# Energy utilization for *United States* and *England*


## Aishik Dasgupta

Linkedin : https://www.linkedin.com/in/aishik-dasgupta/

Github : https://github.com/AishikDasgupta


In [1]:
```
# importing required librabries
import pandas as pd
import seaborn as sns
import matplotlib .pyplot as plt
import numpy as np
import matplotlib.dates as mdates
import warnings
warnings.filterwarnings("ignore")
plt.style.use("fivethirtyeight")
import matplotlib.style as style
style.available
df = pd.read_csv("Energy_data_usage_USA.csv", parse_dates=["Date"], index_co
df.head(1)
```

Out[1]:

| Date | Unnamed: 0 | United States : all sectors | United States : electric utility | United States : independent power producers | United States : all commercial | United States : all industrial | New England : all sectors | New England : electric utility | Eng ele uti |
|---|---|---|---|---|---|---|---|---|---|
| 2001-01-01 | 0 | 332493 | 236467 | 82269 | 629 | 13128 | 10005 | 2467 | |

In [2]:
```
# dropping Unneeded columns
df = df.drop(["Unnamed: 0", "New England : electric utility.1"], axis = 1, )
```

In [3]:
```
# DatetimeIndex of dataset
df.index
```

Out[3]:
```
DatetimeIndex(['2001-01-01', '2001-02-01', '2001-03-01', '2001-04-01',
               '2001-05-01', '2001-06-01', '2001-07-01', '2001-08-01',
               '2001-09-01', '2001-10-01',
               ...
               '2021-06-01', '2021-07-01', '2021-08-01', '2021-09-01',
               '2021-10-01', '2021-11-01', '2021-12-01', '2022-01-01',
               '2022-02-01', '2022-03-01'],
              dtype='datetime64[ns]', name='Date', length=255, freq=None)
```

In [4]:
```
# splitting index into weeks months and year
df.index=pd.to_datetime(df.index)
df["days_of_the week"]=df.index.dayofweek
df["weeks of the year"]=df.index.week
df["months of the year"]= df.index.month
df["year"]=df.index.year
```

In [5]:
```
df.head(1)
```

Out[5]:

| Date | United States : all sectors | United States : electric utility | United States : independent power producers | United States : all commercial | United States : all industrial | New England : all sectors | New England : electric utility | New England : independent power producers | c |
|---|---|---|---|---|---|---|---|---|---|
| 2001-01-01 | 332493 | 236467 | 82269 | 629 | 13128 | 10005 | 2467 | 6959 | |

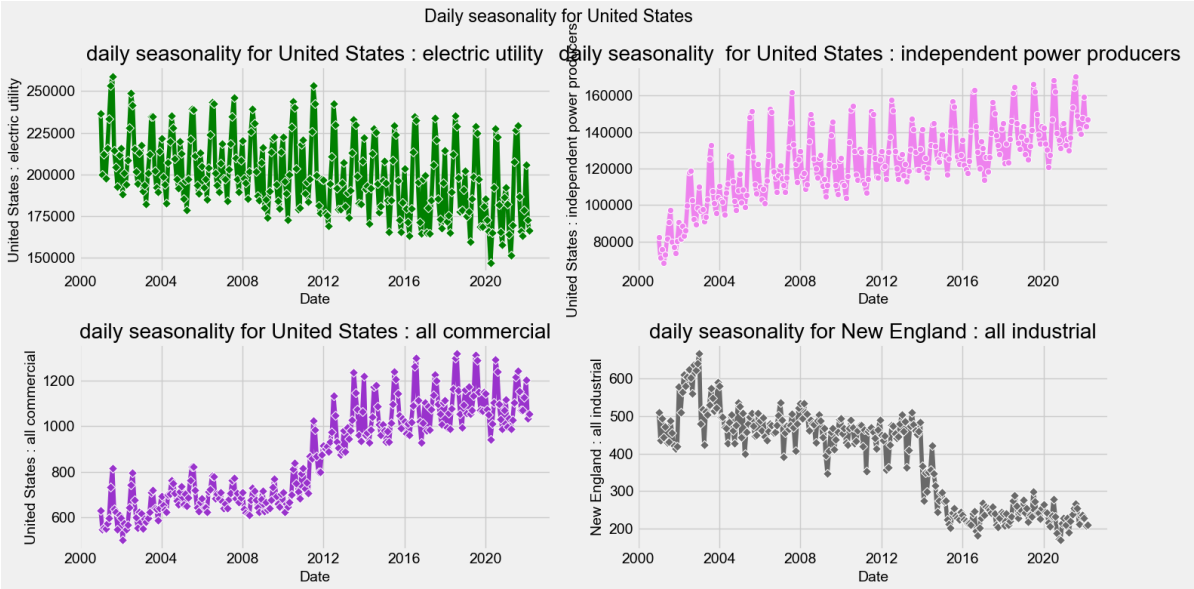
In [6]:
```
#checking data achitecture
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 255 entries, 2001-01-01 to 2022-03-01
Data columns (total 14 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   United States : all sectors                 255 non-null    int64
 1   United States : electric utility            255 non-null    int64
 2   United States : independent power producers  255 non-null    int64
 3   United States : all commercial              255 non-null    int64
 4   United States : all industrial              255 non-null    int64
 5   New England : all sectors                   255 non-null    int64
 6   New England : electric utility              255 non-null    int64
 7   New England : independent power producers   255 non-null    int64
 8   New England : all commercial                255 non-null    int64
 9   New England : all industrial                255 non-null    int64
 10  days_of_the week                            255 non-null    int64
 11  weeks of the year                           255 non-null    int64
 12  months of the year                          255 non-null    int64
 13  year                                        255 non-null    int64
dtypes: int64(14)
memory usage: 29.9 KB
```

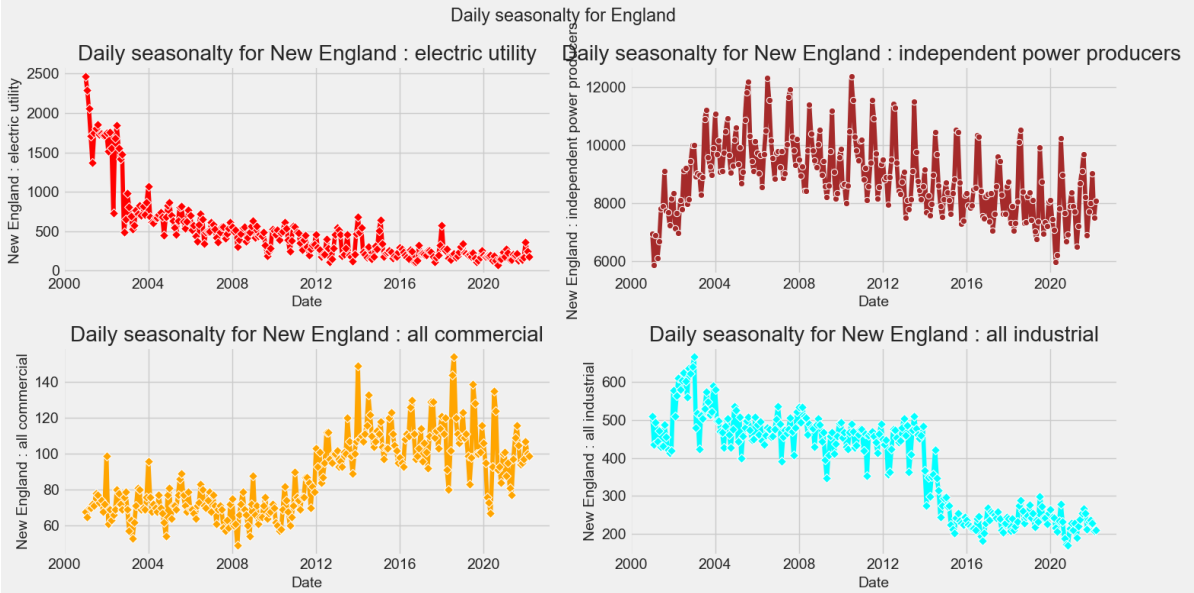
# Daily Seasonality

- the seasonality of our data with lineplots, using seaborn's lineplot() function to group the data by different time periods and display the distributions for each group. We'll first group the data by month, to visualize yearly seasonality.

In [7]:
```
# plot suplots for daily seasonality in united states
plt.style.use("fivethirtyeight")
fig,axs = plt.subplots (nrows=2, ncols=2, sharey=False ,figsize= (15,8))
fig.suptitle("Daily seasonality for United States")
g= sns.lineplot(df["United States : electric utility"],ax=axs[0,0],marker="D
g.set_title("daily seasonality for United States : electric utility")
g= sns.lineplot(df["United States : independent power producers"], ax=axs[0,
g.set_title("daily seasonality  for United States : independent power produc
g= sns.lineplot(df["United States : all commercial"],ax=axs[1,0],marker="D",
g.set_title("daily seasonality for United States : all commercial")
g= sns.lineplot( df["New England : all industrial"],ax=axs[1,1],marker="D",c
g.set_title("daily seasonality for New England : all industrial")
g = sns.set_style("darkgrid")
fig.tight_layout()
plt.show()
```



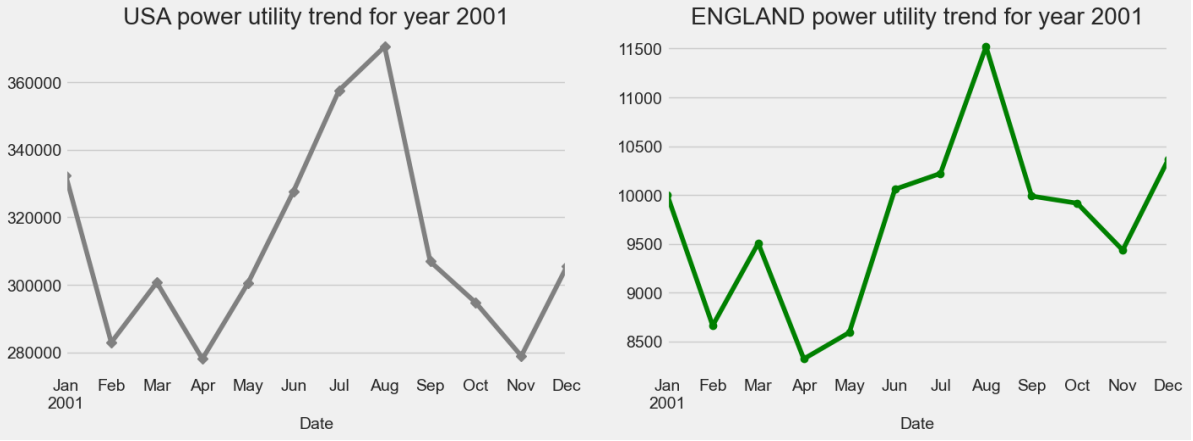

In [8]:
```
# plot suplots for daily seasonality in England
style.use("fivethirtyeight")
fig,axs = plt.subplots (nrows=2, ncols=2, sharey=False ,figsize= (15,8))
fig.suptitle("Daily seasonalty for England")
g= sns.lineplot(df["New England : electric utility"],ax=axs[0,0],marker="D",
g.set_title("Daily seasonalty for New England : electric utility")
g= sns.lineplot(df["New England : independent power producers"], ax=axs[0,1]
g.set_title(  "Daily seasonalty for New England : independent power producer
g= sns.lineplot(df["New England : all commercial"],ax=axs[1,0],marker="D",co
g.set_title("Daily seasonalty for New England : all commercial")
g= sns.lineplot( df["New England : all industrial"],ax=axs[1,1],marker="D",c
g.set_title("Daily seasonalty for New England : all industrial")
g = sns.set_style("darkgrid")
fig.tight_layout()
plt.show()
```




# loc in timeseries

loc function is highly beneficial most espcially in timeseries analysis. its helps in manipulation of time series data enabling tasking subsetting filtering and resampling with precision

In [9]:
```
# plot  monthly subplots for single specific year for USA and England
style.use("fivethirtyeight")
f,ax = plt.subplots (1,2,figsize = (15 ,5))
df.loc["2001", "United States : all sectors"].plot(ax = ax [0],marker="D",co
df.loc["2001", "New England : all sectors"].plot(ax = ax [1],marker="o",colo

plt.show ()
```



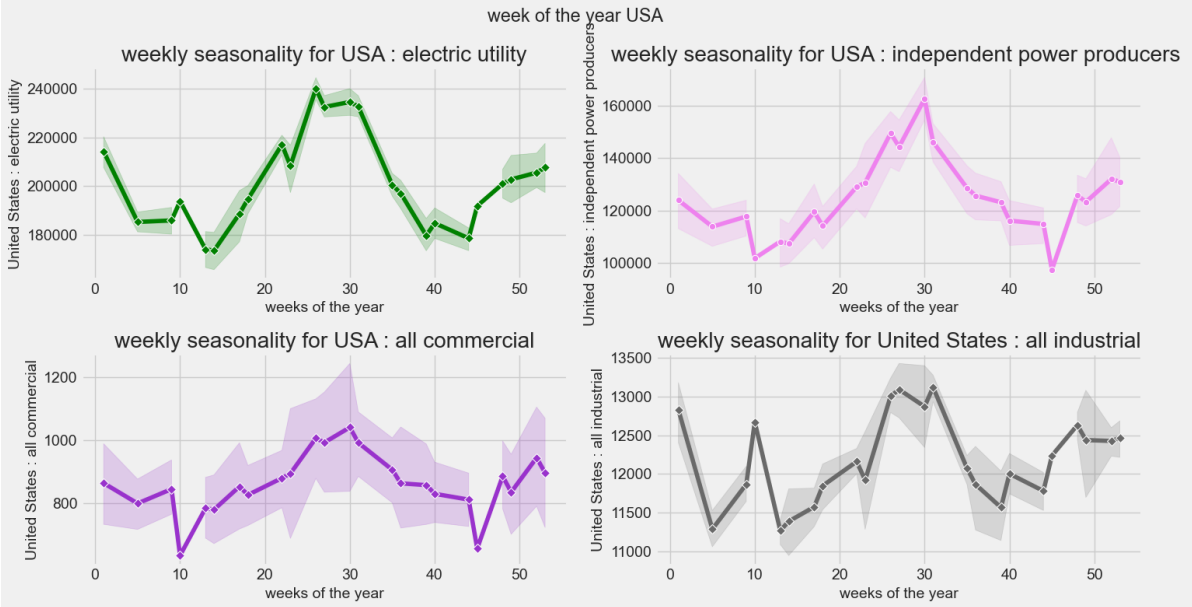

In [10]:
```
# show data archetecture
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 255 entries, 2001-01-01 to 2022-03-01
Data columns (total 14 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   United States : all sectors                 255 non-null    int64
 1   United States : electric utility            255 non-null    int64
 2   United States : independent power producers  255 non-null    int64
 3   United States : all commercial              255 non-null    int64
 4   United States : all industrial              255 non-null    int64
 5   New England : all sectors                   255 non-null    int64
 6   New England : electric utility              255 non-null    int64
 7   New England : independent power producers   255 non-null    int64
 8   New England : all commercial                255 non-null    int64
 9   New England : all industrial                255 non-null    int64
 10  days_of_the week                            255 non-null    int64
 11  weeks of the year                           255 non-null    int64
 12  months of the year                          255 non-null    int64
 13  year                                        255 non-null    int64
dtypes: int64(14)
memory usage: 29.9 KB
```

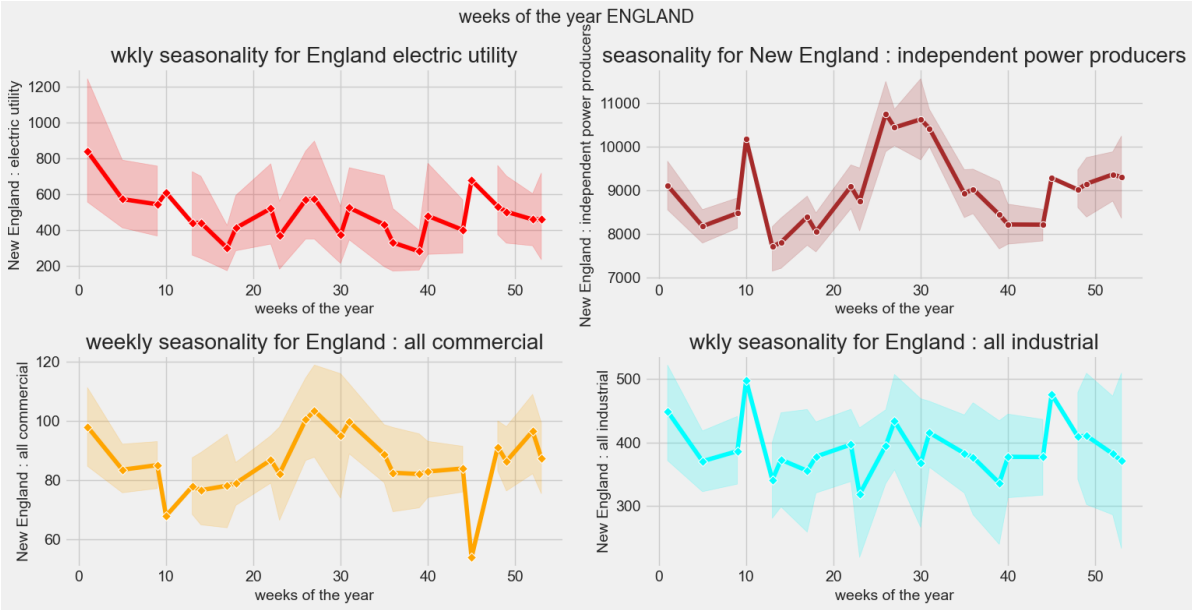
# weekly seasonality

In [11]:
```
# plot suplots for weekly seasonality in united states
style.use("fivethirtyeight")
fig,axs = plt.subplots (nrows=2, ncols=2, sharey=False ,figsize= (15,8))
fig.suptitle('week of the year USA')
g= sns.lineplot(x=df["weeks of the year"], y=df["United States : electric ut
g.set_title("weekly seasonality for USA : electric utility")
g= sns.lineplot(x=df["weeks of the year"],y=df["United States : independent
g.set_title( "weekly seasonality for USA : independent power producers ")
g= sns.lineplot(x=df["weeks of the year"],y=df["United States : all commerci
g.set_title("weekly seasonality for USA : all commercial")
g= sns.lineplot( x=df["weeks of the year"],y=df["United States : all industr
g.set_title("weekly seasonality for United States : all industrial")
```

```
g = sns.set_style("darkgrid")
fig.tight_layout()
plt.show()
```

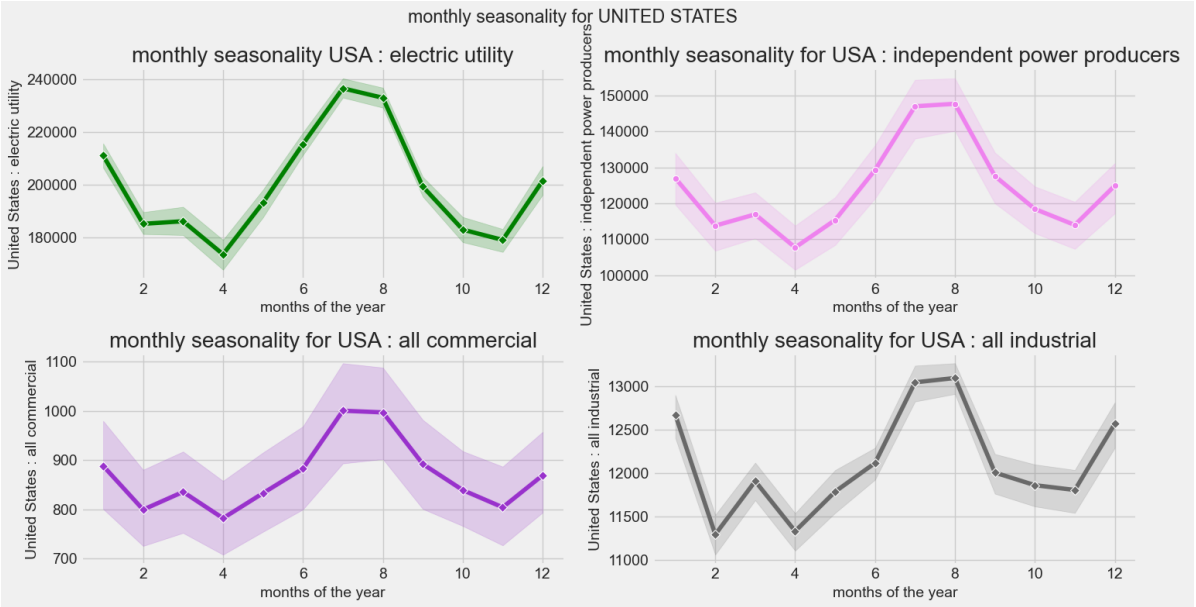



In [12]:
```
# plot suplots for weekly seasonality in england
style.use("fivethirtyeight")
fig,axs = plt.subplots (nrows=2, ncols=2, sharey=False ,figsize= (15,8))
fig.suptitle("weeks of the year ENGLAND")
g= sns.lineplot(x=df["weeks of the year"], y=df["New England : electric util
g.set_title("wkly seasonality for England electric utility")
g= sns.lineplot(x=df["weeks of the year"],y=df["New England : independent po
g.set_title(  "seasonality for New England : independent power producers")
g= sns.lineplot(x=df["weeks of the year"],y=df["New England : all commercial
g.set_title("weekly seasonality for England : all commercial")
g= sns.lineplot(x=df["weeks of the year"],y=df["New England : all industrial
g.set_title("wkly seasonality for England : all industrial")
fig.tight_layout()
plt.show()
```

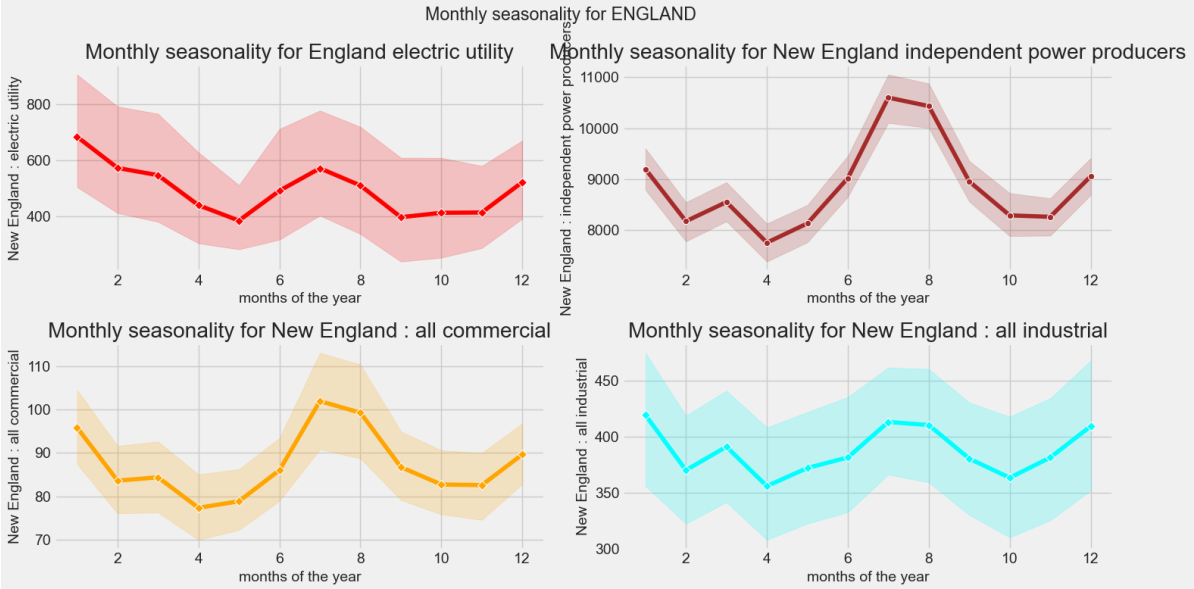



# monthly seasonality

In [13]:
```
# plot suplots for monthly seasonality in united states
style.use("fivethirtyeight")
fig,axs = plt.subplots (nrows=2, ncols=2, sharey=False ,figsize= (15,8))
fig.suptitle("monthly seasonality for UNITED STATES")
g= sns.lineplot(x=df["months of the year"], y=df["United States : electric u
g.set_title("monthly seasonality USA : electric utility")
g= sns.lineplot(x=df["months of the year"],y=df["United States : independent
g.set_title(  "monthly seasonality for USA : independent power producers ")
g= sns.lineplot(x=df["months of the year"],y=df["United States : all commerc
g.set_title("monthly seasonality for USA : all commercial")
g= sns.lineplot( x=df["months of the year"],y=df["United States : all indust
g.set_title("monthly seasonality for USA : all industrial")
g = sns.set_style("darkgrid")
fig.tight_layout()
plt.show()
```




In [14]:
```
# plot suplots for monthly seasonality in united states
style.use("fivethirtyeight")
fig,axs = plt.subplots (nrows=2, ncols=2, sharey=False ,figsize= (15,8))
fig.suptitle("Monthly seasonality for ENGLAND")
g= sns.lineplot(x=df["months of the year"], y=df["New England : electric uti
g.set_title("Monthly seasonality for England electric utility")
g= sns.lineplot(x=df["months of the year"],y=df["New England : independent p
g.set_title("Monthly seasonality for New England independent power producers
g= sns.lineplot(x=df["months of the year"],y=df["New England : all commercia
g.set_title("Monthly seasonality for New England : all commercial")
g= sns.lineplot(x=df["months of the year"],y=df["New England : all industria
g.set_title("Monthly seasonality for New England : all industrial")
g = sns.set_style("darkgrid")
fig.tight_layout()
plt.show()
```

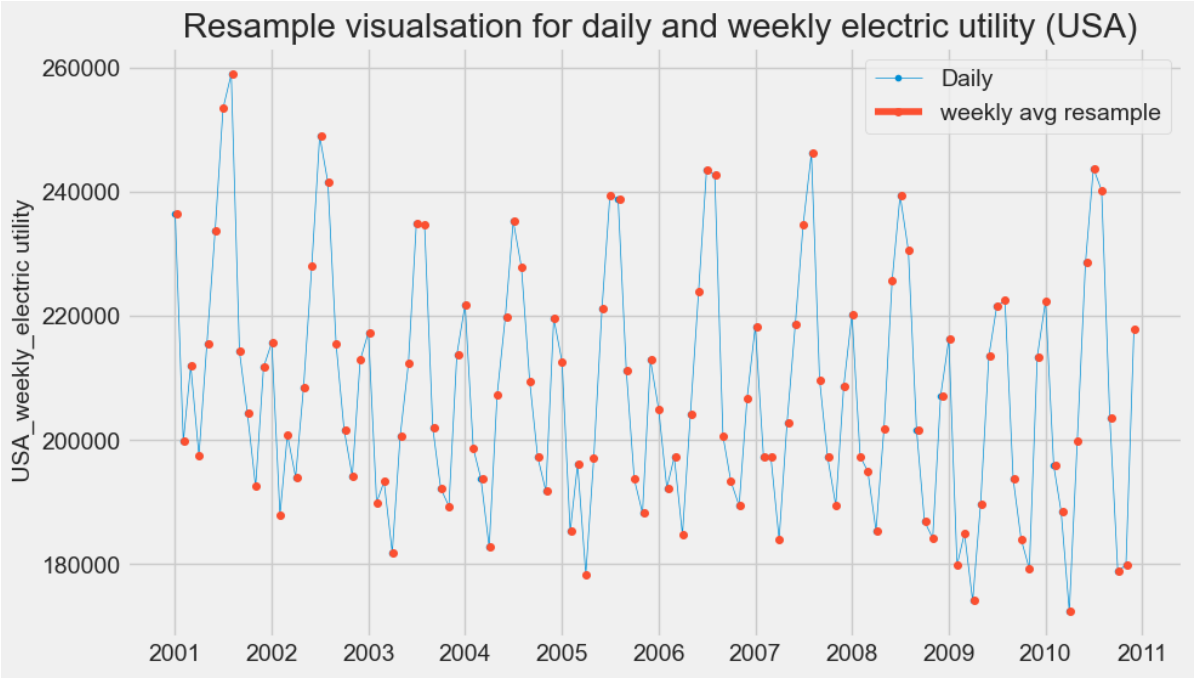



# Resampling

- The resampling method transforms time series data that are occurring in irregular time intervals into equispaced data. The method is also useful for transforming equispaced data from one frequency level to another (for example, minutes to hours ,hours to days,days to weeks,weeks to months and months to year).

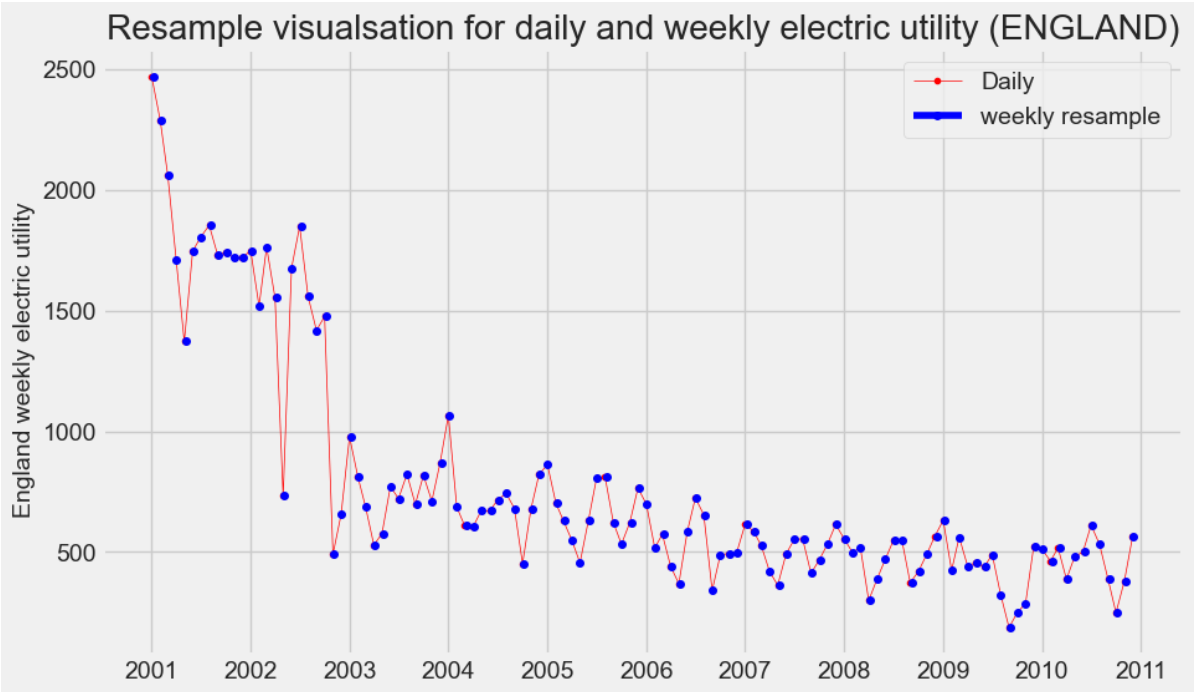
## ploting weekly time series over a 10 year period(rsp)

In [15]:
```
# plot style
style.use("fivethirtyeight")
#  create  new data
cols_USA = ["United States : electric utility","United States : independent
       ,"United States : all commercial"]
# resample data to weekly frequency
df_weekly_rsp_USA =  df[cols_USA].resample("w").mean() # average power utili
#create a timeline
start,end =  "2001", "2010"
# plot the resampled data
fig, ax= plt.subplots(figsize=(10 ,6))
ax.plot(df.loc[start:end,"United States : electric utility"],marker=".",line
ax.plot(df_weekly_rsp_USA.loc[start : end, "United States : electric utility
ax.set_ylabel(" USA_weekly_electric utility")
ax.set_title(" Resample visualsation for daily and weekly electric utility (
ax.legend()
plt.show()
```



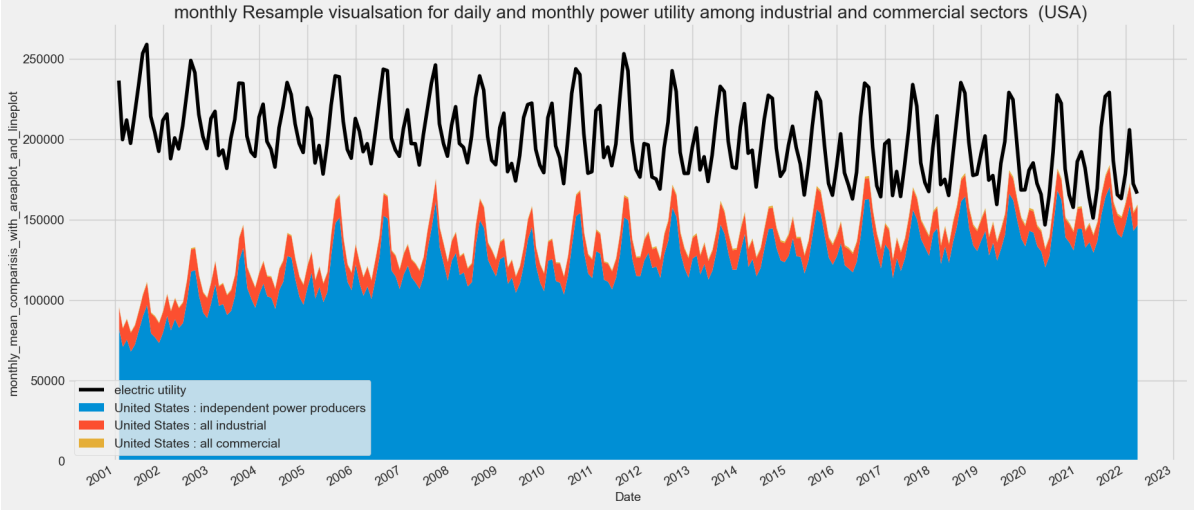

In [16]:

```
# plot style
style.use("fivethirtyeight")
#  create  new data
cols_ENG = ["New England : electric utility","New England : independent powe
       ,"New England : all commercial"]
# resample data to weekly frequency
df_weekly_avg_ENG =  df[cols_ENG].resample("w").mean() # average power utili
#create a timeline
start,end =  "2001", "2010"
# plot the resampled data
fig, ax= plt.subplots(figsize=(10 ,6))
ax.plot(df.loc[start:end,"New England : electric utility"],marker=".",linest
ax.plot(df_weekly_avg_ENG.loc[start : end, "New England : electric utility"]
ax.set_ylabel("England weekly electric utility  ")
ax.set_title("Resample visualsation for daily and weekly electric utility (E
ax.legend()
plt.show()
```

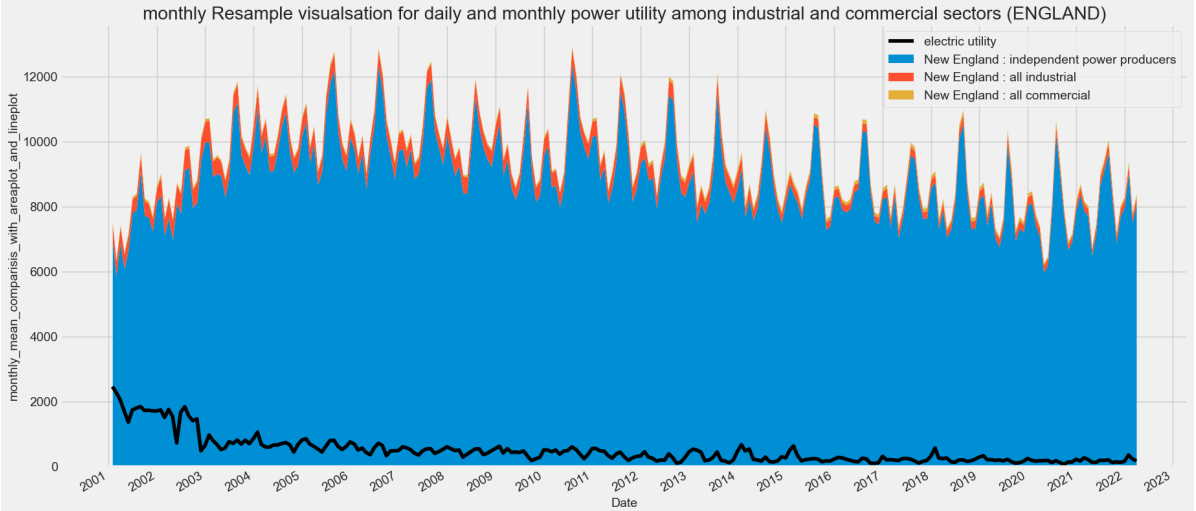



# monthly(rsp)

In [17]:
```
# plot style
style.use("fivethirtyeight")
#  create new data
df_monthly_USA= df[cols_USA].resample("M").mean() # average power utility po
# plot the resampled data
fig, ax = plt.subplots(figsize = (20,10))
ax.plot(df_monthly_USA["United States : electric utility"],color="black",lab
df_monthly_USA[["United States : independent power producers","United States
ax.xaxis.set_major_locator(mdates.YearLocator())
ax.legend()
ax.set_title(" monthly Resample visualsation for daily and monthly power uti
ax.set_ylabel("monthly_mean_comparisis_with_areaplot_and_lineplot")
plt.show()
```



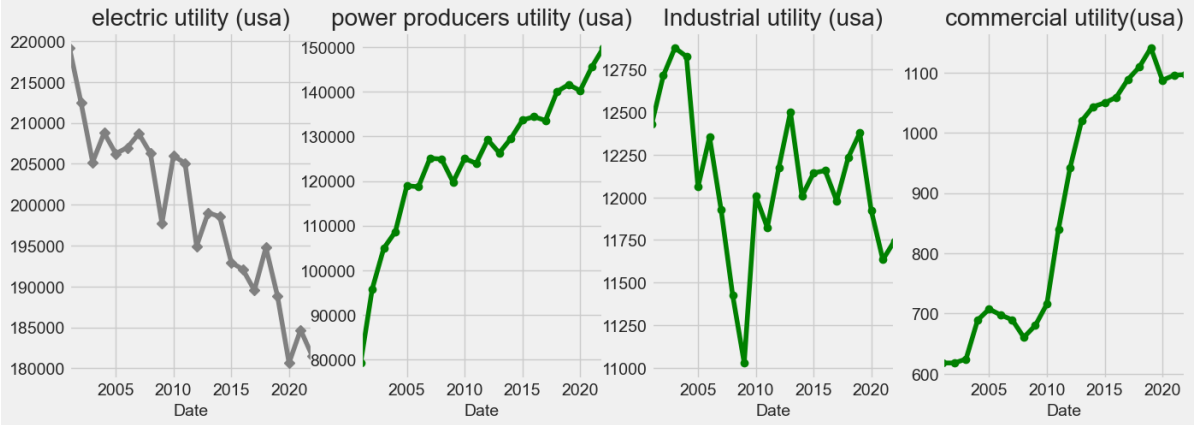

In [18]:
```
# plot style
style.use("fivethirtyeight")
#  create new data
df_monthly_ENG= df[cols_ENG].resample("M").mean()# average power utility pow
# plot the resampled data
fig, ax = plt.subplots(figsize = (20,10))
ax.plot(df_monthly_ENG["New England : electric utility"],color="black",label
df_monthly_ENG[["New England : independent power producers","New England : a
ax.xaxis.set_major_locator(mdates.YearLocator())
ax.legend()
ax.set_title("monthly Resample visualsation for daily and monthly power util
ax.set_ylabel("monthly_mean_comparisis_with_areaplot_and_lineplot")
plt.show()
```



# yearly (rsp)

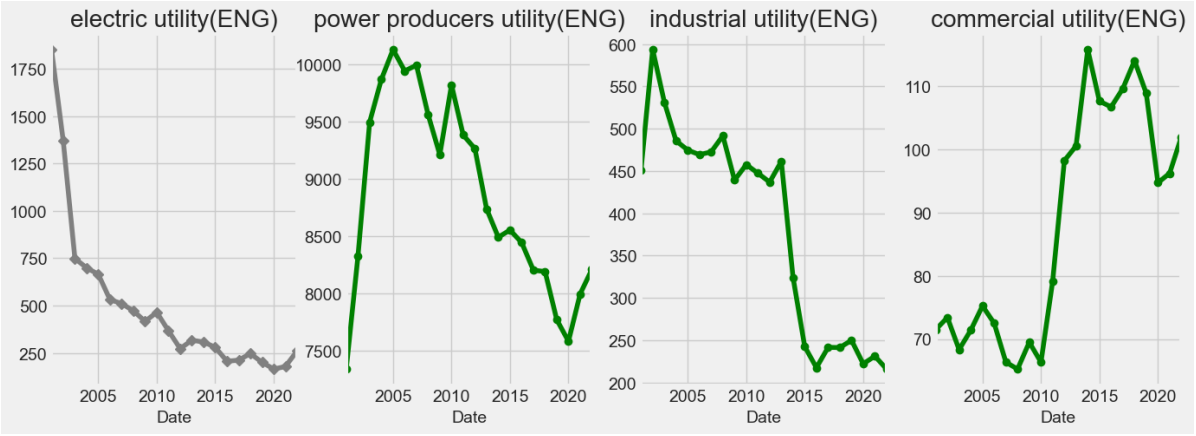
## USA Annual Resample in a space of 5 year interval

In [19]:
```
# plot style
style.use("fivethirtyeight")
#  create new data
df_annual_USA= df[cols_USA].resample("A").mean()# average power utility powe
# plot the resampled data
f,ax = plt.subplots (1,4,figsize = (15 ,5))
df_annual_USA.loc["2001" :, "United States : electric utility"].plot(ax = ax
df_annual_USA.loc["2001" :, "United States : independent power producers"].p
df_annual_USA.loc["2001" :, "United States : all industrial"].plot(ax = ax |
df_annual_USA.loc["2001" :, "United States : all commercial"].plot(ax = ax |


plt.show ()
```




## ENGLAND Annual Resample in a space of 5 year interval

In [20]:
```
# plot style
style.use("fivethirtyeight")
#  create new data
df_annual_ENG= df[cols_ENG].resample("A").mean()
# plot the resampled data
f,ax = plt.subplots (1,4,figsize = (15 ,5))
df_annual_ENG.loc["2001" :, "New England : electric utility"].plot(ax = ax |
df_annual_ENG.loc["2001" :, "New England : independent power producers"].plo
df_annual_ENG.loc["2001" :, "New England : all industrial"].plot(ax = ax [2]
df_annual_ENG.loc["2001" :, "New England : all commercial"].plot(ax = ax [3]

plt.show ()
```

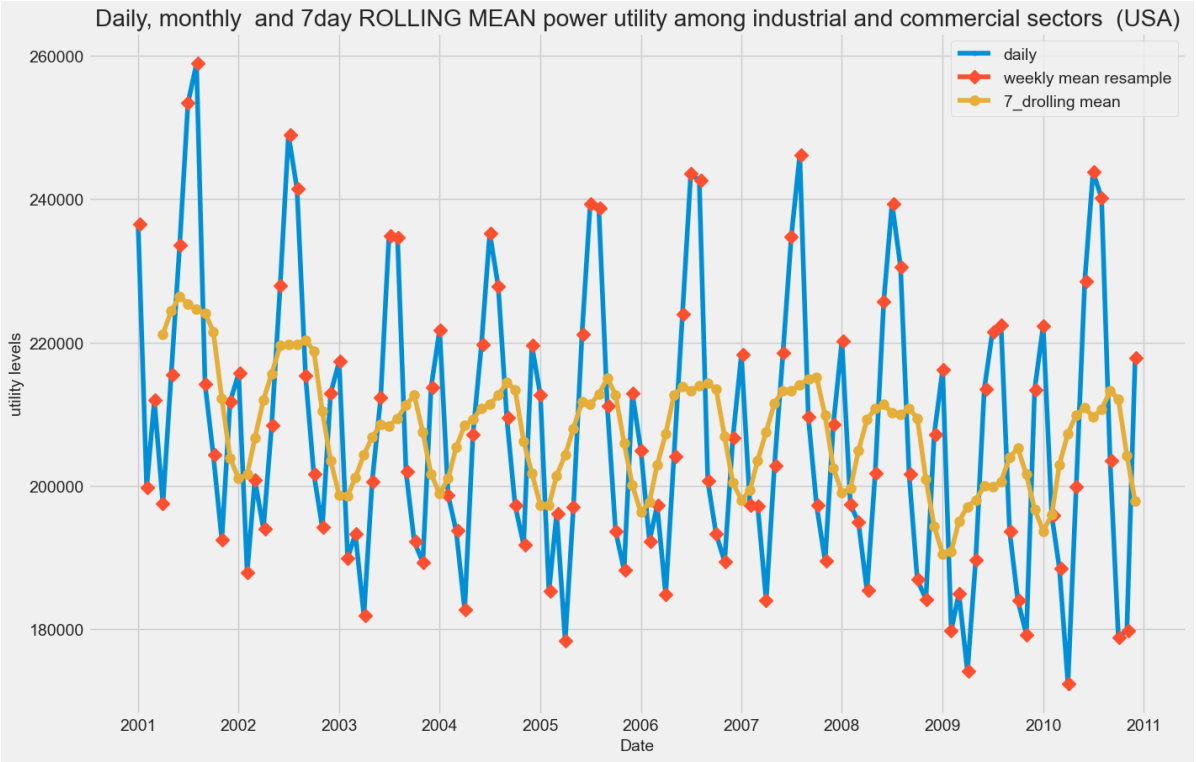



# TRENDS ( rolling mean) 7 days

- Rolling Mean, also known as moving average, calculates the mean of the values within the window of values.
- rolling average = sum of data over time / time period.

## USA 7 day Rolling mean comparision between daily, and weekly mean resample

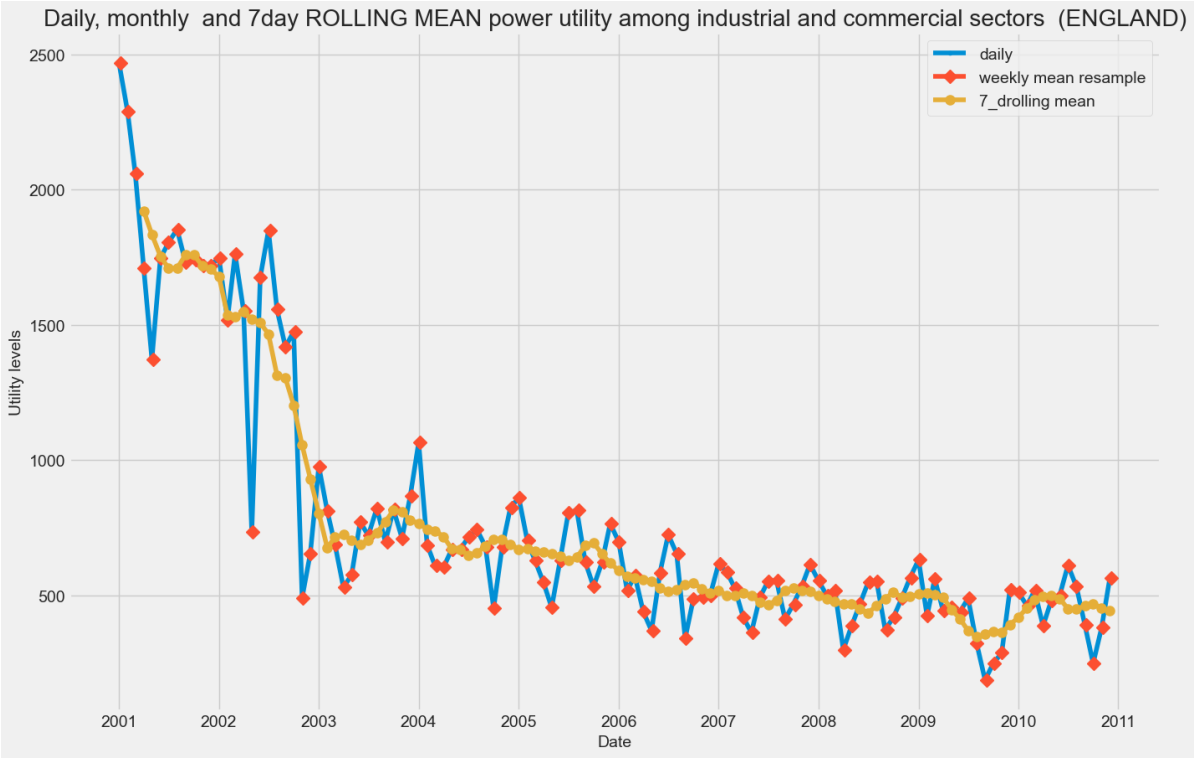
In [21]:
```
# plot style
style.use("fivethirtyeight")
#  create new rolling data
dfrolling_USA= df[cols_USA].rolling(7, center = True).mean()
# create timeline
start , end= "2001", "2010"
# plot rolling data
fig, ax = plt.subplots(figsize = (15,10))
ax.plot(df.loc[start:end ,"United States : electric utility"], marker=".",li
ax.plot(df_weekly_rsp_USA.loc[start : end,"United States : electric utility"
ax.plot(dfrolling_USA.loc[start : end,"United States : electric utility"],ma
ax.set_ylabel("utility levels")
ax.set_xlabel("Date")
ax.set_title("Daily, monthly  and 7day ROLLING MEAN power utility among indu
ax.legend()
plt.show()
```




# ENGLAND

ENGLAND 7 day Rolling mean comparision between daily, and weekly mean resample

In [22]:
```
# plot style
style.use("fivethirtyeight")
#  create new rolling data
dfrolling_ENG= df[cols_ENG].rolling(7, center = True).mean()
# create timeline
start , end= "2001", "2010"
# plot rolling data
fig, ax = plt.subplots(figsize = (15,10))
ax.plot(df.loc[start:end ,"New England : electric utility"], marker=".",line
ax.plot(df_weekly_avg_ENG.loc[start : end,"New England : electric utility"],
ax.plot(dfrolling_ENG.loc[start : end,"New England : electric utility"],mark
ax.set_ylabel("Utility levels")
ax.set_xlabel("Date")
ax.set_title("Daily, monthly  and 7day ROLLING MEAN power utility among indu
ax.legend()
plt.show()
```

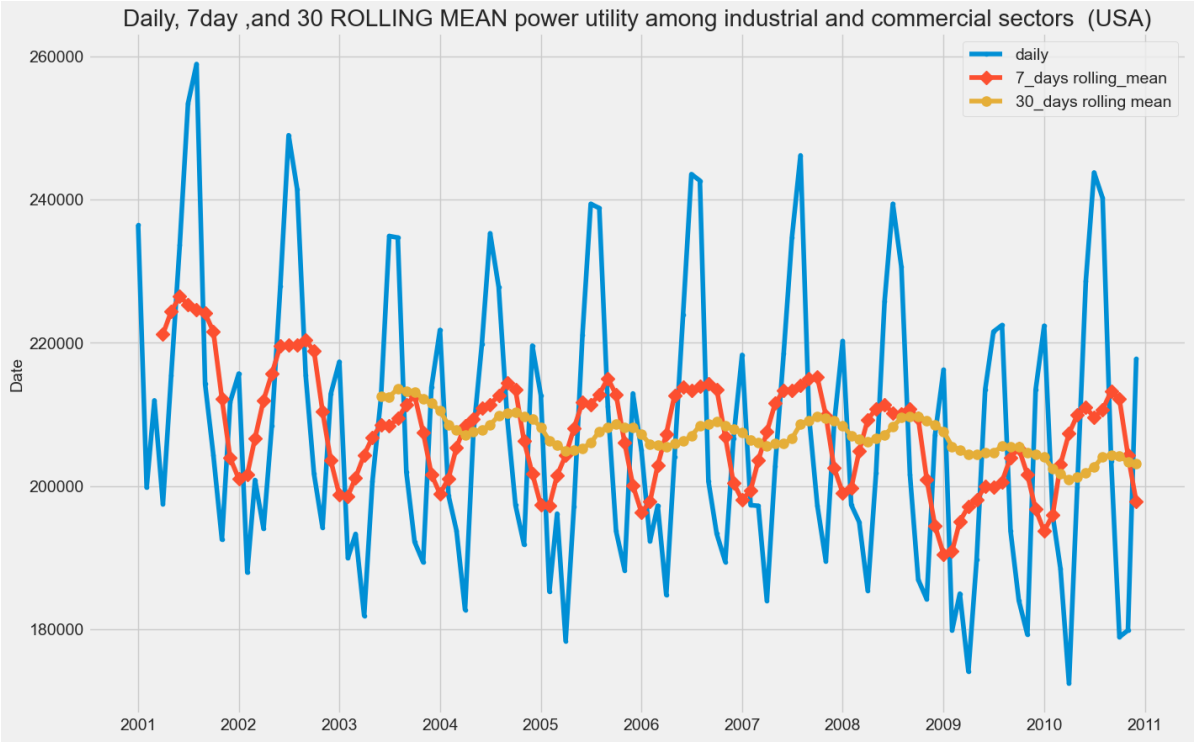



# TRENDS 30 days

# USA

USA 30 day Rolling mean comparision between daily, and 7day_rolling

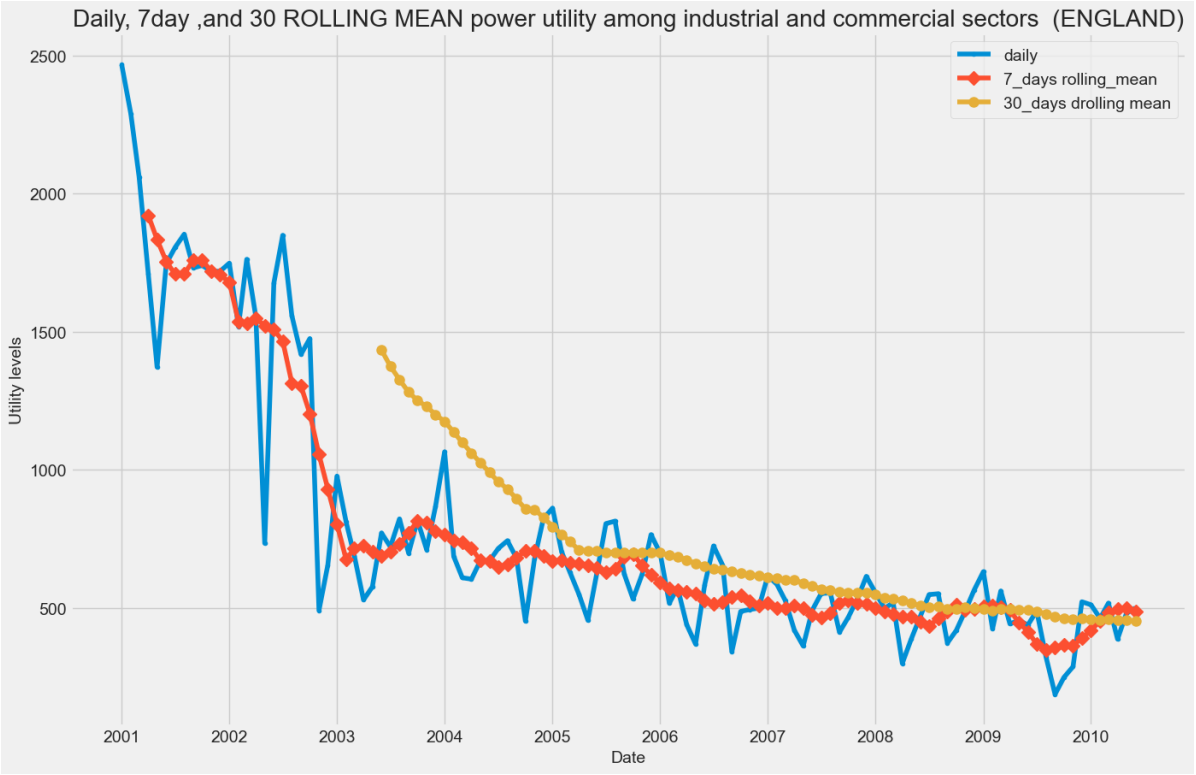
In [23]:
```
# plot style
style.use("fivethirtyeight")
#  create new rolling data
dfrollmd_mo_USA= df[cols_USA].rolling(window= 30).mean()
# create timeline
start , end= "2001", "2010"
# plot rolling data
fig, ax = plt.subplots(figsize = (15,10))
ax.plot(df.loc[start:end ,"United States : electric utility"], marker=".",li
ax.plot(dfrolling_USA.loc[start : end,"United States : electric utility"],ma
ax.plot(dfrollmd_mo_USA.loc[start : end,"United States : electric utility"],
ax.set_ylabel("utility levels")
ax.set_ylabel("Date")
ax.set_title("Daily, 7day ,and 30 ROLLING MEAN power utility among industria
ax.legend()
plt.show()
```




# ENGLAND

ENGLAND 30 day Rolling mean comparision between daily, and 7_day rolling mean

In [24]:
```
# plot style
style.use('fivethirtyeight')
#  create new rolling data
dfrollmd_mo_ENG= df[cols_ENG].rolling(window=30).mean()
# create timeline
start , end= "2001", "2010-06"
# plot rolling data
fig, ax = plt.subplots(figsize = (15,10))
ax.plot(df.loc[start:end ,"New England : electric utility"], marker=".",line
ax.plot(dfrolling_ENG.loc[start : end,"New England : electric utility"],mark
ax.plot(dfrollmd_mo_ENG.loc[start : end,"New England : electric utility"],ma
ax.set_ylabel("Utility levels")
ax.set_title("Daily, 7day ,and 30 ROLLING MEAN power utility among industria
ax.set_xlabel("Date")
ax.legend()
plt.show()
```




In [25]:
```
df.head(1)
```

Out[25]:

| Date | United States : all sectors | United States : electric utility | United States : independent power producers | United States : all commercial | United States : all industrial | New England : all sectors | New England : electric utility | New England : independent power producers | c |
|---|---|---|---|---|---|---|---|---|---|
| 2001-01-01 | 332493 | 236467 | 82269 | 629 | 13128 | 10005 | 2467 | 6959 | |

In [26]:
```
# rename column names of dataset
df_new=df.rename(columns={df.columns[0] : "United_States_all_sectors" , df.c
                  df.columns[3] : "United_States_all_commercial" ,df.columns
                  df.columns[5]: "New_England_all_sectors",df.columns[6] :"N
                  df.columns[7] :"New_England_electric_utility",df.columns[8
                  df.columns[9] :"New_England_all_commercial",df.columns[10]
```

In [27]:
```
df_new.head(1)
```

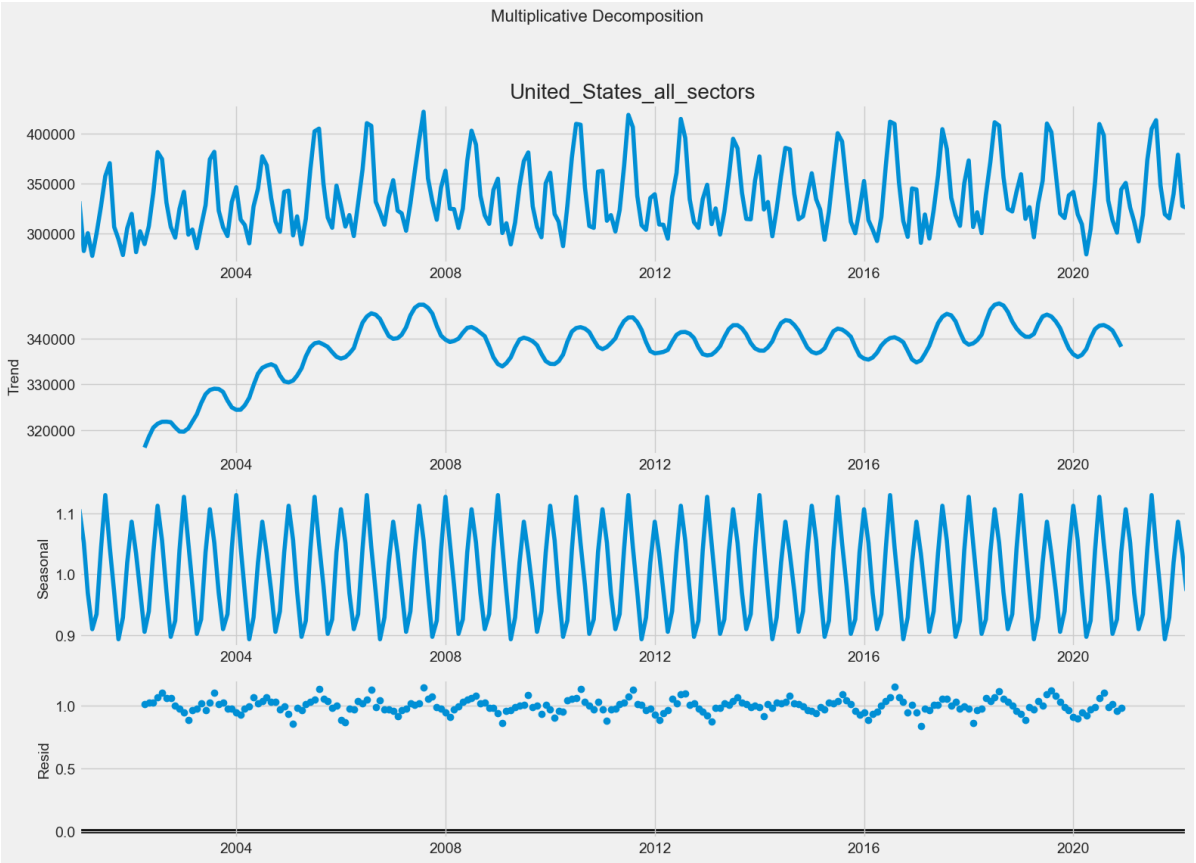
Out[27]:

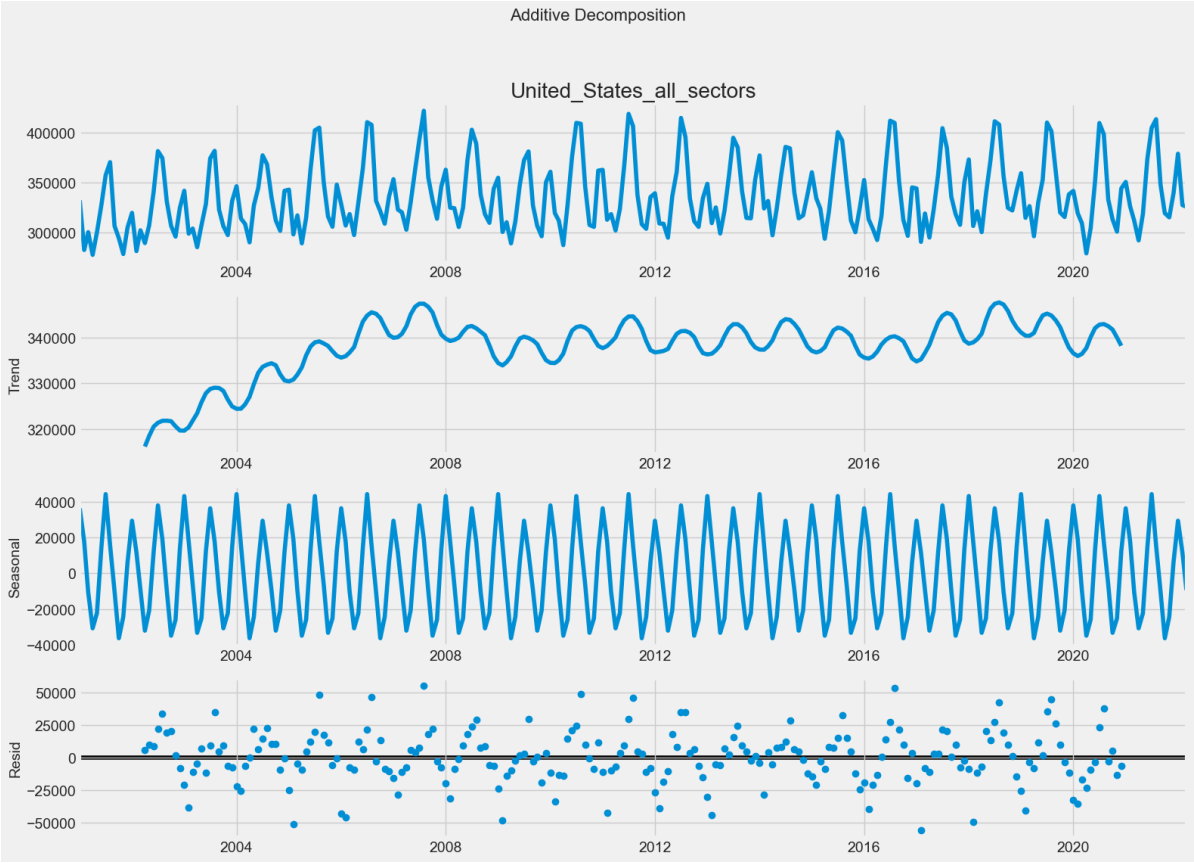
| Date | United_States_all_sectors | United_States_electric_utility | United_States_independent_power_ |
|---|---|---|---|
| 2001-01-01 | 332493 | 236467 | |

# Decomposition (U.S.A)

In [28]:
```
style.use("fivethirtyeight")
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
# Multiplicative Decomposition
multiplicative_decomposition = seasonal_decompose(df_new["United_States_all_

# Additive Decomposition
additive_decomposition = seasonal_decompose(df_new["United_States_all_sector

# Plot
plt.rcParams.update({"figure.figsize": (16,12)})
multiplicative_decomposition.plot().suptitle("Multiplicative Decomposition",
plt.tight_layout(rect=[0, 0.03, 1, 0.95])

additive_decomposition.plot().suptitle("Additive Decomposition", fontsize=16
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

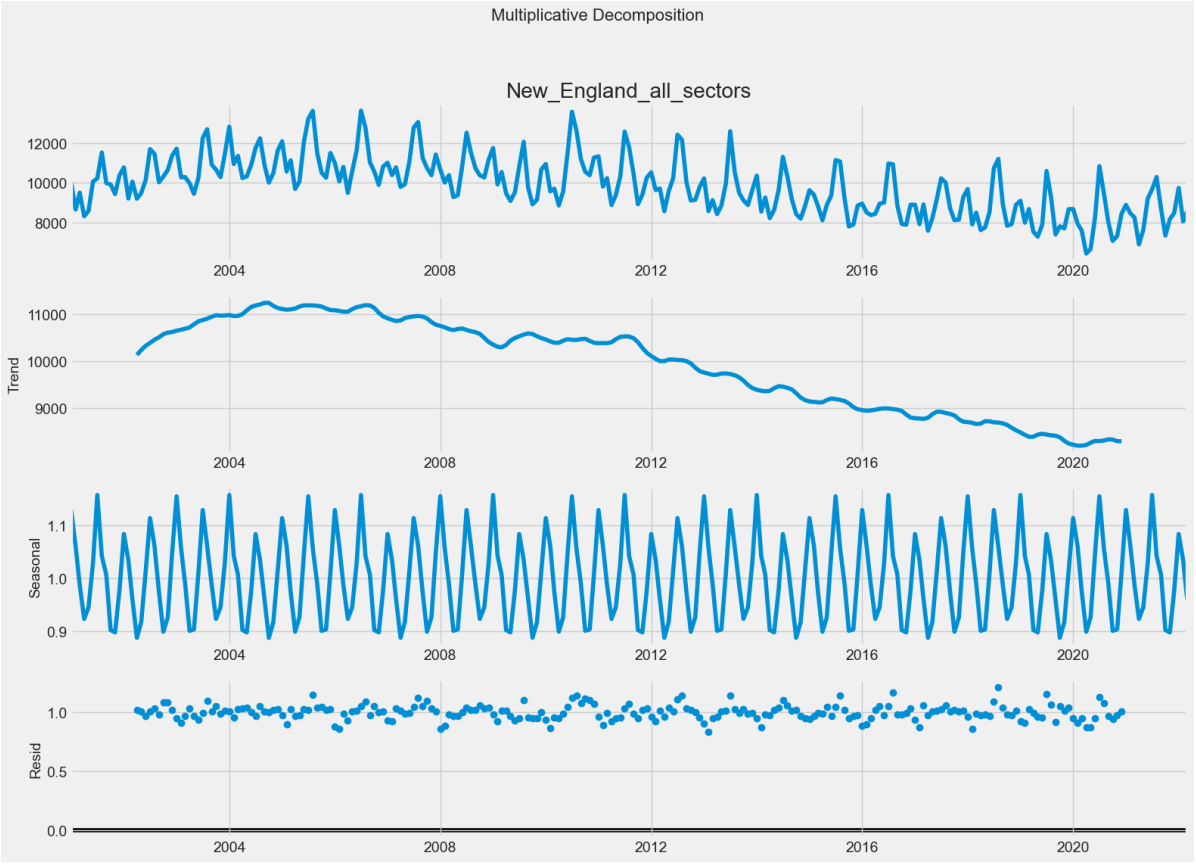


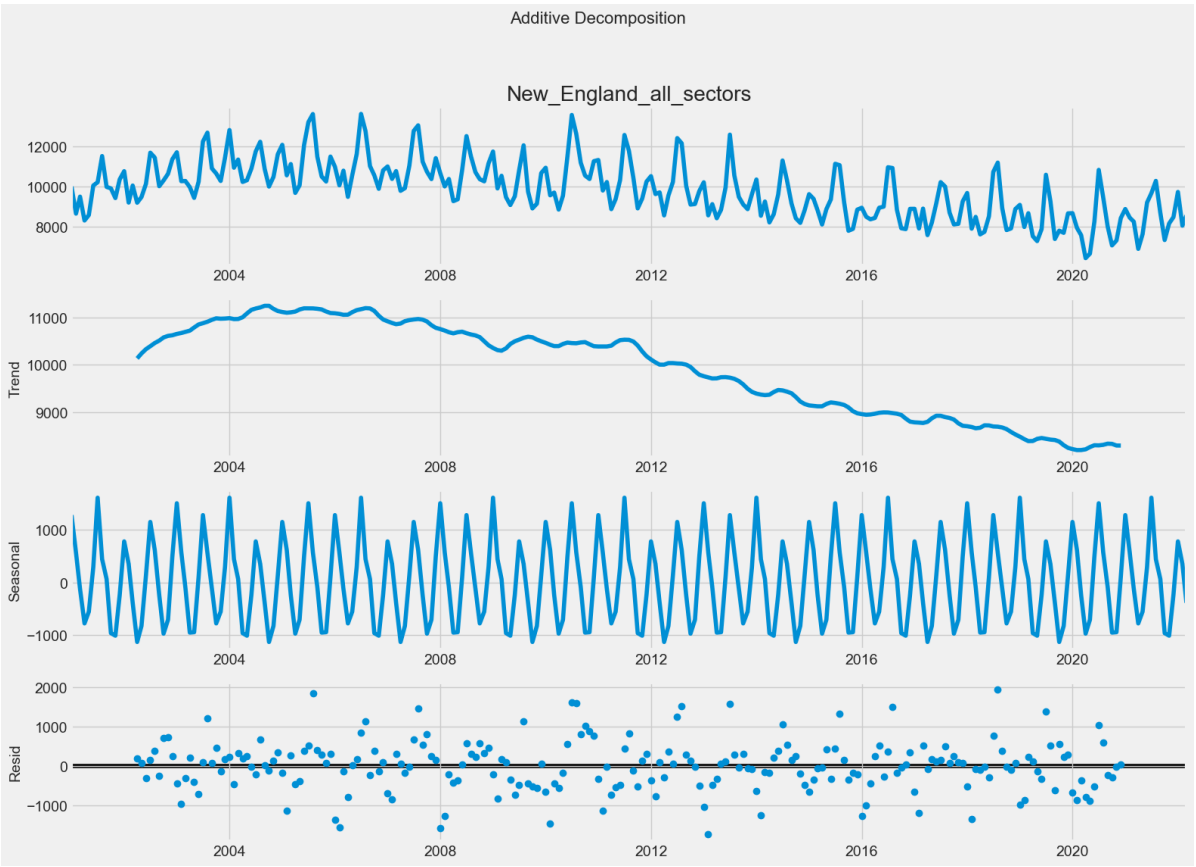



# Decomposition (ENGLAND)

In [29]:

```
style.use("fivethirtyeight")
from statsmodels.tsa.seasonal import seasonal_decompose



# Multiplicative Decomposition
multiplicative_decomposition = seasonal_decompose(df_new["New_England_all_se

# Additive Decomposition
additive_decomposition = seasonal_decompose(df_new["New_England_all_sectors'

# Plot
plt.rcParams.update({"figure.figsize": (16,12)})
multiplicative_decomposition.plot().suptitle("Multiplicative Decomposition",
plt.tight_layout(rect=[0, 0.03, 1, 0.95])

additive_decomposition.plot().suptitle("Additive Decomposition", fontsize=16
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```




In [30]:
```
df_new.head(1)
```

Out[30]:

| Date | United_States_all_sectors | United_States_electric_utility | United_States_independent_power_ |
|---|---|---|---|
| 2001-01-01 | 332493 | 236467 | |

# Autocorrelation (USA)

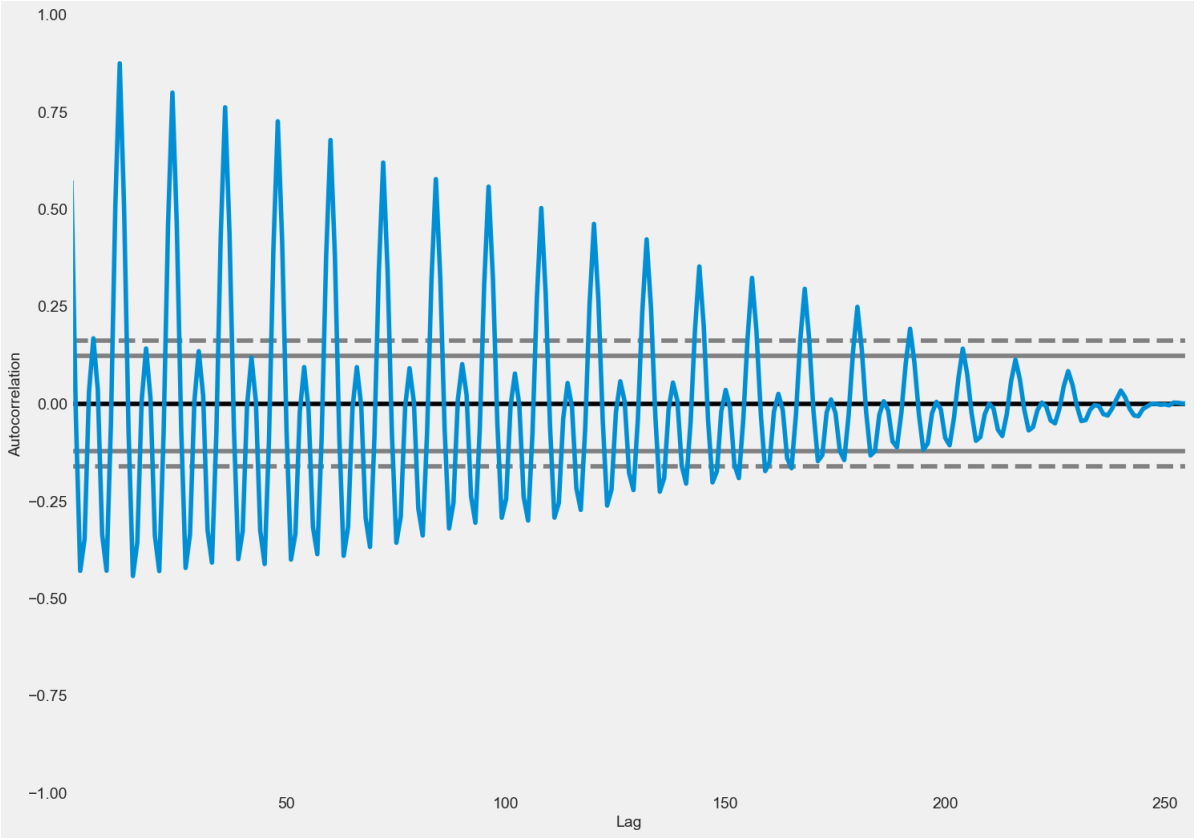
- Now we use auto correlation to understand how past values relate to current values. ###### note : auto correlated lags could be either positive or negative

In [34]:
```
autocorrelation_lag1=df_new["United_States_all_sectors"].autocorr(lag=1)
print("One MOnth Lag :", autocorrelation_lag1)

autocorrelation_lag2=df_new["United_States_all_sectors"].autocorr(lag=2)
print("two MOnth Lag :", autocorrelation_lag2)

autocorrelation_lag3=df_new["United_States_all_sectors"].autocorr(lag=3)
print("three MOnth Lag :", autocorrelation_lag3)

autocorrelation_lag4=df_new["United_States_all_sectors"].autocorr(lag=4)
print("four MOnth Lag :", autocorrelation_lag4)
```

```
One MOnth Lag : 0.5731447691626406
two MOnth Lag : -0.027734834619947163
three MOnth Lag : -0.43422384933985514
four MOnth Lag : -0.35367649096891013
```

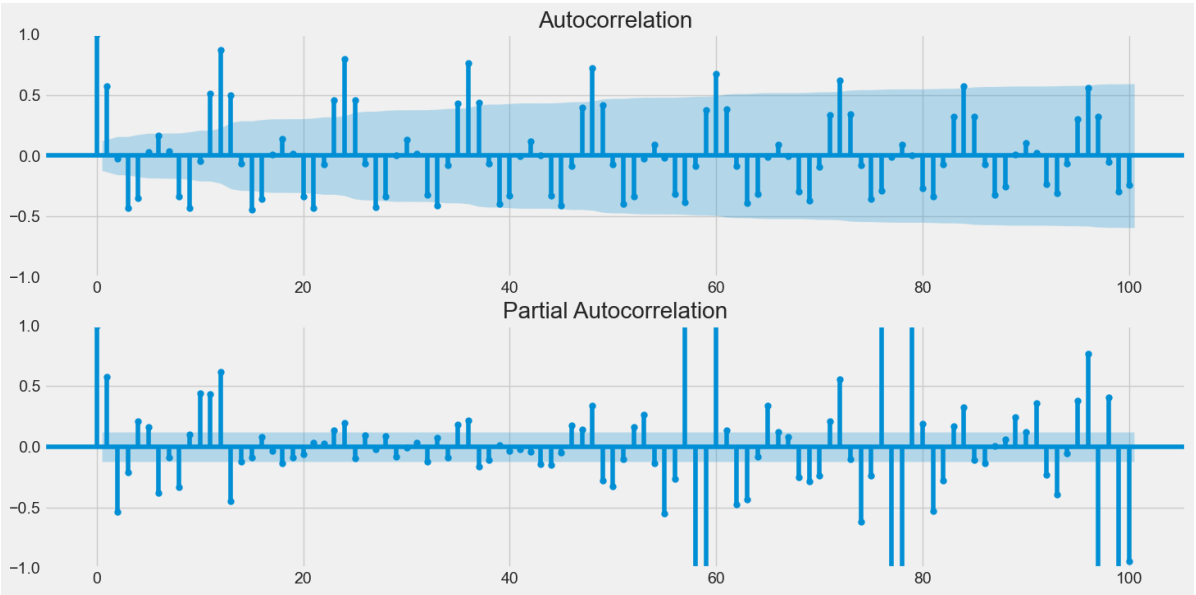
- we can see that we have negative auto-correlation for **United_States_all_sectors** suggesting that high value in the current time period will be followed by a low value in the next time period

In [35]:
```
#plot autocorrellation
style.use("fivethirtyeight")
from pandas.plotting import autocorrelation_plot

autocorrelation_plot(df_new["United_States_all_sectors"])
plt.show()
```




In [36]:

```
style.use("fivethirtyeight")
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

f, ax = plt.subplots(nrows=2, ncols=1, figsize=(16, 8))

plot_acf(df_new["United_States_all_sectors"], lags=100, ax=ax[0])
plot_pacf(df_new["United_States_all_sectors"], lags=100, ax=ax[1])

plt.show()
```




In [37]:

```
df_new.head(1)
```

Out[37]:

| Date | United_States_all_sectors | United_States_electric_utility | United_States_independent_power_ |
|---|---|---|---|
| 2001-01-01 | 332493 | 236467 | |

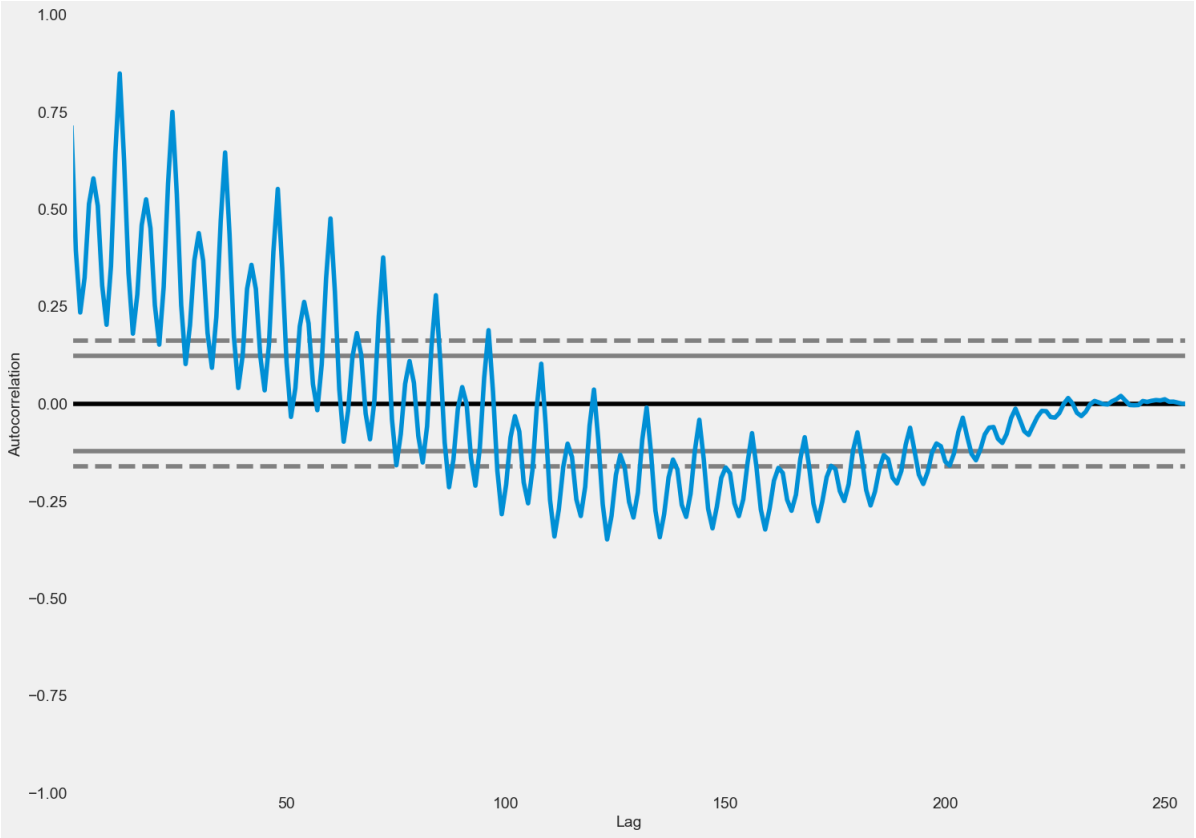
# Autocorrelation (England)

In [41]:
```
autocorrelation_lag1=df_new["New_England_all_sectors"].autocorr(lag=1)
print("One MOnth Lag :", autocorrelation_lag1)

autocorrelation_lag2=df_new["New_England_all_sectors"].autocorr(lag=2)
print("two MOnth Lag :", autocorrelation_lag2)

autocorrelation_lag3=df_new["New_England_all_sectors"].autocorr(lag=3)
print("three MOnth Lag :", autocorrelation_lag3)

autocorrelation_lag4=df_new["New_England_all_sectors"].autocorr(lag=4)
print("four MOnth Lag :", autocorrelation_lag4)

autocorrelation_lag6=df_new["New_England_all_sectors"].autocorr(lag=6)
print("five MOnth Lag :", autocorrelation_lag6)
```

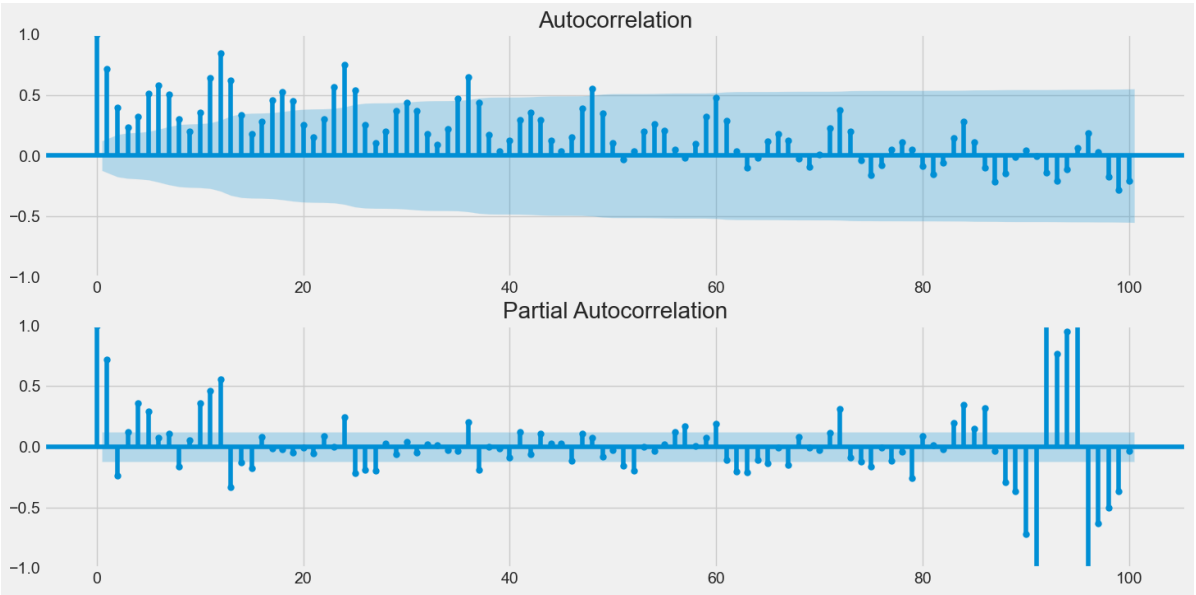
```
One MOnth Lag : 0.715458957704897
two MOnth Lag : 0.3961097475099391
three MOnth Lag : 0.2353956759443829
four MOnth Lag : 0.3266717154869509
five MOnth Lag : 0.590497331800669
```

- we can see that we have positive auto-correlation for **New_England_all_sectors** suggesting that high value in the current time period would be a high tomorrow as well

In [35]:
```
style.use("fivethirtyeight")
from pandas.plotting import autocorrelation_plot

autocorrelation_plot(df_new["New_England_all_sectors"])
plt.show()
```



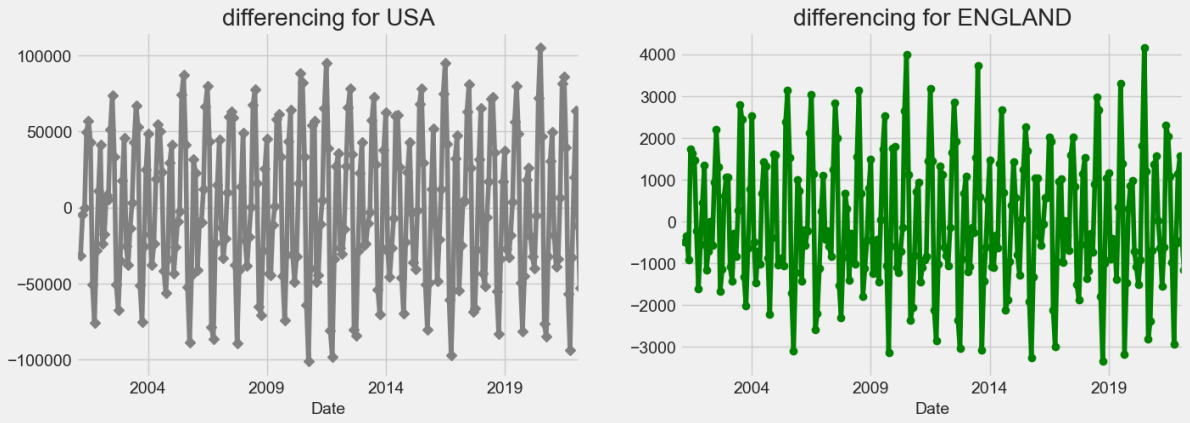

In [36]:
```
style.use("fivethirtyeight")
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

f, ax = plt.subplots(nrows=2, ncols=1, figsize=(16, 8))

plot_acf(df_new["New_England_all_sectors"], lags=100, ax=ax[0])
plot_pacf(df_new["New_England_all_sectors"], lags=100, ax=ax[1])

plt.show()
```




# Differencing

- The diff function helps remove trends from the data to view their stationarity.

