1. AirBNB:

Q1.1 Hierarchial

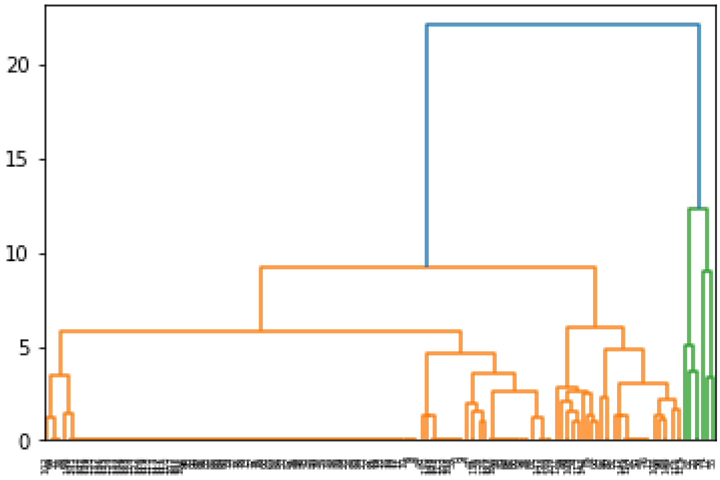
Hierarchical clustering with more records will take huge time and resources so taking a subset will solve the probelm and here I am taking 150 sample

In [2]:

```
import pandas as pd

reviews_df = pd.read_csv("C:/Users/satya/OneDrive - Texas State University/ShareKnowledge/Courses/QMST5336-ANA/Assignment/2/reviews.csv",header = 0, delimiter = ",")
reviews_null_drop_df = reviews_df.dropna()
reviews_col_null_drop_df = reviews_null_drop_df.drop(['id','host_since'], axis=1)
sample_df = reviews_col_null_drop_df.sample (150)
```

In [5]:

```
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
# create dendrogram with data df_0
dendrogram = sch.dendrogram(sch.linkage(sample_df, method = 'ward'))
# create clusters
hc = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean', linkage = 'ward')
# save clusters for chart
y_hc = hc.fit_predict ( sample_df )
# Here you can use different data for prediction
```



Q1.2 K-means clustering

Kmean clustering with 3 clusters.

NO results of Hierarchical and K-means clustering looks different. The Hierarchical clustering divided all the data points into only two groups while and K-means clustering divided into 3 groups

In [9]:

```
from sklearn.cluster import KMeans # must use sklean . cluster
kmeans = KMeans(n_clusters = 3)
# fit kmeans object to data df_1
kmeans.fit( sample_df )
# print location of clusters learned by kmeans object
print(kmeans.cluster_centers_ )
# save new clusters for chart
y_km = kmeans.fit_predict (sample_df)
# Here you can predict using different data
```

```
[[9.86428571 9.77857143 9.94285714 9.97142857 9.80714286 9.75714286]
 [8.         3.33333333 9.         6.33333333 8.66666667 5.        ]
 [8.85714286 8.71428571 7.71428571 8.42857143 8.57142857 8.14285714]]
```

Q1.3 Reviews4Cluste

In [ ]:

```
sample_df['review_scores_accuracy'].mode
sample_df['review_scores_accuracy'].value_counts()
```

In [20]:

```
read_hdf = pd.HDFStore('Reviews4Cluster.h5')
read_hdf_prop = read_hdf['df_review_cluster']
read_hdf.close ()
```

In [23]:

```
read_hdf_prop['review_scores_accuracy'].value_counts()
```

Out[23]:

```
10.0    21
9.0      6
8.0      1
Name: review_scores_accuracy, dtype: int64
```

