
pickle.dump(LE,open('encoder.pkl','wb'))
pickle.dump(imp_mode,open('imputer.pkl','wb'))
pickle.dump(sc,open('scale.pkl','wb'))
```

In [ ]: