In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
```

In [5]:

```
pwd
```

Out[5]:

```
'C:\\Users\\mohit'
```

In [6]:

```
df = pd.read_csv(r'C:\\Users\\mohit\\data.txt')

df.sample(5)
```

Out[6]:

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
749	12.7	19.5	8.0	1017.6	69.0	1.3	8.3	18.3	179.116667
655	31.9	35.0	28.3	1001.3	72.0	2.1	2.8	9.4	38.250000
928	15.9	22.4	8.4	1017.6	66.0	2.4	7.2	16.5	332.708333
136	28.6	35.6	25.3	1005.4	79.0	2.6	5.7	13.0	43.958333
544	14.4	21.1	6.8	1019.1	65.0	1.6	5.6	13.0	245.916667

In [7]:

```
print("df.shape : ", df.shape)
```

```
df.shape :  (1093, 9)
```

In [8]:

```
print(df.columns)
```

```
Index(['T', 'TM', 'Tm', 'SLP', 'H', 'VV', 'V', 'VM', 'PM 2.5'], dtype='obj
ect')
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1093 entries, 0 to 1092
Data columns (total 9 columns):
T          1093 non-null float64
TM         1093 non-null float64
Tm         1093 non-null float64
SLP        1093 non-null float64
H          1093 non-null float64
VV         1093 non-null float64
V          1093 non-null float64
VM         1093 non-null float64
PM 2.5     1092 non-null float64
dtypes: float64(9)
memory usage: 76.9 KB
```

In [10]:

```
df[df['PM 2.5'].isnull()]
```

Out[10]:

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
184	14.3	19.2	10.9	1020.5	91.0	1.6	4.8	11.1	NaN

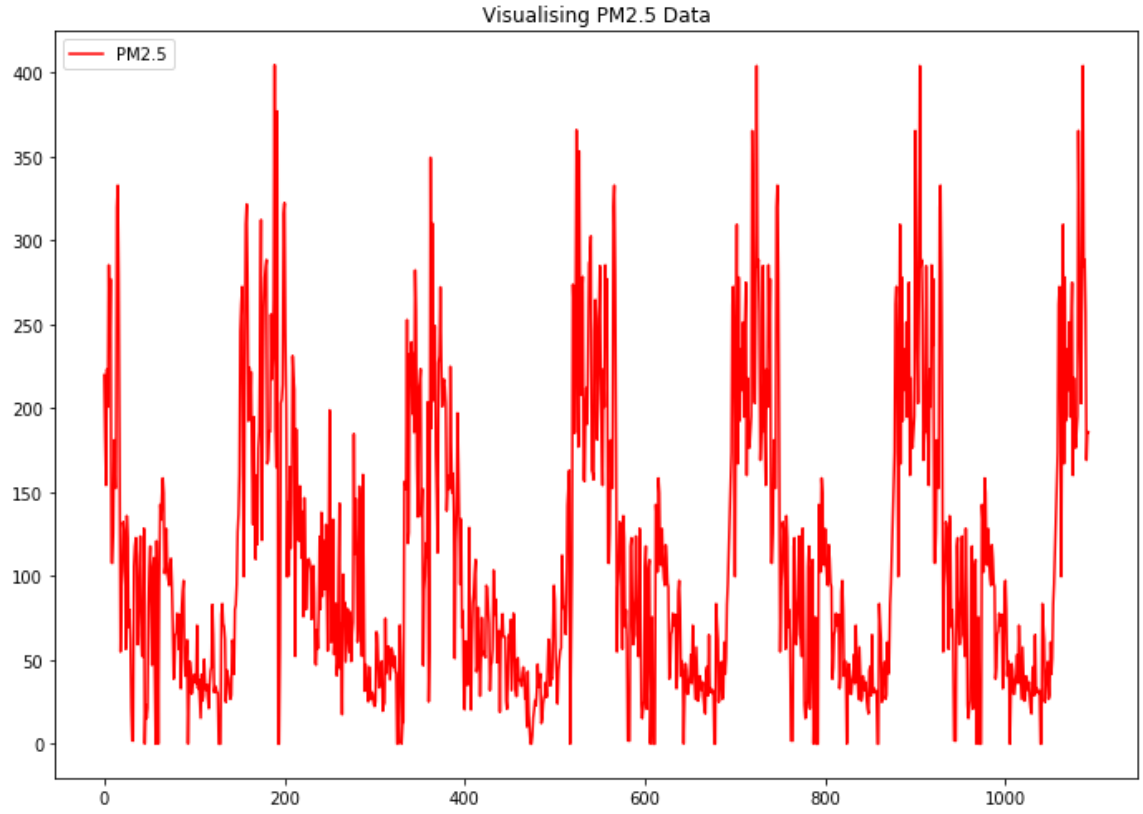
In [11]:

```
df.fillna(method='ffill', inplace = True)
```

In [12]:

```
rcParams['figure.figsize'] = 11.7,8.27

sns.lineplot(data = df["PM 2.5"], color="red", label="PM2.5")
plt.title('Visualising PM2.5 Data')
plt.show()
```



In [13]:

```
duplicateRowsDF = df[df.duplicated()]

print("Duplicate Rows except first occurrence based on all columns are :")
print(duplicateRowsDF)
```

Duplicate Rows except first occurrence based on all columns are :

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
731	16.9	25.1	6.6	1021.3	65.0	1.1	2.0	7.6	284.795833
732	15.5	24.1	7.7	1021.0	71.0	1.1	3.5	11.1	219.720833
733	14.9	22.8	8.0	1018.4	73.0	1.1	5.9	13.0	182.187500
734	18.3	24.7	11.5	1018.1	85.0	0.5	1.1	7.6	154.037500
735	16.5	24.7	12.0	1017.4	82.0	0.6	8.0	13.0	223.208333
736	16.6	22.2	10.2	1015.4	65.0	1.8	10.0	20.6	200.645833
737	18.2	25.1	12.0	1014.4	77.0	0.6	5.0	11.1	285.225000
738	15.6	23.2	10.9	1015.6	82.0	0.6	4.3	9.4	236.825000
739	13.0	19.0	9.6	1016.9	85.0	0.8	10.7	16.5	276.908333
740	12.3	19.0	8.9	1015.8	91.0	0.5	4.6	9.4	108.000000
741	11.9	15.7	7.5	1018.9	86.0	0.8	6.1	11.1	107.625000
742	12.0	17.0	6.7	1019.8	75.0	1.3	7.6	13.0	125.891667
743	11.7	18.0	6.4	1017.3	76.0	0.8	4.4	13.0	181.012500
744	14.6	21.0	6.8	1015.6	81.0	0.6	2.4	7.6	152.554167
745	17.7	24.0	9.0	1014.5	69.0	1.4	4.4	9.4	152.320833
746	17.4	27.2	12.2	1016.4	79.0	1.1	7.8	18.3	319.737500
747	15.9	22.4	8.4	1017.6	66.0	2.4	7.2	16.5	332.708333
748	15.2	22.6	6.6	1018.2	56.0	1.9	8.9	22.2	279.600000
749	12.7	19.5	8.0	1017.6	69.0	1.3	8.3	18.3	179.116667
750	15.6	24.6	10.7	1015.7	81.0	1.3	2.8	9.4	54.791667
751	17.1	23.2	9.0	1017.1	69.0	1.8	5.9	11.1	93.375000
752	17.6	24.0	10.1	1014.8	67.0	1.4	5.9	14.8	103.000000
753	17.2	26.0	11.5	1014.7	65.0	2.3	5.7	14.8	132.208333
754	17.7	24.4	9.7	1017.0	64.0	2.1	8.0	18.3	127.708333
755	17.4	24.4	10.1	1018.9	61.0	2.3	6.1	11.1	109.333333
756	23.1	30.0	15.6	1014.9	71.0	1.4	4.1	9.4	56.458333
757	23.7	29.8	18.7	1014.0	75.0	1.4	11.1	18.3	135.833333
758	20.9	27.8	15.8	1014.4	62.0	2.6	17.2	27.8	122.500000
759	22.6	30.3	11.5	1015.4	57.0	2.4	5.4	11.1	69.666667
760	22.7	30.4	12.8	1017.7	56.0	2.6	2.4	7.6	79.833333
...	...	...	...	...	...	...	...	...	...
1063	23.4	31.1	14.2	1016.0	60.0	1.1	3.0	7.6	217.375000
1064	22.6	30.6	15.8	1015.4	69.0	0.5	1.1	5.4	309.375000
1065	23.9	32.0	14.5	1015.6	63.0	1.0	2.0	5.4	166.916667
1066	23.2	30.2	14.0	1012.6	55.0	1.1	7.4	18.3	278.000000
1067	22.5	29.5	15.6	1013.3	56.0	1.1	11.7	22.2	192.333333
1068	21.7	30.0	12.2	1014.5	52.0	1.1	6.3	16.5	235.333333
1069	22.2	30.2	13.6	1015.6	63.0	1.6	1.5	3.5	221.541667
1070	20.8	29.3	12.1	1016.7	58.0	1.8	3.5	9.4	210.541667
1071	20.7	28.8	10.6	1017.8	56.0	1.8	6.1	16.5	251.208333
1072	20.8	29.0	10.5	1013.9	61.0	1.8	2.6	7.6	194.708333
1073	22.0	31.0	10.7	1013.2	56.0	1.8	2.4	14.8	250.875000
1074	21.4	30.6	10.8	1011.9	54.0	2.6	7.8	18.3	274.875000
1075	24.2	31.0	16.5	1013.0	36.0	2.9	13.9	22.2	160.000000
1076	22.7	30.8	13.6	1012.0	38.0	3.4	11.9	20.6	217.916667
1077	21.4	29.6	12.7	1013.7	40.0	2.6	8.0	20.6	184.208333
1078	19.2	25.9	11.9	1017.0	84.0	0.8	2.6	9.4	176.375000
1079	18.9	25.8	11.5	1017.9	77.0	0.6	5.0	13.0	185.958333
1080	19.3	27.0	11.7	1019.6	57.0	1.8	5.2	11.1	195.375000
1081	17.8	24.8	11.2	1017.0	74.0	0.6	0.7	7.6	365.291667
1082	15.8	24.3	8.7	1015.8	77.0	0.8	1.7	7.6	312.208333
1083	18.2	27.0	7.7	1013.7	72.0	0.6	1.1	7.6	237.625000
1084	19.0	27.1	10.5	1015.1	61.0	1.6	3.3	9.4	202.875000
1085	18.0	26.6	10.5	1017.1	61.0	1.9	8.0	14.8	260.166667
1086	17.2	24.2	10.0	1017.2	58.0	1.6	9.3	22.2	403.958333
1087	16.3	23.0	6.6	1016.9	61.0	1.0	9.4	20.6	284.166667
1088	18.1	24.0	11.2	1015.4	56.0	1.8	15.9	25.9	288.416667
1089	17.8	25.0	10.7	1015.8	54.0	2.3	9.4	22.2	256.833333
1090	13.9	24.5	11.4	1015.0	95.0	0.6	8.7	14.8	169.000000

```
1091  16.3  23.0   9.8  1016.9  78.0  1.1   7.4  16.5  186.041667
1092  16.3  23.4   9.0  1017.3  68.0  1.3   7.8  18.3  185.583333

[362 rows x 9 columns]
```

In [14]:

```
df = df[df.duplicated() == False]

print(df.shape)
```

```
(731, 9)
```

In [15]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 731 entries, 0 to 730
Data columns (total 9 columns):
T         731 non-null float64
TM        731 non-null float64
Tm        731 non-null float64
SLP       731 non-null float64
H         731 non-null float64
VV        731 non-null float64
V         731 non-null float64
VM        731 non-null float64
PM 2.5    731 non-null float64
dtypes: float64(9)
memory usage: 57.1 KB
```

In [16]:

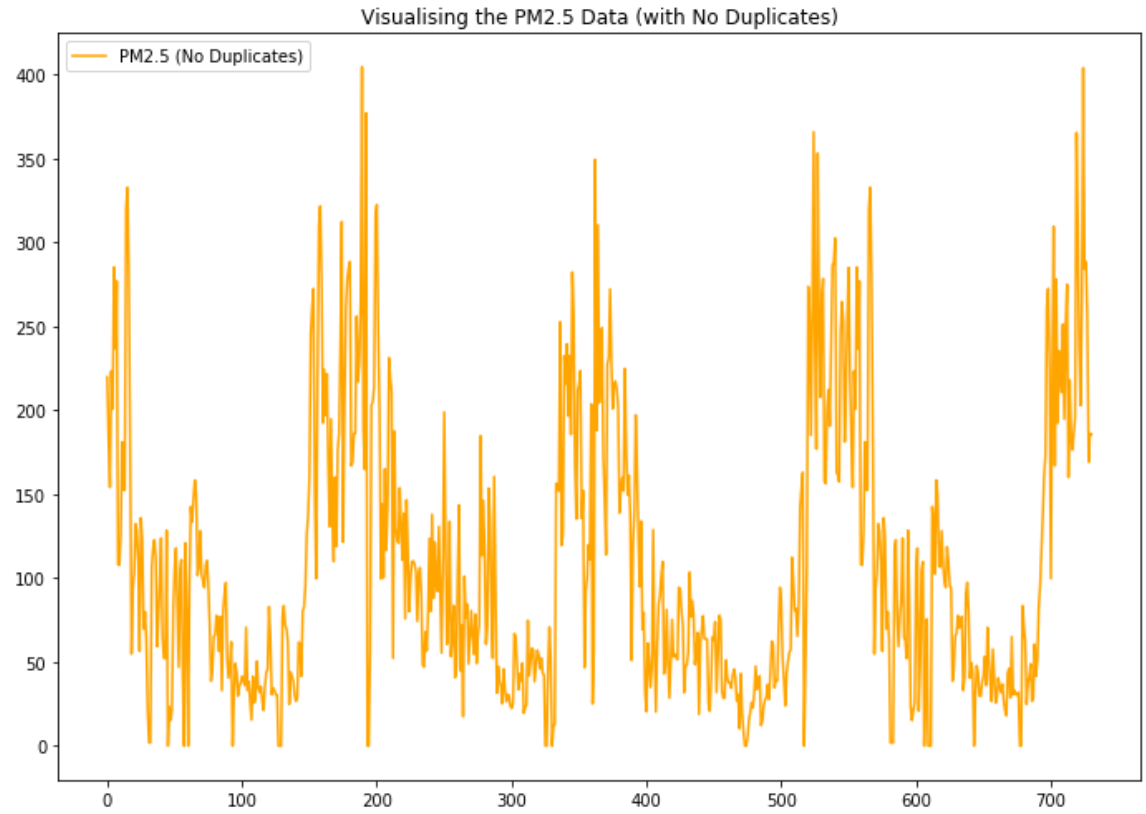
```
df.describe().T
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
T	731.0	25.556908	7.286760	6.7	18.850000	27.700000	31.2000	38.5
TM	731.0	32.231874	6.852548	9.8	27.550000	34.100000	36.9000	45.5
Tm	731.0	19.273598	7.454202	0.0	12.000000	21.000000	25.9000	32.7
SLP	731.0	1008.196854	7.564601	991.5	1001.300000	1008.500000	1015.0000	1023.2
H	731.0	64.134063	15.460460	20.0	55.000000	66.000000	75.0000	98.0
VV	731.0	1.891929	0.686446	0.3	1.400000	1.900000	2.4000	5.8
V	731.0	6.630369	3.967064	0.4	3.500000	6.300000	9.1000	24.4
VM	731.0	15.696580	7.654019	1.9	11.100000	14.800000	18.3000	57.6
PM 2.5	731.0	108.036007	82.570588	0.0	43.583333	81.833333	154.0375	404.5

In [17]:

```
sns.lineplot(data = df["PM 2.5"], color="orange", label="PM2.5 (No Duplicates)")
plt.title('Visualising the PM2.5 Data (with No Duplicates)')
plt.show()
```



In [18]:

```
pm_ = list(df['PM 2.5'])

Acceptable = []

for pm in pm_:
    if pm >= 151:
        Acceptable.append(0)
    else:
        Acceptable.append(1)

df['Acceptable'] = Acceptable

df.sample(3)
```

Out[18]:

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5	Acceptable
282	35.9	40.5	29.6	996.3	46.0	1.9	13.7	27.8	71.000000	1
546	13.7	21.0	5.6	1018.8	71.0	1.3	1.5	7.6	253.291667	0
139	29.1	34.0	23.8	1007.8	72.0	1.9	6.5	13.0	30.333333	1

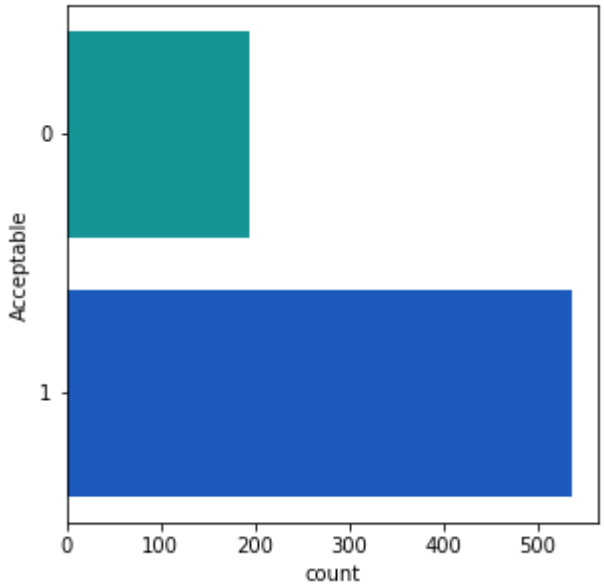
In [19]:

```
rcParams['figure.figsize'] = 4.75,4.75

print(df.Acceptable.value_counts())

sns.countplot(y = "Acceptable", data = df, palette = 'winter_r')
plt.show()
```

```
1    536
0    195
Name: Acceptable, dtype: int64
```

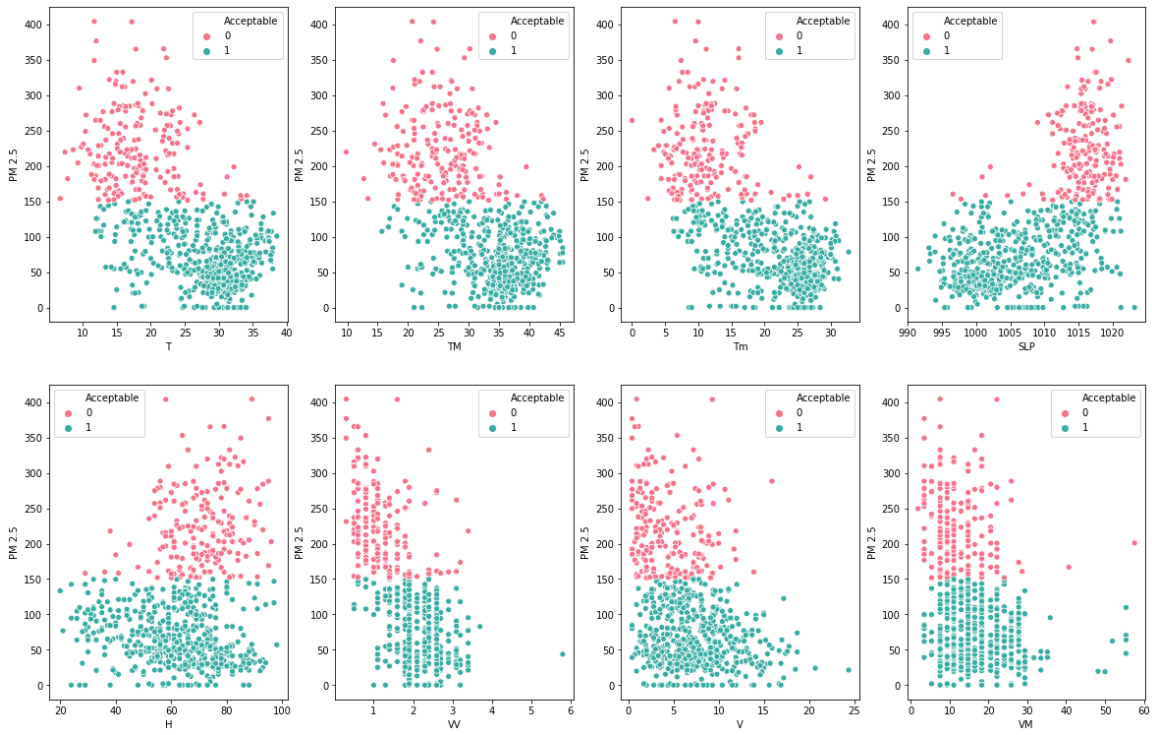


In [20]:

```
cols =  list(df.columns)

plt.figure(figsize=(20, 20))

for i in range(1, 9):
    plt.subplot(3, 4, i)
    sns.scatterplot(x = cols[i - 1], y = df['PM 2.5'],data = df, hue = "Acceptable", pa
lette = "husl")
```

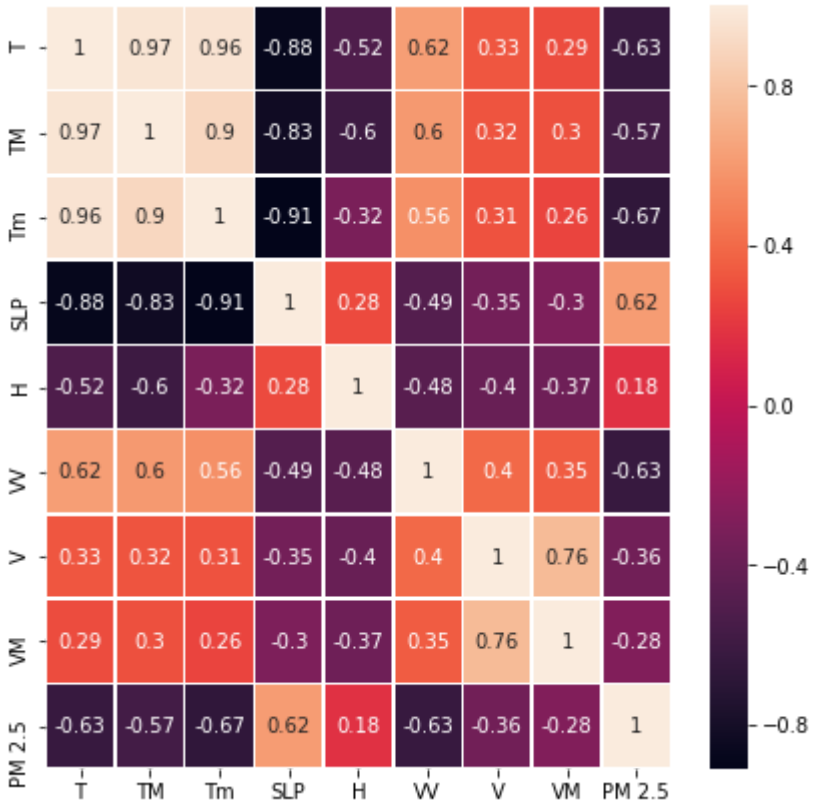


In [21]:

```
corrmat = df.corr()
top_corr_features = corrmat.index[:-1] # dropping 'Acceptable'

plt.figure(figsize=(7,7))

#plot heat map
sns.heatmap(df[top_corr_features].corr(),annot = True, linewidths=.5)
# print(corrmat)
plt.show()
```

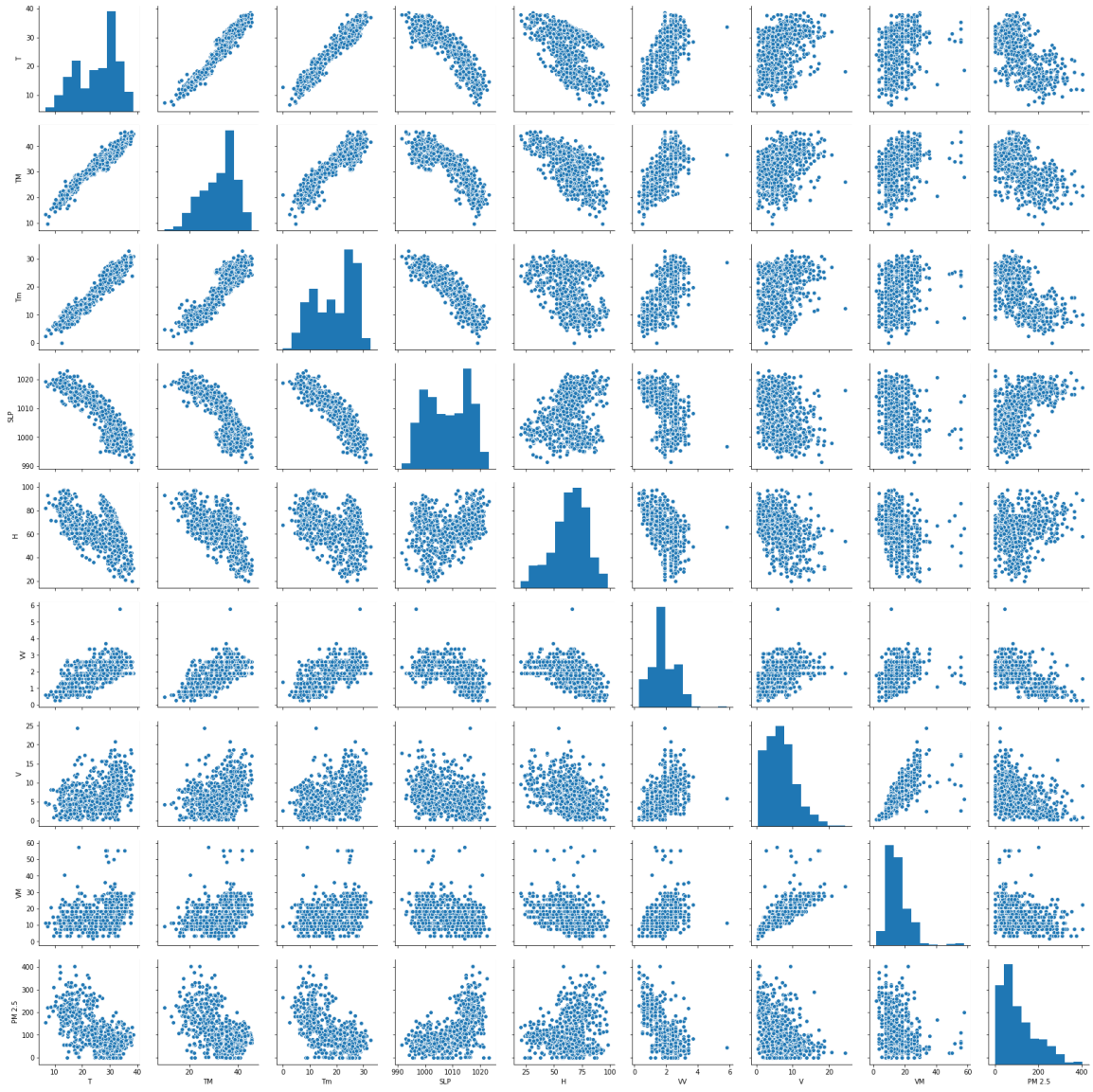


In [124]:

```
df=df.drop(columns=['Acceptable'])
sns.pairplot(df)
```

Out[124]:

```
<seaborn.axisgrid.PairGrid at 0x2ac1b822048>
```



In [111]:

```
from sklearn.model_selection import train_test_split

X = df[[ 'T','TM','Tm','SLP', 'H', 'VV','VM','V']]
y = df['PM 2.5']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.10, random_stat
e = 45)
```

In [112]:

```
print('Length of X_train', len(X_train))
print('Length of X_test', len(X_test))
print('Length of y_train', len(y_train))
print('Length of y_test', len(y_test))
```

```
Length of X_train 657
Length of X_test 74
Length of y_train 657
Length of y_test 74
```

In [113]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_train, y_train)

print("lr.score : ", lr.score(X_test, y_test))
print("lr.coef_ : ",lr.coef_)
print("lr.intercept_ : ",lr.intercept_)
```

```
lr.score :  0.5815697908768345
lr.coef_ :  [ 2.47563967e+00  1.18110511e-01 -6.76615900e+00  6.82122386e-
01
 -8.52242344e-01 -5.11596250e+01  4.83066566e-04 -2.38566327e+00]
lr.intercept_ :  -347.7946587578919
```

In [114]:

```
print("lr.score of training data: ", lr.score(X_train, y_train))
```

```
lr.score of training data:  0.5830171993914903
```

In [115]:

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(lr , X, y, cv = 5)

print(score)
print("Mean score : ",score.mean())
```

```
[0.51308156 0.48833908 0.64009098 0.5746025  0.11785428]
Mean score :  0.46679367929828236
```

In [116]:

```
coeff_df = pd.DataFrame(lr.coef_, X.columns, columns = ['Coefficient'])
coeff_df
```

Out[116]:

	Coefficient
T	2.475640
TM	0.118111
Tm	-6.766159
SLP	0.682122
H	-0.852242
VV	-51.159625
VM	0.000483
V	-2.385663

In [117]:

```
from sklearn import metrics

prediction = lr.predict(X_test)

print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 35.85179578758336
MSE: 2346.3462266920264
RMSE: 48.43909812013459
```

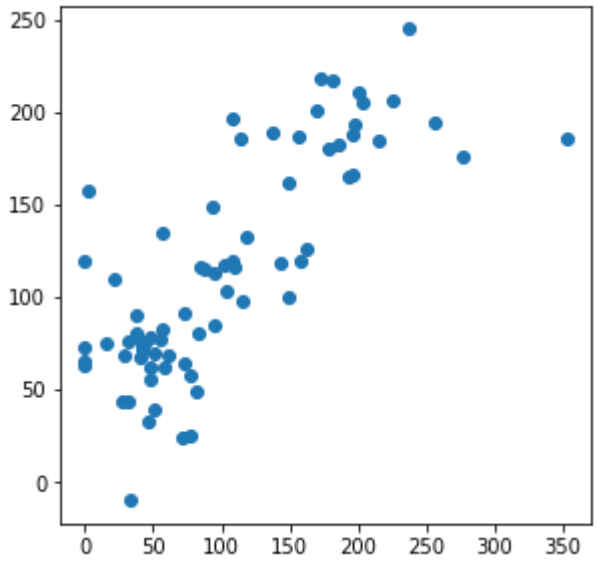
In [118]:

```
print("Prediction Values:",prediction)
```

```
Prediction Values: [ 84.9000544   70.09310635 194.48334457  61.98594328 20
6.53816528
  72.53579137  77.54957535 161.77080028 115.81803067 112.77372192
  62.29374444 180.84318833 218.25113149 125.97593756 185.52993937
 245.14122294  75.59316248 166.43686914 176.44095854  99.84425111
  24.35985227  57.85442018  75.85377473  83.16723478 134.77803406
  74.21361449 120.21984023  68.83337599 205.93042145  55.5785411
  44.15169644 132.404925    43.54120596 157.35196631 196.2669262
 165.13133185 192.93915462  80.65740297 210.28494422  65.59881865
  72.73257801 185.87864154 103.6041744   68.79168989 187.04457541
  63.65788708 117.76850183  33.08776217  49.31486929  90.3951732
  68.19873978 116.0178151  182.22027739 117.03277852 217.72724318
  64.61171007  80.74515108 119.04699867 109.62013309  98.3338109
 184.40469849  76.26861545 120.20854029  78.76025376  91.14734821
 200.98936259 120.24557938 189.34429322  25.38968336 149.52164827
  -9.30565996  39.38568419  75.26452782 188.11108454]
```

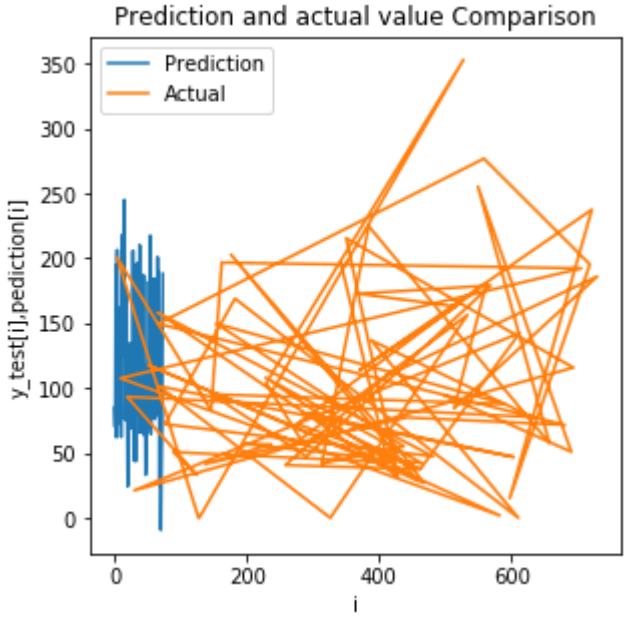
In [119]:

```
plt.scatter(y_test,prediction)
plt.show()
```



In [120]:

```
plt.title("Prediction and actual value Comparison")
plt.plot(prediction)
plt.ylabel('y_test[i],pediction[i]')
plt.xlabel('i')
plt.plot(y_test)
plt.legend(['Prediction','Actual'],loc='upper left')
plt.show()
```



In [ ]:

In [ ]:

```

```