In [42]:
```
# plot suplots
style.use("fivethirtyeight")
# plot subplot
f,ax = plt.subplots (1,2,figsize = (15 ,5))
df_new.United_States_all_sectors.diff(2).plot(ax = ax [0],marker="D",color='
df_new.New_England_all_sectors.diff(2).plot(ax = ax [1],marker="o",color="gr

plt.show ()
```

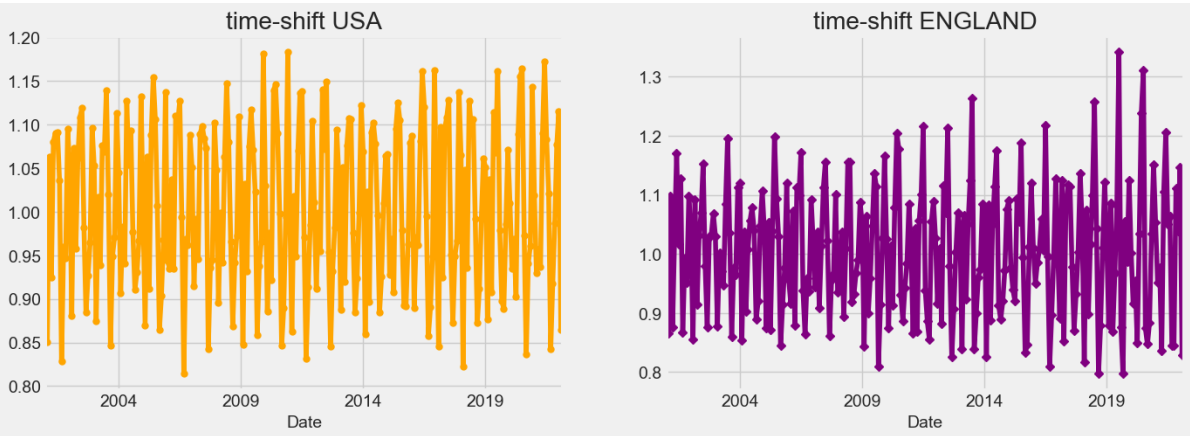



# shift

- we use the shift funtion to view our data's past and current values of previous day and present day

In [44]:
```
# create new column for shift
df_new["USA Shift"] = df_new.United_States_all_sectors.div(df_new.United_Sta
df_new["England Shift"] = df_new.New_England_all_sectors.div(df_new.New_Engl
```

In [45]:
```
style.use("fivethirtyeight")
# plot shift
f,ax = plt.subplots (1,2,figsize = (15 ,5))
df_new["USA Shift"].plot(ax = ax [0],marker="o", markersize= 5,color="orange
df_new["England Shift"].plot(ax = ax [1],marker="D",markersize= 5,color="pur

plt.show ()
```

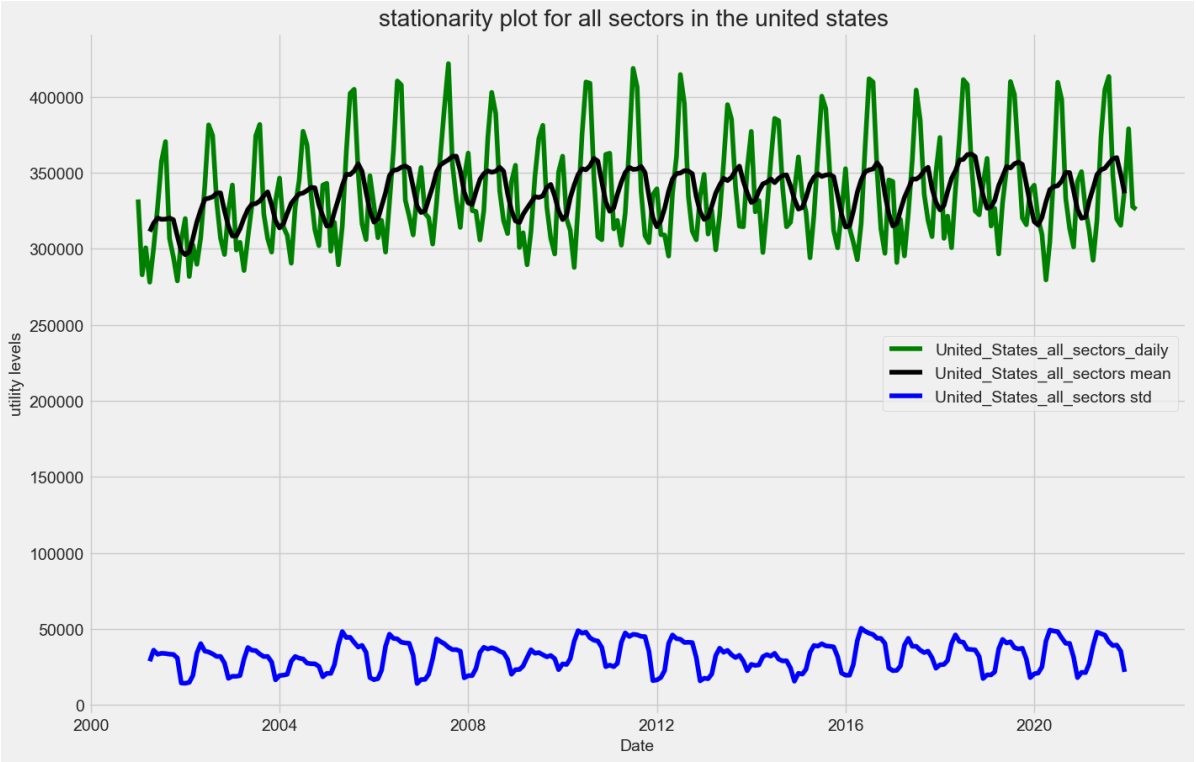



# stationarity

- For forecasting purposes, it is desirable for a time series to exhibit stationarity. In a stationary time series, the mean and standard deviation remain constant over time. Stationarity is important because it allows us to make reliable predictions based on the assumption that the future behavior of the time series will be similar to its past behavior.

In [47]:
```
# preparing stationery data
df_usa_all_sectors=df_new.United_States_all_sectors
dfrolling_USA_sectors_mean= df_new.United_States_all_sectors.rolling(7, cent
dfrolling_USA_sectors_std= df_new.United_States_all_sectors.rolling(7, cente
```

In [50]:
```
# ploting stationery data
fig, ax = plt.subplots(figsize = (15,10))
ax.plot(df_usa_all_sectors,color="green" ,label="United_States_all_sectors_d
ax.plot(dfrolling_USA_sectors_mean,color ="black" ,label="United_States_all_
ax.plot(dfrolling_USA_sectors_std ,color= "blue",label= "United_States_all_s
ax.set_title("stationarity plot for all sectors in the united states ")
ax.set_ylabel("utility levels")
ax.set_xlabel("Date")
plt.legend(loc= "best")
plt.show()
```




In [51]:
```
from statsmodels.tsa.stattools import adfuller
result=adfuller(df_new.United_States_all_sectors)
adf_statistic= result[0]
p_value=result[1]
critical_values= result[4]

print("ADF Statistic:" , adf_statistic)
print("p_value :",p_value)
print ("Critical Values:")
for key, value in critical_values.items():
    print(f"{key}:{value}")
```

```
ADF Statistic: -3.5613572783625376
p_value : 0.006544248574970173
Critical Values:
1%:-3.4578942529658563
5%:-2.8736593200231484
10%:-2.573228767361111
```

- There are several statistical tests available to determine the stationarity of a time series. Here we are going to use Augmented Dickey-Fuller (ADF) Test. The ADF tests the null hypothesis that a unit root is present in the time series (indicating non-stationarity)
- If the p-value obtained from the ADF test is below a chosen significance level (e.g., 0.05), we can reject the null hypothesis and conclude that the series is stationary.

# TIMESERIES FORCASTING

In [52]:
```
# import necessary libraries
from prophet import Prophet
```

In [53]:
```
import prophet
```

## USA power utility forecasting for all sectors

In [54]:
```
# reset dataset for forcasting
df_model= df_new.reset_index()
```

In [55]:
```
df_model.head(1) #  first roll forcast data
```

Out[55]:

| | Date | United_States_all_sectors | United_States_electric_utility | United_States_independent_pow |
|---|---|---|---|---|
| **0** | 2001-01-01 | 332493 | 236467 | |

In [56]:
```
# creat data for data with only data  and target variable
df = pd.DataFrame()
df_model["ds"] = pd.to_datetime(df_model["Date"])
df_model["y"] = df_model["United_States_all_sectors"]
```

In [57]:
```
# model forcasting for 10 years
m = Prophet()
m.fit(df_model)
future = m.make_future_dataframe(periods=12 * 10,
                                 freq="M")
```

```
16:17:11 - cmdstanpy - INFO - Chain [1] start processing
16:17:12 - cmdstanpy - INFO - Chain [1] done processing
```

In [58]:
```
# creating future forcasting
forecast = m.predict(future)
forecast[["ds", "yhat", "yhat_lower",
```

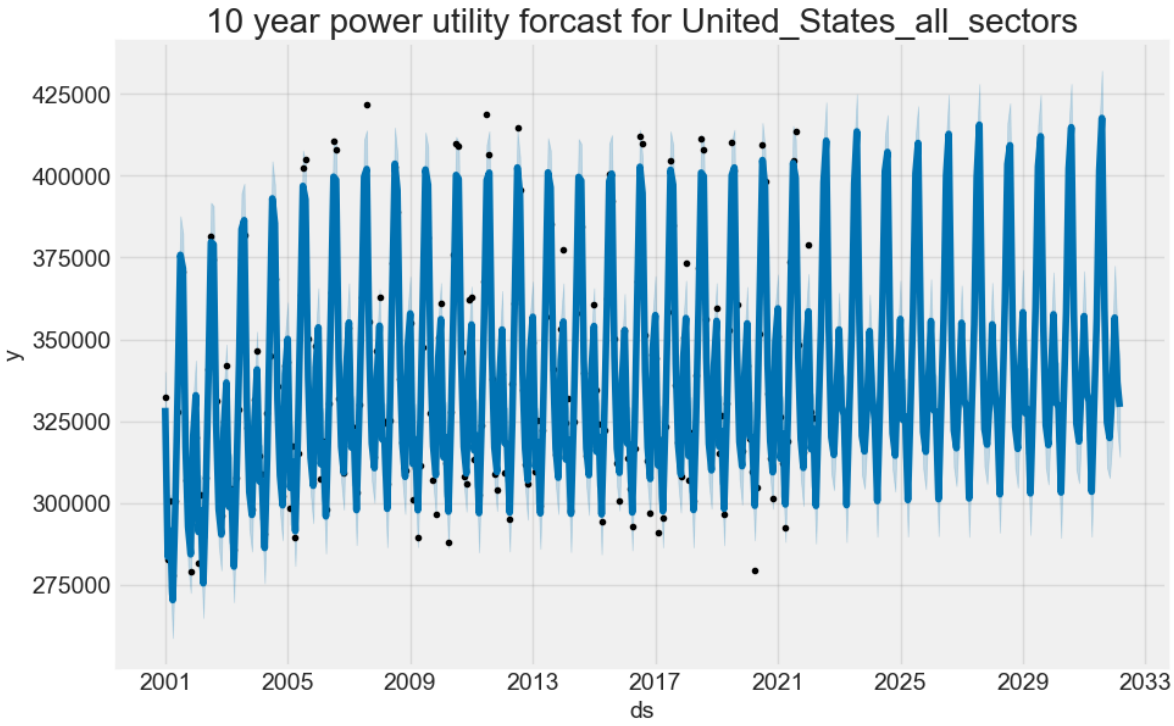
```
            "yhat_upper", "trend",
            "trend_lower", "trend_upper"]].tail()
```

Out[58]:

| | ds | yhat | yhat_lower | yhat_upper | trend | trend_lower | trend |
|---|---|---|---|---|---|---|---|
| **370** | 2031-10-31 | 319848.045469 | 307657.243420 | 334332.172354 | 349091.118320 | 340725.407825 | 357935 |
| **371** | 2031-11-30 | 333293.646552 | 318690.434434 | 347911.510010 | 349133.246912 | 340700.227377 | 358081 |
| **372** | 2031-12-31 | 356629.915191 | 342885.600817 | 372465.574712 | 349176.779791 | 340620.136459 | 358223 |
| **373** | 2032-01-31 | 336798.532061 | 322325.893029 | 351753.002248 | 349220.312670 | 340497.752485 | 358424 |
| **374** | 2032-02-29 | 329335.777514 | 314149.728967 | 343731.615490 | 349261.036976 | 340411.200960 | 358612 |

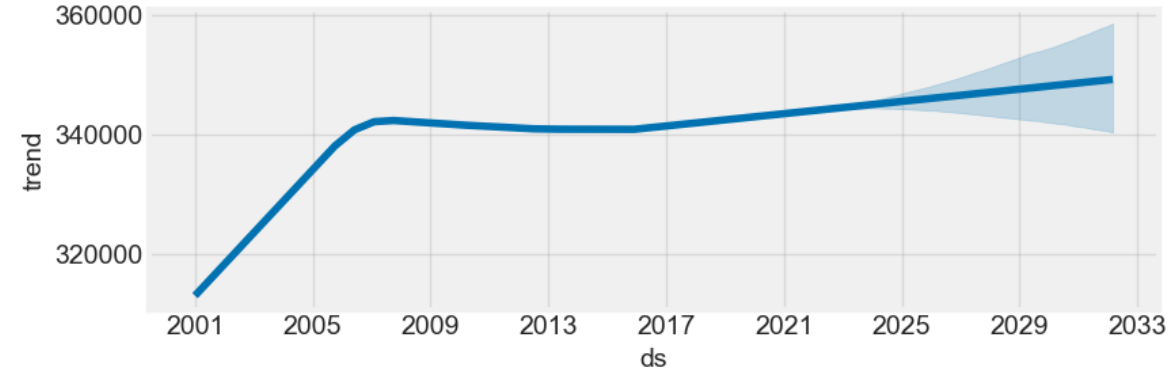
In [60]:

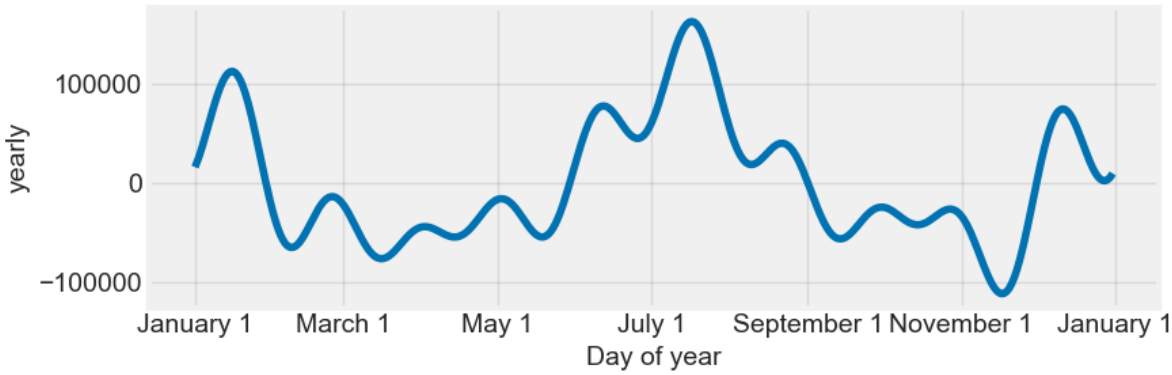
```
# plot style
style.use("fivethirtyeight")
# show plot
fig1 = m.plot(forecast)
plt.title("10 year power utility forcast for United_States_all_sectors")
plt.show()
```