In [22]:

```
read_hdf_prop.value_counts ()
```

Out[22]:

```
review_scores_accuracy  review_scores_cleanliness  review_scores_checkin
review_scores_communication  review_scores_location  review_scores_value
10.0                    10.0                       10.0
10.0                         10.0                    10.0
1

9.0                    1
9.0                     9.0                        10.0
10.0                         10.0                    10.0
1
                        10.0                       9.0
10.0                         10.0                    10.0
1
                                                   10.0
9.0                          10.0                    10.0
1

10.0                         9.0                     10.0
1

10.0                    9.0                    1

10.0                   1
10.0                    8.0                        10.0
10.0                         10.0                    10.0
1
                        9.0                        9.0
10.0                         10.0                    10.0
1
                                                   10.0
9.0                          10.0                    10.0
1

10.0                         9.0                     10.0
1

10.0                    9.0                    1

10.0                   1
                        10.0                       8.0
10.0                         10.0                    10.0
1
                                                   9.0
9.0                          10.0                    10.0
1

10.0                         9.0                     10.0
1

10.0                    9.0                    1

10.0                   1
                                                   10.0
8.0                          10.0                    10.0
1

9.0                          9.0                     10.0
1
```

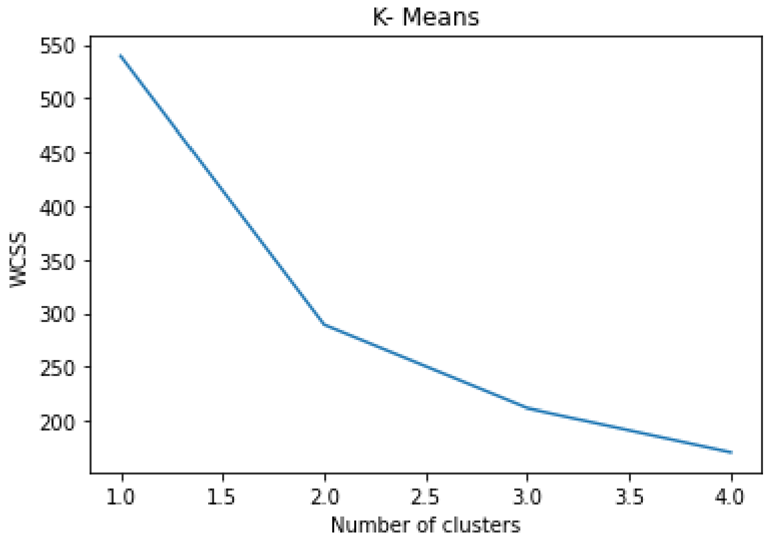
```
10.0                    9.0                   1

10.0                   1

10.0                         8.0                        10.0
1

9.0                     9.0                   1

10.0                   1

10.0                    8.0                   1
8.0                     10.0                    10.0
10.0                         10.0                       10.0
1
dtype: int64
```

Q1.4 Reviews4Cluste

As per the b elow graph of wcss the value at 2 changes shrply which suggests that 2 clusters are more appropriate than 3

In [36]:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
wcss = []
for i in range (1, 5) :
    kmeans = KMeans (n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(sample_df)
    wcss.append(kmeans.inertia_)
plt.plot ( range (1, 5) , wcss )
plt.title ('K- Means')
plt.xlabel ('Number of clusters')
plt.ylabel ('WCSS')
plt.show ()
```



2. Traffic Volumes:

Based on the comparision of below parameter model 1 is better than model 2, because the r sq of m1 is between and error is also less

---------------- | M1 | M2

R square | 0.756 | 0.753

Adjusted R | 0.755 | 0.753

RMSE | 815.13 | 824.68

In [39]:

```
import statsmodels.formula.api as smf

Metro_Interstate_Traffic_AMPeak_cleaned_df = pd.read_csv("C:/Users/satya/OneDrive - Tex
as State University/ShareKnowledge/Courses/QMST5336-ANA/Assignment/2/Metro_Interstate_T
raffic_AMPeak_cleaned.csv",header = 0, delimiter = ",")

m1 = smf.ols(formula = 'traffic_volume ~ weekend + clouds_all + temp + snow_1h', data =
Metro_Interstate_Traffic_AMPeak_cleaned_df)
m2 = smf.ols(formula = 'traffic_volume ~ weekend + clouds_all + snow_1h', data = Metro_
Interstate_Traffic_AMPeak_cleaned_df)
```

In [40]:

```
result_formula = m1.fit()
result_formula.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	traffic_volume	**R-squared:**	0.756
Model:	OLS	**Adj. R-squared:**	0.755
Method:	Least Squares	**F-statistic:**	851.3
Date:	Sun, 04 Apr 2021	**Prob (F-statistic):**	0.00
Time:	17:22:25	**Log-Likelihood:**	-8946.4
No. Observations:	1104	**AIC:**	1.790e+04
Df Residuals:	1099	**BIC:**	1.793e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	3557.5888	566.059	6.285	0.000	2446.910	4668.267
weekend	-3094.9508	53.293	-58.075	0.000	-3199.517	-2990.384
clouds_all	-1.0908	0.622	-1.752	0.080	-2.312	0.131
temp	7.0568	2.014	3.505	0.000	3.106	11.008
snow_1h	-2.24e+04	7187.425	-3.116	0.002	-3.65e+04	-8292.518

Omnibus:	671.357	**Durbin-Watson:**	1.364
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6830.931
Skew:	-2.688	**Prob(JB):**	0.00
Kurtosis:	13.936	**Cond. No.**	8.46e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.46e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [42]:

```
result_formula = m2.fit()
result_formula.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	traffic_volume	**R-squared:**	0.753
Model:	OLS	**Adj. R-squared:**	0.753
Method:	Least Squares	**F-statistic:**	1120.
Date:	Sun, 04 Apr 2021	**Prob (F-statistic):**	0.00
Time:	17:28:12	**Log-Likelihood:**	-8952.6
No. Observations:	1104	**AIC:**	1.791e+04
Df Residuals:	1100	**BIC:**	1.793e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	5536.3183	40.982	135.091	0.000	5455.906	5616.731
weekend	-3097.0914	53.562	-57.823	0.000	-3202.186	-2991.997
clouds_all	-1.1096	0.626	-1.774	0.076	-2.337	0.118
snow_1h	-2.326e+04	7219.912	-3.222	0.001	-3.74e+04	-9095.870

Omnibus:	674.799	**Durbin-Watson:**	1.346
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6833.272
Skew:	-2.709	**Prob(JB):**	0.00
Kurtosis:	13.917	**Cond. No.**	1.77e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.77e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [46]:

```
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import numpy as np

X1 = Metro_Interstate_Traffic_AMPeak_cleaned_df[['weekend', 'clouds_all', 'temp', 'snow_1h']]
X2 = Metro_Interstate_Traffic_AMPeak_cleaned_df[['weekend', 'clouds_all', 'snow_1h']]
y = Metro_Interstate_Traffic_AMPeak_cleaned_df.traffic_volume

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y, random_state=1)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y, random_state=1)

# Instantiate model
lm1 = LinearRegression()
lm2 = LinearRegression()

# Fit Model
lm1.fit(X1_train, y1_train)
lm2.fit(X2_train, y2_train)

# Predict
y1_pred = lm1.predict(X1_test)
y2_pred = lm2.predict(X2_test)

# RMSE
print(np.sqrt(metrics.mean_squared_error(y1_test, y1_pred)))
print(np.sqrt(metrics.mean_squared_error(y2_test, y2_pred)))
```

```
815.1257778173795
824.6769868778764
```

3. Predict Temperature:

Q3.1

In [64]:

```
Metro_Interstate_Traffic_AMPeak_weekly_df = Metro_Interstate_Traffic_AMPeak_cleaned_df[
Metro_Interstate_Traffic_AMPeak_cleaned_df['weekend'] == 0]
display(Metro_Interstate_Traffic_AMPeak_weekly_df)
```

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	dat
0	None	289.21	0.00	0.0	1	Clear	sky is clear	20 08
1	None	288.55	0.00	0.0	1	Clear	sky is clear	20 08
4	None	294.00	0.66	0.0	90	Rain	heavy intensity rain	20 08
5	None	285.84	0.00	0.0	1	Clear	sky is clear	20 08
6	None	288.81	0.00	0.0	1	Clear	sky is clear	20 08
...	...	...	...	...	...	...	...	
1097	None	287.19	0.00	0.0	75	Clouds	broken clouds	20 08
1098	None	284.49	0.00	0.0	90	Haze	haze	20 08
1099	None	279.43	0.25	0.0	75	Rain	light rain	20 08
1100	None	285.05	0.00	0.0	1	Clear	sky is clear	20 08
1101	None	278.83	0.00	0.0	1	Clear	sky is clear	20 08

784 rows × 11 columns

In [125]:

```
Metro_Interstate_Traffic_AMPeak_weekly_date_temp_df = Metro_Interstate_Traffic_AMPeak_w
eekly_df[['date','temp']]

df_1 = Metro_Interstate_Traffic_AMPeak_weekly_date_temp_df.set_index('date')
df_1.index.names = ['index']
display(df_1)
```

	temp
index	
2015-07-02	289.21
2015-07-03	288.55
2015-07-06	294.00
2015-07-07	285.84
2015-07-08	288.81
...	...
2018-09-24	287.19
2018-09-25	284.49
2018-09-26	279.43
2018-09-27	285.05
2018-09-28	278.83

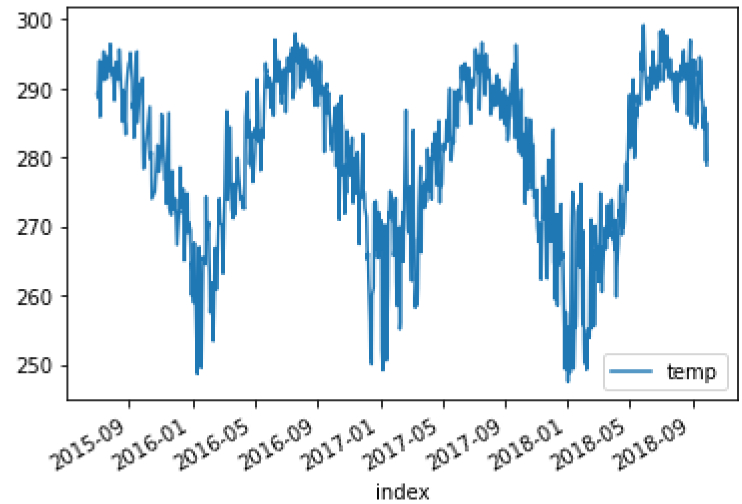
784 rows × 1 columns

In [126]:

```
df_1.plot()
#ax.set_xticklabels(xticklabels)
```

Out[126]:

```
<AxesSubplot:xlabel='index'>
```



In [127]:

```python
from statsmodels.tsa.seasonal import seasonal_decompose

# Multiplicative Decomposition
result_mul = seasonal_decompose(df_1['temp'], model= 'multiplicative')
#Setting extrapolate_trend='freq' takes care of any missing values in the trend and res
iduals at the
#beginning of the series.
# Additive Decomposition
result_add = seasonal_decompose(df_1['temp'], model= 'additive')
# Plot
plt.rcParams.update({'figure.figsize': (10,10)})
result_mul.plot().suptitle('Multiplicative Decompose', fontsize=22)
result_add.plot().suptitle('Additive Decompose', fontsize=22)
plt.show()
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-127-3d5534d0c2aa> in <module>
      2
      3 # Multiplicative Decomposition
----> 4 result_mul = seasonal_decompose(df_1['temp'], model= 'multiplicati
ve')
      5 #Setting extrapolate_trend='freq' takes care of any missing values
in the trend and residuals at the
      6 #beginning of the series.

~\anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args,
**kwargs)
    197                 else:
    198                     kwargs[new_arg_name] = new_arg_value
--> 199             return func(*args, **kwargs)
    200
    201         return cast(F, wrapper)

~\anaconda3\lib\site-packages\statsmodels\tsa\seasonal.py in seasonal_deco
mpose(x, model, filt, period, two_sided, extrapolate_trend)
    140             period = pfreq
    141         else:
--> 142             raise ValueError("You must specify a period or x must
 be a "
    143                              "pandas object with a DatetimeIndex w
ith "
    144                              "a freq not set to None")

ValueError: You must specify a period or x must be a pandas object with a
 DatetimeIndex with a freq not set to None
```

In [66]:

```
Metro_Interstate_Traffic_AMPeak_weekly_df.date = pd.to_datetime(Metro_Interstate_Traffi
c_AMPeak_weekly_df.date)
Metro_Interstate_Traffic_AMPeak_weekly_df.date.head()
```

```
C:\Users\satya\anaconda3\lib\site-packages\pandas\core\generic.py:5159: Se
ttingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

Out[66]:

```
0   2015-07-02
1   2015-07-03
4   2015-07-06
5   2015-07-07
6   2015-07-08
Name: date, dtype: datetime64[ns]
```

In [67]:

```
dd_start = np.datetime64(min(Metro_Interstate_Traffic_AMPeak_weekly_df.date.unique()),
'D')
dd_start
```

Out[67]:

```
numpy.datetime64('2015-07-02')
```

In [68]:

```
dd_end = np.datetime64(max(Metro_Interstate_Traffic_AMPeak_weekly_df.date.unique()),'D'
)
dd_end
```

Out[68]:

```
numpy.datetime64('2018-09-28')
```

In [70]:

```
dd_end - dd_start
```

Out[70]:

```
numpy.timedelta64(1184,'D')
```

In [71]:

```
dd_interval = Metro_Interstate_Traffic_AMPeak_weekly_df.date - dd_start
dd_interval.head()
```

Out[71]:

```
0    0 days
1    1 days
4    4 days
5    5 days
6    6 days
Name: date, dtype: timedelta64[ns]
```

In [72]:

```
num_days = dd_interval.dt.days
num_days.head()
```

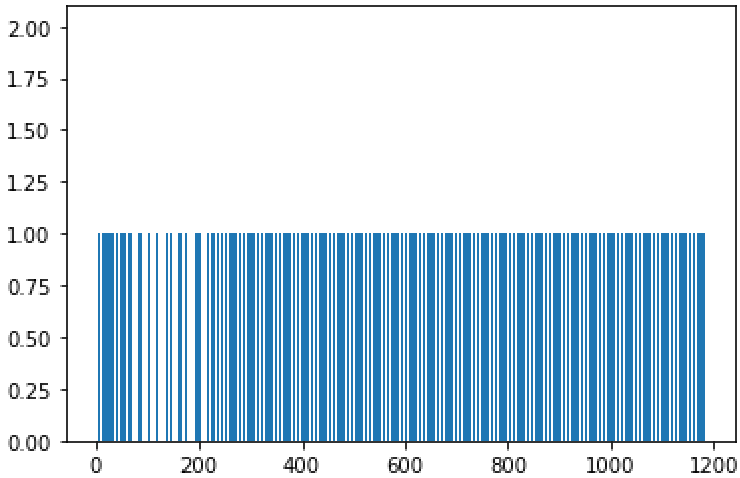
Out[72]:

```
0     0
1     1
4     4
5     5
6     6
Name: date, dtype: int64
```

In [73]:

```
n, x, p = plt.hist(num_days, bins = 1184)
print(n)
```

```
[1. 1. 0. ... 1. 1. 2.]
```



In [76]:

```
print(n)
```

```
[1. 1. 0. ... 1. 1. 2.]
```

In [77]:

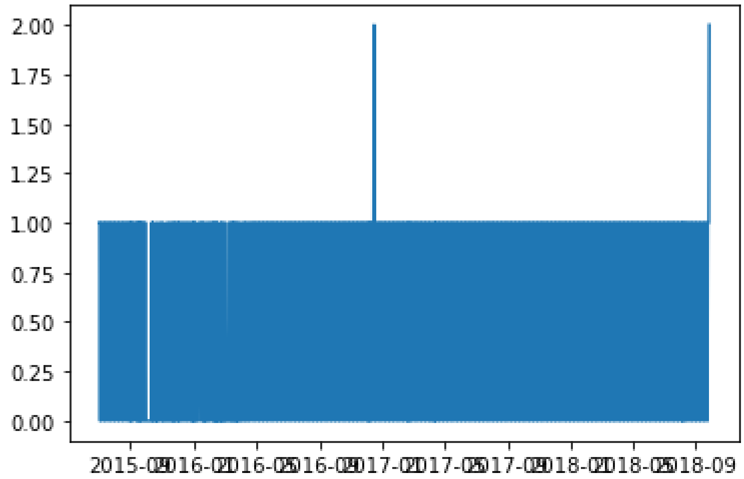
```
a_days = np.arange('2015-07-02', '2018-09-28', dtype='datetime64[D]')
```

In [78]:

```
plt.plot(a_days, n)
```

Out[78]:

```
[<matplotlib.lines.Line2D at 0x1fa4b356e80>]
```



In [104]:

```
day_review = {date: val for date, val in zip(a_days, n)}
```

In [107]:

```
df0 = pd.DataFrame.from_dict(day_review, orient = 'index', columns = ['tmps'])
df0.head()
```

Out[107]:

	tmps
2015-07-02	1.0
2015-07-03	1.0
2015-07-04	0.0
2015-07-05	0.0
2015-07-06	1.0

In [108]:

```
df1 = df0.reset_index()
df1.head()
```

Out[108]:

	index	tmps
0	2015-07-02	1.0
1	2015-07-03	1.0
2	2015-07-04	0.0
3	2015-07-05	0.0
4	2015-07-06	1.0

In [109]:

```
df1['index'] = df1['index'].astype('datetime64[ns]')
```

In [130]:

```
df2 = df1.resample('W', label='right', closed = 'right', on='index').mean()#sum()
df2.head()
```

Out[130]:

	tmps
index	
2015-07-05	0.500000
2015-07-12	0.714286
2015-07-19	0.714286
2015-07-26	0.714286
2015-08-02	0.714286

In [131]:

```
df_1.head()
```

Out[131]:

	temp
index	
2015-07-02	289.21
2015-07-03	288.55
2015-07-06	294.00
2015-07-07	285.84
2015-07-08	288.81

In [136]:

```
df_join =  df2.join(df_1,on='index',how='right')
#df_app = pd.concat([df2,Metro_Interstate_Traffic_AMPeak_weekly_df['temp']])
#df2.tmps = Metro_Interstate_Traffic_AMPeak_weekly_df['temp']
```

In [137]:

```
df_join.head()
```

Out[137]:

	index	tmps	temp
NaT	2015-07-02	NaN	289.21
NaT	2015-07-03	NaN	288.55
NaT	2015-07-06	NaN	294.00
NaT	2015-07-07	NaN	285.84
NaT	2015-07-08	NaN	288.81

In [86]:

```
Metro_Interstate_Traffic_AMPeak_weekly_date_temp_df = Metro_Interstate_Traffic_AMPeak_w
eekly_df[['date','temp']]
Metro_Interstate_Traffic_AMPeak_weekly_date_temp_df
```

Out[86]:

	date	temp
0	2015-07-02	289.21
1	2015-07-03	288.55
4	2015-07-06	294.00
5	2015-07-07	285.84
6	2015-07-08	288.81
...	...	...
1097	2018-09-24	287.19
1098	2018-09-25	284.49
1099	2018-09-26	279.43
1100	2018-09-27	285.05
1101	2018-09-28	278.83

784 rows × 2 columns

In [88]:

```
df1['date'] = df1['date'].astype('datetime64[ns]')
```

In [90]:

```

```

Out[90]:

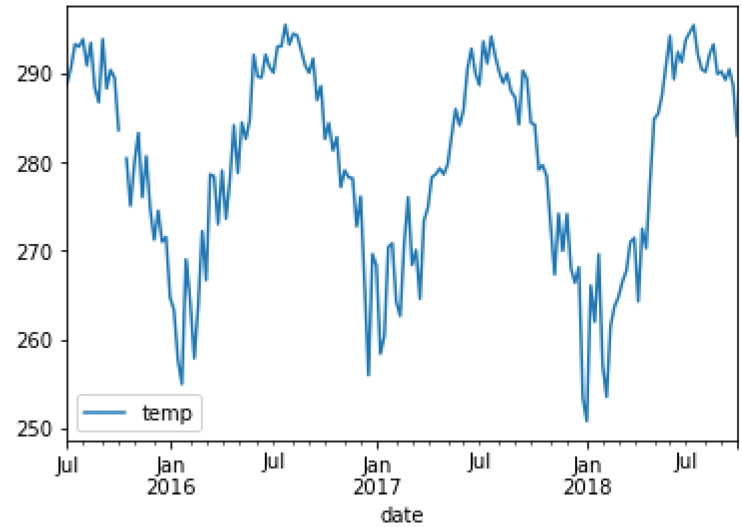
	index	temp
date		
2015-07-05	0.5	288.880
2015-07-12	6.0	290.704
2015-07-19	13.0	293.320
2015-07-26	20.0	293.020
2015-08-02	27.0	293.860

In [94]:

```
df3 = df2.drop(['index'],axis=1)
df3.plot()
```

Out[94]:

```
<AxesSubplot:xlabel='date'>
```



In [102]:

```
df4 = df3.dropna()
df4.dtypes
```

Out[102]:

```
temp    float64
dtype: object
```

In [138]:

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Multiplicative Decomposition
result_mul = seasonal_decompose(df2['tmps'], model= 'multiplicative', extrapolate_trend
= 'freq')
#Setting extrapolate_trend='freq' takes care of any missing values in the trend and res
iduals at the
#beginning of the series.
# Additive Decomposition
#result_add = seasonal_decompose(df3['temp'], model= 'additive')
# Plot
#plt.rcParams.update({'figure.figsize': (10,10)})
#result_mul.plot().suptitle('Multiplicative Decompose', fontsize=22)
#result_add.plot().suptitle('Additive Decompose', fontsize=22)
#plt.show()
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-138-fa910f22fd4d> in <module>
      2
      3 # Multiplicative Decomposition
----> 4 result_mul = seasonal_decompose(df2['tmps'], model= 'multiplicativ
e', extrapolate_trend= 'freq')
      5 #Setting extrapolate_trend='freq' takes care of any missing values
in the trend and residuals at the
      6 #beginning of the series.

~\anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args,
**kwargs)
    197                 else:
    198                     kwargs[new_arg_name] = new_arg_value
--> 199             return func(*args, **kwargs)
    200
    201         return cast(F, wrapper)

~\anaconda3\lib\site-packages\statsmodels\tsa\seasonal.py in seasonal_deco
mpose(x, model, filt, period, two_sided, extrapolate_trend)
    132     if model.startswith('m'):
    133         if np.any(x <= 0):
--> 134             raise ValueError("Multiplicative seasonality is not ap
propriate "
    135                              "for zero and negative values")
    136

ValueError: Multiplicative seasonality is not appropriate for zero and neg
ative values
```