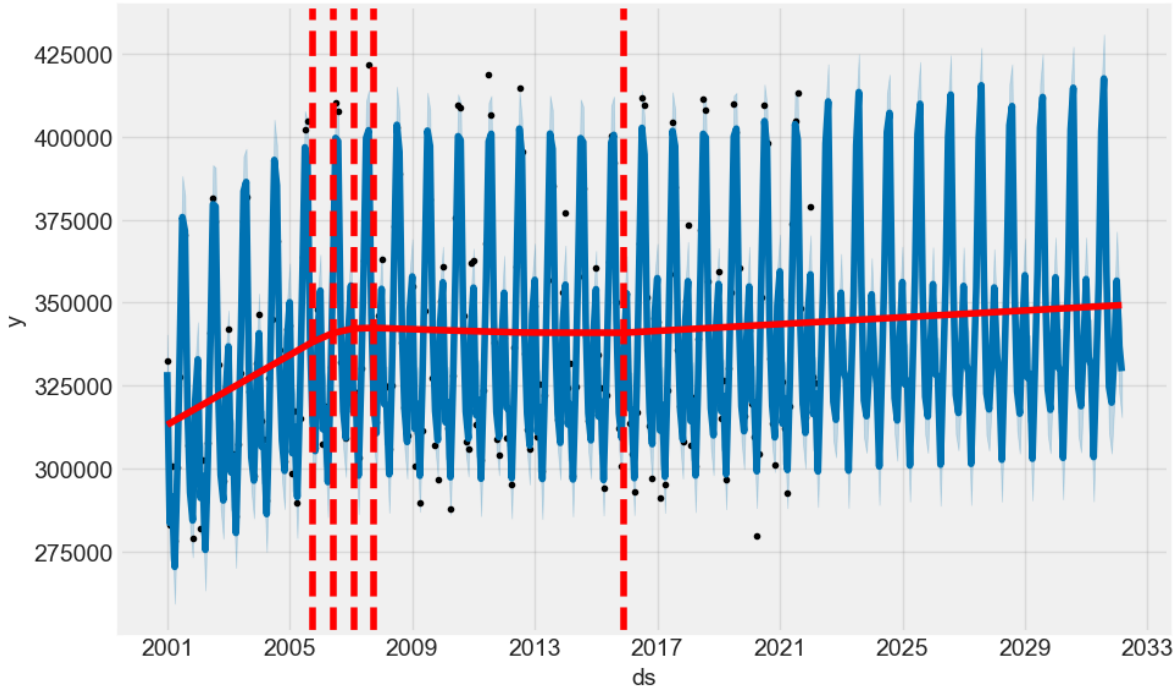

In [61]:

```
# plot style
style.use("fivethirtyeight")
fig2 = m.plot_components(forecast)
# show plot
fig2=plt.show()
fig2
```




In [56]:
```
# plot style
style.use("fivethirtyeight")
# show changes
from prophet.plot import add_changepoints_to_plot
fig = m.plot(forecast)
a = add_changepoints_to_plot(fig.gca(),
                             m, forecast)
plt.show()
```




# TIMESERIES PREDICTION

Difference between forcasting and prediction

- A forecast can be said to be a calculation or an estimation which uses data from previous events, together with recent trends to come up a future outcome. A prediction is an actual act of indicating that something will happen in the future with or without prior information. in other words forcasting **shows** the future while prediction **tells** the future

In [61]:
```
# reset dataset for predections
df_model_2= df_new.reset_index()
```

In [62]:
```
# new data for predictive model
df_model_2.head(1)
```

Out[62]:

| | Date | United_States_all_sectors | United_States_electric_utility | United_States_independent_pow |
|---|---|---|---|---|
| **0** | 2001-01-01 | 332493 | 236467 | |

In [64]:
```
# Getting the test and train size of the data
train_size = int(0.99 * len(df_model_2))
test_size = len(df_model_2) - train_size
# Train models
Train_df_model_2 = df_model_2[["Date", "United_States_electric_utility"]].co
Train_df_model_2.columns = ["ds", "y"]
train = Train_df_model_2.iloc[:train_size, :]
x_train, y_train = pd.DataFrame(Train_df_model_2.iloc[:train_size, 0]), pd.D
x_valid, y_valid = pd.DataFrame(Train_df_model_2.iloc[train_size:, 0]), pd.D
# length of model
print(len(train), len(x_valid))
```

```
252 3
```

In [73]:
```
# viewing training data
train
```

Out[73]:

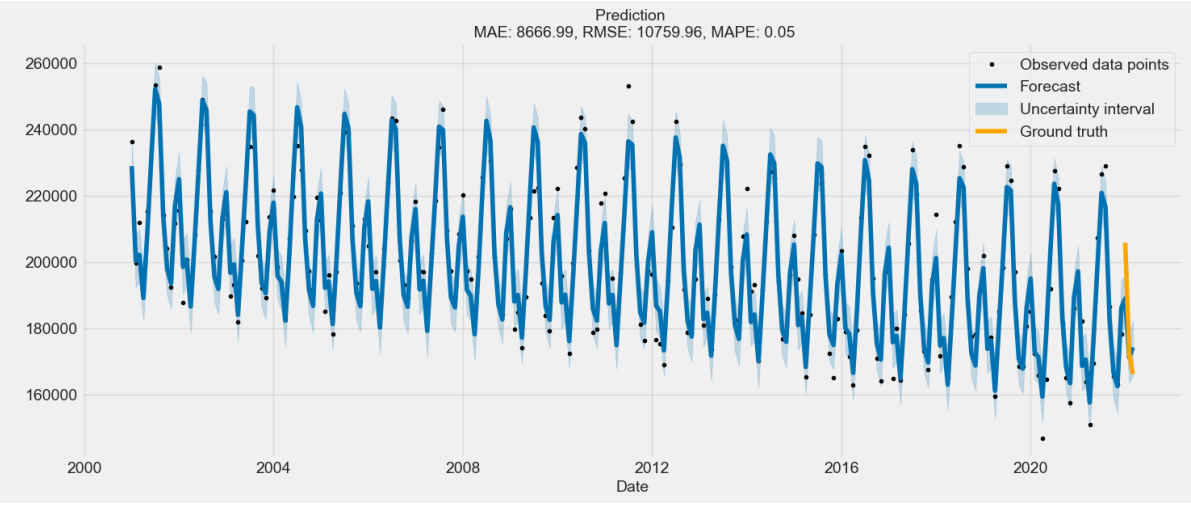
| | ds | y |
|---|---|---|
| **0** | 2001-01-01 | 236467 |
| **1** | 2001-02-01 | 199802 |
| **2** | 2001-03-01 | 211942 |
| **3** | 2001-04-01 | 197499 |
| **4** | 2001-05-01 | 215508 |
| **...** | ... | ... |
| **247** | 2021-08-01 | 229150 |
| **248** | 2021-09-01 | 186742 |
| **249** | 2021-10-01 | 165575 |
| **250** | 2021-11-01 | 163203 |
| **251** | 2021-12-01 | 178473 |

252 rows × 2 columns

In [71]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error,mean_abs
import math

from prophet import Prophet


# Train the model
model = Prophet()
model.fit(train)

x_valid = model.make_future_dataframe(periods=test_size, freq="M")

# Predict on valid set
y_pred = model.predict(x_valid)

# Calcuate metrics
score_mae = mean_absolute_error(y_valid, y_pred.tail(test_size)["yhat"])
score_rmse = math.sqrt(mean_squared_error(y_valid, y_pred.tail(test_size)["y
score_mape = mean_absolute_percentage_error(y_valid, y_pred.tail(test_size)[
print("MAE is {}".format(score_mae))
print("RMSE: {}".format(score_rmse))
print("MAPE is {}".format(score_mape))
```

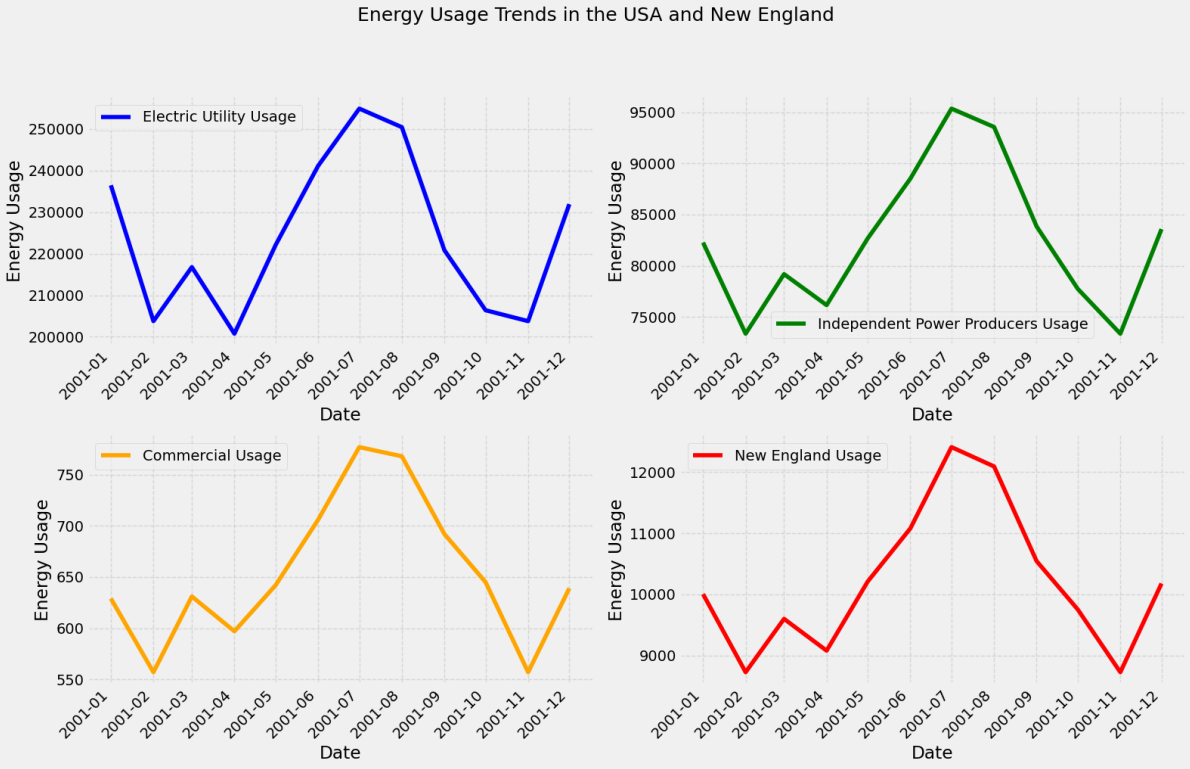
```
16:53:01 - cmdstanpy - INFO - Chain [1] start processing
16:53:01 - cmdstanpy - INFO - Chain [1] done processing
MAE is 8666.993245216241
RMSE: 10759.959643823686
MAPE is 0.04558199198660601
```

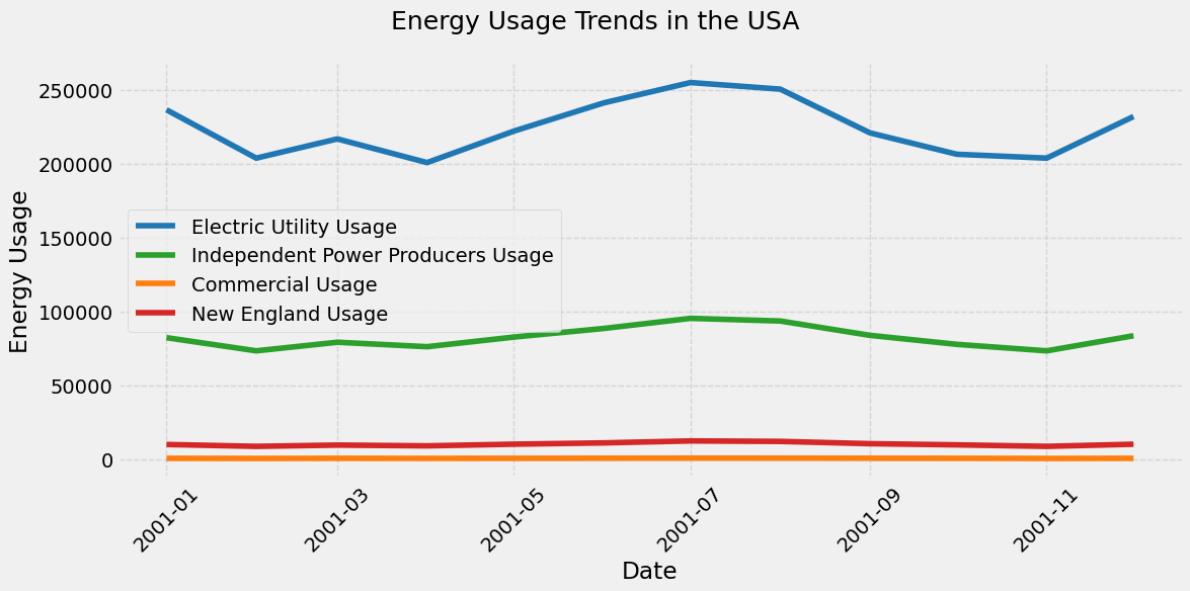
In [72]:
```
style.use("fivethirtyeight")
# plot prediction models
f, ax = plt.subplots(1)
f.set_figheight(6)
f.set_figwidth(15)

model.plot(y_pred, ax=ax)
sns.lineplot(x=x_valid["ds"], y=y_valid["y"], ax=ax, color="orange", label='

ax.set_title(f"Prediction \n MAE: {score_mae:.2f}, RMSE: {score_rmse:.2f}, M
ax.set_xlabel(xlabel="Date", fontsize=14)
ax.set_ylabel(ylabel=" ", fontsize=14)

plt.show()
```



# Energy usage Trends(USA & New England)

In [25]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('Energy_data_usage_USA.csv')

df['Date'] = pd.to_datetime(df['Date'])

fig, axs = plt.subplots(2, 2, figsize=(16, 10))
fig.suptitle('Energy Usage Trends in the USA and New England', fontsize=18,

axs[0, 0].plot(df['Date'], df['United States: electric utility'], color='blu
axs[0, 1].plot(df['Date'], df['United States: independent power producers'],
axs[1, 0].plot(df['Date'], df['United States: all commercial'], color='orang
axs[1, 1].plot(df['Date'], df['New England: all sectors'], color='red', labe

for ax in axs.flat:
    ax.set_xticks(df['Date'].values)
    ax.set_xticklabels(df['Date'].dt.strftime('%Y-%m'), rotation=45, ha='rig

for ax in axs.flat:
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.set_xlabel('Date')
    ax.set_ylabel('Energy Usage')
    ax.legend()

for ax in axs.flat:
    ax.grid(True, linestyle='--', alpha=0.7)

fig.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()

fig, ax = plt.subplots(figsize=(12, 6))
fig.suptitle('Energy Usage Trends in the USA', fontsize=18)

ax.plot(df['Date'], df['United States: electric utility'], color='tab:blue',
ax.plot(df['Date'], df['United States: independent power producers'], color=
ax.plot(df['Date'], df['United States: all commercial'], color='tab:orange',
ax.plot(df['Date'], df['New England: all sectors'], color='tab:red', linesty

for tick in ax.get_xticklabels():
    tick.set_rotation(45)

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.set_xlabel('Date')
ax.set_ylabel('Energy Usage')
ax.legend()
ax.grid(True, linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```


Energy Usage Trends in the USA and New England
Electric Utility Usage
Independent Power Producers Usage
Commercial Usage
New England Usage
Energy Usage
Date


Energy Usage Trends in the USA
Electric Utility Usage
Independent Power Producers Usage
Commercial Usage
New England Usage
Energy Usage
Date

In [ ]:

In [ ]:

In [ ]: