

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.simplefilter('ignore')
```

```
In [2]: df = pd.read_csv("/kaggle/input/daily-power-generation-in-india-2013-2023/Daily_Power_Gen_States_march_23.csv")
display(df.shape)
df.head()
```

	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date
0	NER	Mizoram		77	1.0	1.2 2015-01-01
1	WR	DD		214	0.0	4.8 2015-01-01
2	WR	Goa		383	0.0	7.3 2015-01-01
3	WR	Maharashtra		14837	57.0	315.0 2015-01-01
4	WR	MP		5740	0.0	109.8 2015-01-01

```
In [3]: df1 = pd.read_csv("/kaggle/input/daily-power-generation-in-india-2013-2023/Daily_Power_Gen_Source_march_23.csv")
display(df1.shape)
df1.head()
```

	source	NR	WR	SR	ER	NER	All India	date
0	Hydro	139.0	43.0	72.0	30.0	7.0	292	2013-03-31
1	Total	675.0	820.0	697.0	306.0	28.0	2526	2013-03-31
2	Wind Gen(MU)	2.0	19.0	13.0	0.0	0.0	34	2013-03-31
3	Hydro	137.0	43.0	83.0	32.0	5.0	300	2013-04-01
4	Total	683.0	841.0	706.0	316.0	29.0	2575	2013-04-01

```
In [4]: df['Shortage during maximum Demand(MW)'] = df['Shortage during maximum Demand(MW)'].fillna(df['Shortage during maximum Demand(MW)'].mean())
df['Energy Met (MU)'] = df['Energy Met (MU)'].fillna(df['Energy Met (MU)'].mean())
```

```
In [5]: print("Start date:", df['date'].min())
print("End date:", df['date'].max())
```

Start date: 2013-03-31  
End date: 2023-03-31

```
In [6]: df['date'] = pd.to_datetime(df['date'])

df['DayName'] = pd.to_datetime(df['date']).apply(lambda x: x.day_name())
df['MonthName'] = pd.to_datetime(df['date']).apply(lambda x: x.month_name())
df['Year'] = pd.to_datetime(df['date']).dt.year
df['Quarter'] = pd.to_datetime(df['date']).dt.quarter
df['Month'] = pd.to_datetime(df['date']).dt.month

df["Season"] = [ "Winter" if i < 3 or i > 11 else "Spring" if 3 <= i < 6 else "Summer" if 6 <= i < 9 else "Autumn" if 9 <= i < 12 else "Fall" for i in range(1, 13)]
```

	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
0	NER	Mizoram	77	1.0	1.2	2015-01-01	Thursday	January	2015	1	1	Winter
1	WR	DD	214	0.0	4.8	2015-01-01	Thursday	January	2015	1	1	Winter
2	WR	Goa	383	0.0	7.3	2015-01-01	Thursday	January	2015	1	1	Winter
3	WR	Maharashtra	14837	57.0	315.0	2015-01-01	Thursday	January	2015	1	1	Winter
4	WR	MP	5740	0.0	109.8	2015-01-01	Thursday	January	2015	1	1	Winter

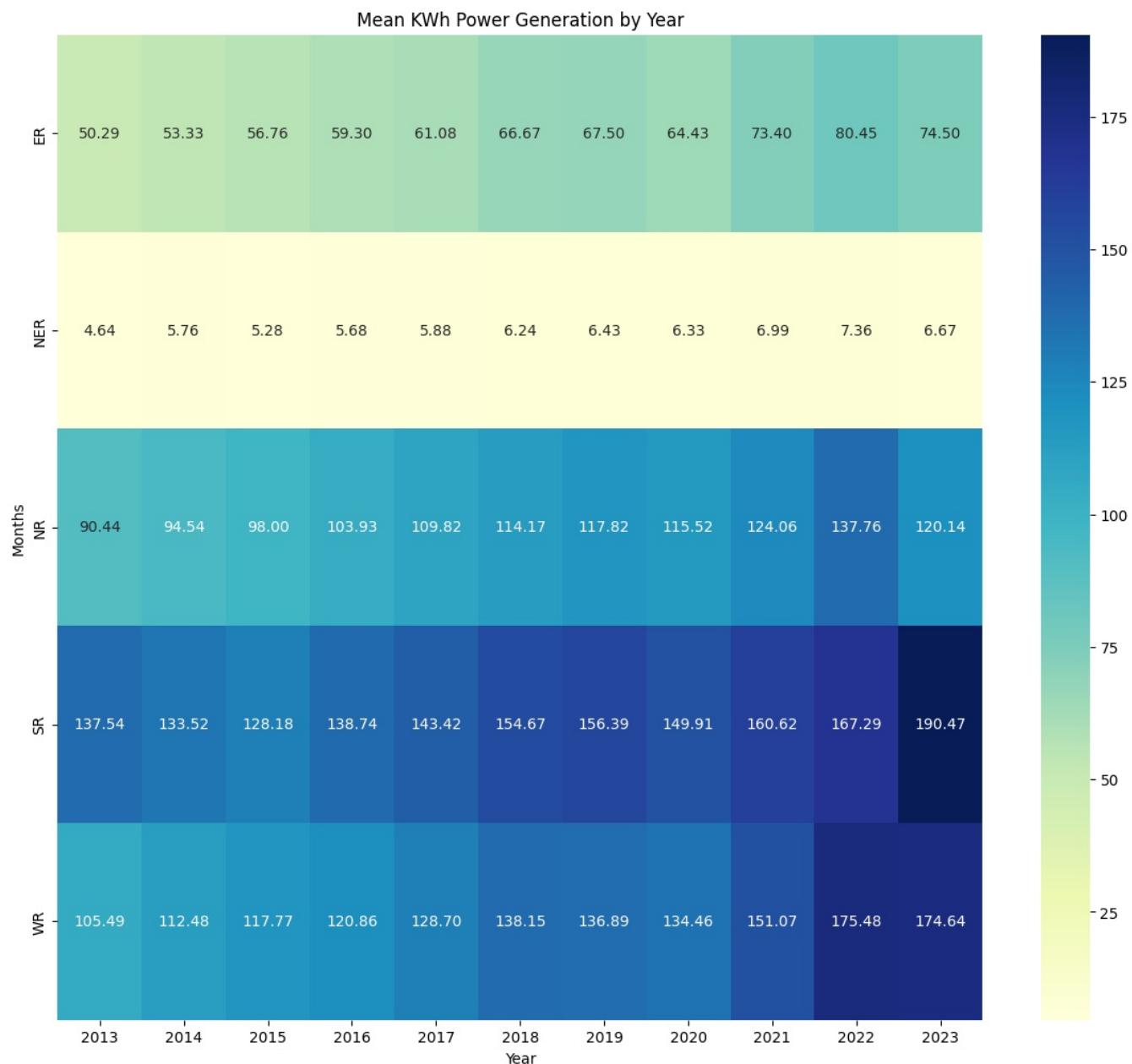
```
In [7]: # Mean Price by Month
df1 = df.copy('deep')

# Mean Price by Month
```

```

pivot_table = df1.pivot_table(values='Energy Met (MU)', index='Region', columns='Year', aggfunc='mean')
plt.figure(figsize=(14, 12))
sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="YlGnBu")
plt.title('Mean KWh Power Generation by Year')
plt.xlabel('Year')
plt.ylabel('Months')
plt.show()

```



```
In [8]: names_to_drop = ['AMNSIL', 'DNHDDPDCL', 'BALCO', 'Railways_NR ISTS', 'DNH', 'DD', 'DVC', 'Essar steel']
df.drop(df[df['States'].isin(names_to_drop)].index, inplace=True)
```

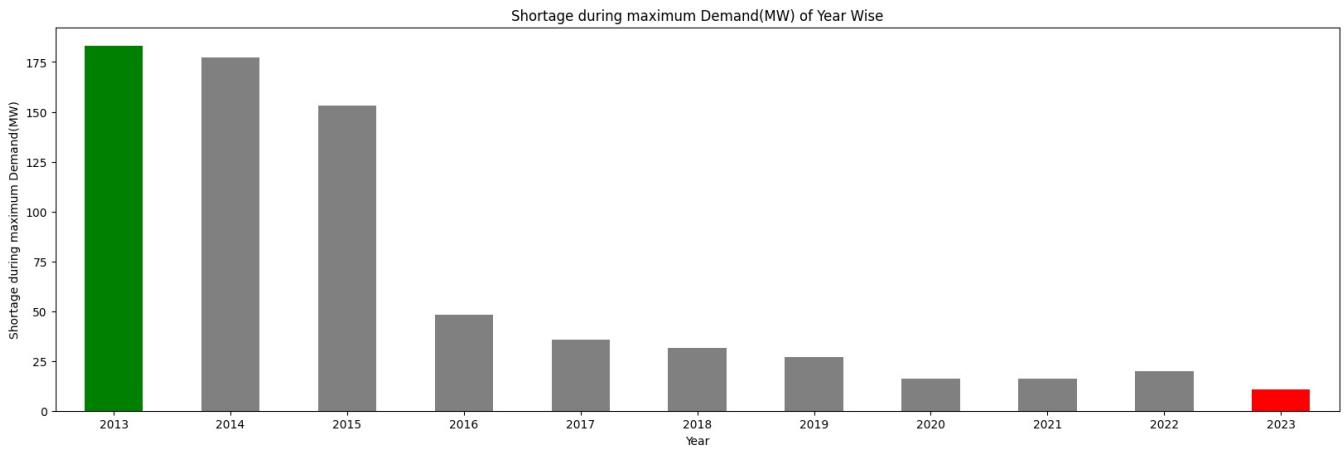
```
In [9]: df["States"].replace({'J&K(UT) & Ladakh(UT)' : 'J&K(UT) & Ladakh(UT)', 'J&K(UT) \nLadakh(UT)' : 'J&K(UT) & Ladakh(UT)', 'J&K(UT) and Ladakh(UT)' : 'J&K(UT) & Ladakh(UT)', 'J&K' : 'J&K(UT) & Ladakh(UT)', 'Pondy'}
```

```
In [10]: # Mean Price by Month
df2 = df.copy('deep')

# Mean Price by Month
pivot_table = df2.pivot_table(values='Energy Met (MU)', index='States', columns='Year', aggfunc='mean')
plt.figure(figsize=(14, 12))
sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="summer")
plt.title('Mean Power Generation by Year')
plt.xlabel('Year')
plt.ylabel('Months')
plt.show()
```

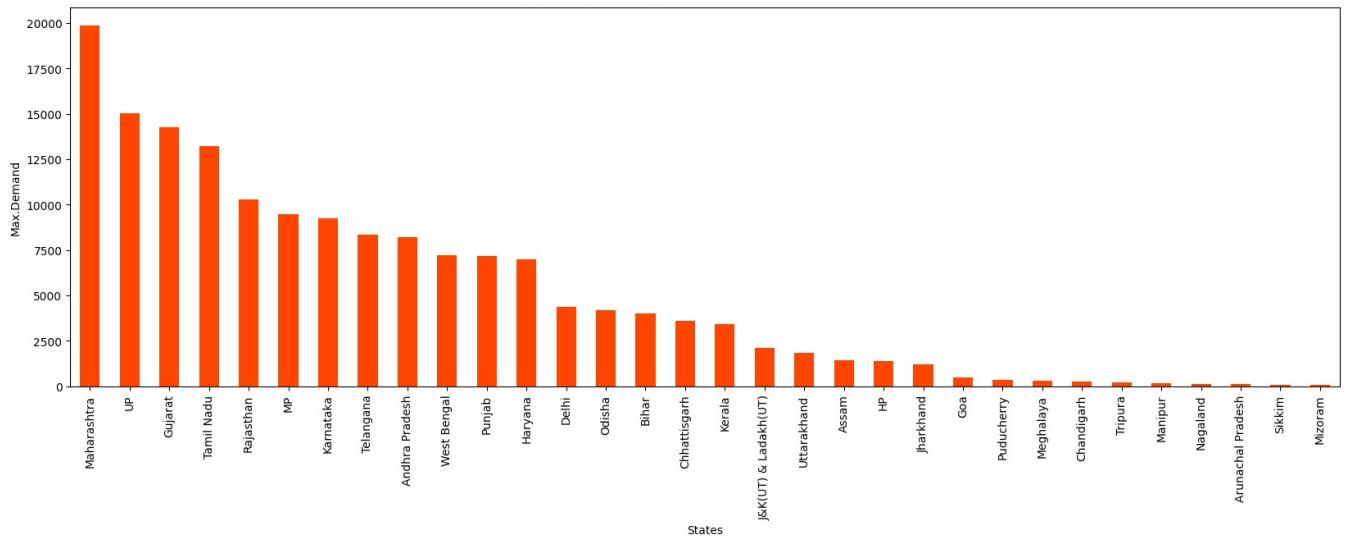
Mean Power Generation by Year												
Months	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	
	Andhra Pradesh	235.39	182.73	136.90	146.66	155.62	173.75	175.15	168.51	184.86	194.59	211.52
Arunachal Pradesh	1.30	1.91	1.54	1.91	2.08	2.16	2.17	2.02	2.25	2.35	2.51	
Assam	19.86	24.41	22.45	24.30	24.82	26.04	26.38	26.15	29.74	31.40	26.37	
Bihar	38.76	45.83	58.86	65.49	70.40	80.93	85.77	88.75	98.01	105.52	91.74	
Chandigarh	4.54	4.41	4.43	4.53	4.43	4.25	4.41	4.15	4.38	4.88	3.80	
Chhattisgarh	62.15	67.83	75.00	74.66	79.56	82.07	86.30	81.24	92.23	99.90	109.61	
Delhi	81.74	79.75	79.90	83.10	85.83	87.60	90.20	79.33	84.08	95.48	72.29	
Goa	7.64	8.06	8.48	8.77	9.20	10.25	11.58	9.48	11.57	12.59	13.33	
Gujarat	233.26	260.99	274.79	283.90	298.93	329.27	328.33	312.13	352.21	381.38	385.50	
HP	23.60	24.01	24.52	24.93	25.84	26.53	28.07	26.82	30.58	32.46	32.99	
Haryana	124.29	125.73	128.02	134.54	137.49	142.15	148.91	140.99	151.28	164.01	136.27	
J&K(UT) & Ladakh(UT)	31.22	32.73	37.75	37.82	40.22	41.57	43.65	46.16	49.59	52.46	59.24	
Jharkhand	19.52	20.15	21.86	21.47	23.05	23.39	23.99	25.31	27.20	30.50	28.15	
Karnataka	151.55	165.33	167.63	183.06	183.53	194.99	196.40	191.17	196.72	207.36	262.06	
Kerala	55.26	59.51	61.14	65.17	66.18	67.85	71.46	68.87	72.17	75.71	80.58	
MP	141.19	154.87	164.83	172.92	186.43	202.08	206.84	222.96	234.42	251.91	280.34	
Maharashtra	338.81	379.32	389.47	393.99	423.68	446.60	429.59	416.48	474.32	516.15	550.80	
Manipur	1.40	2.11	2.05	2.04	2.21	2.34	2.50	2.62	2.67	2.80	3.02	
Meghalaya	4.33	5.09	4.72	4.70	4.92	5.42	5.79	5.57	5.95	6.23	6.77	
Mizoram	1.05	1.30	1.18	1.27	1.35	1.58	1.74	1.63	1.63	1.75	1.93	
Nagaland	1.42	2.03	1.85	1.81	2.04	2.13	2.17	2.24	2.34	2.45	2.17	
Odisha	66.83	69.94	69.22	73.19	76.89	85.76	84.02	78.53	100.02	115.64	105.06	
Puducherry	6.08	6.24	6.29	6.74	6.84	7.21	7.77	7.08	7.87	8.67	8.61	
Punjab	136.32	131.17	131.39	142.10	144.40	154.63	154.65	153.25	170.77	188.40	145.29	
Rajasthan	157.31	181.73	189.76	195.05	195.34	211.86	222.05	228.02	241.28	269.72	278.18	
Sikkim	1.27	1.29	1.32	1.32	1.30	1.30	1.20	1.40	1.40	1.58	1.75	
Tamil Nadu	239.43	256.96	263.17	291.52	288.87	300.91	304.93	283.11	301.67	316.65	322.09	
Telangana		128.16	133.97	139.32	159.48	183.32	182.63	180.71	200.44	200.75	257.98	
Tripura	3.13	3.41	3.15	3.71	3.77	3.99	4.27	4.07	4.38	4.53	3.92	
UP	223.82	238.66	251.34	277.75	318.83	321.41	329.78	325.62	345.57	390.46	326.21	
Uttarakhand	31.11	32.68	34.88	35.57	36.01	37.48	38.30	35.33	38.98	41.95	39.99	
West Bengal	119.17	127.13	131.86	134.22	130.77	142.76	145.97	133.62	147.89	156.70	146.25	

```
In [11]: plt.figure(figsize=(20, 6))
data = df[['Year', 'Shortage during maximum Demand(MW)']].copy('deep')
color = list(np.full(12, 'grey'))
color[0], color[10] = 'Green', 'Red'
data.groupby('Year').mean()['Shortage during maximum Demand(MW)'].plot(kind='bar', title='Shortage during maximum Demand(MW) of Year Wise')
plt.ylabel('Shortage during maximum Demand(MW)')
```



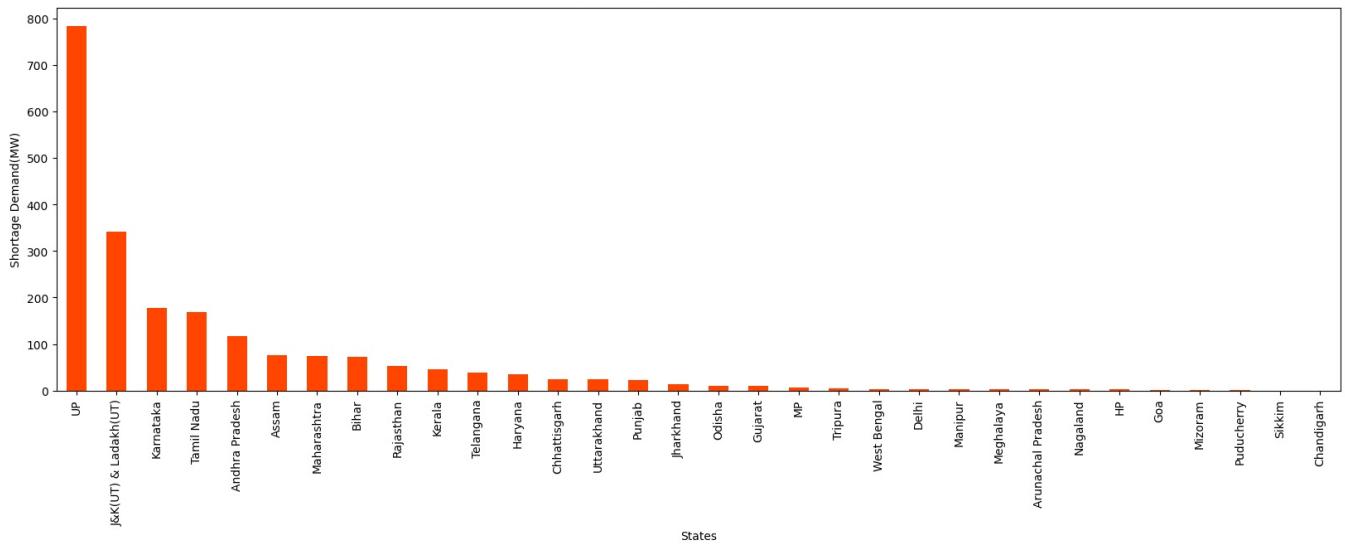
```
In [12]: plt.figure(figsize=(20, 6))
chart = df.groupby('States')['Max.Demand Met during the day(MW)'].mean().sort_values(ascending = False).plot(kind='bar', title='Max.Demand Met during the day(MW) by States', fontweight='bold', color='red', pad=12)
chart.set_xticklabels(chart.get_xticklabels(), rotation = 90)
plt.xlabel('States')
plt.ylabel('Max.Demand')
```

### Max.Demand Met during the day(MW) by States



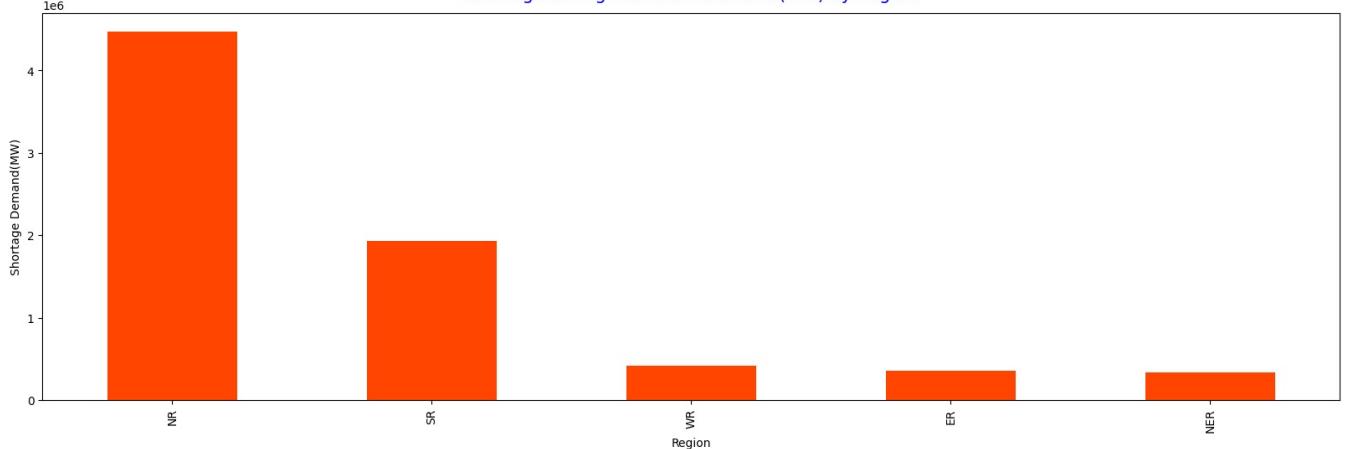
```
In [13]: plt.figure(figsize=(20, 6))
chart = df.groupby('States')['Shortage during maximum Demand(MW)'].mean().sort_values(ascending = False).plot(kind='bar')
chart.set_xticklabels(chart.get_xticklabels(), rotation = 90)
plt.title('Shortage during maximum Demand(MW) by States', fontsize = 15, color = 'b', pad = 12)
plt.xlabel('States')
plt.ylabel('Shortage Demand(MW)');
```

Shortage during maximum Demand(MW) by States



```
In [14]: plt.figure(figsize=(20, 6))
chart = df.groupby('Region')['Shortage during maximum Demand(MW)'].sum().sort_values(ascending = False).plot(kind='bar')
chart.set_xticklabels(chart.get_xticklabels(), rotation = 90)
plt.title('Shortage during maximum Demand(MW) by Region', fontsize = 15, color = 'b', pad = 12)
plt.xlabel('Region')
plt.ylabel('Shortage Demand(MW)');
```

Shortage during maximum Demand(MW) by Region



```
In [15]: df.head(1)
```

Out[15]:

	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
0	NER	Mizoram	77	1.0	1.2	2015-01-01	Thursday	January	2015	1	1	Winter

In [16]:

```

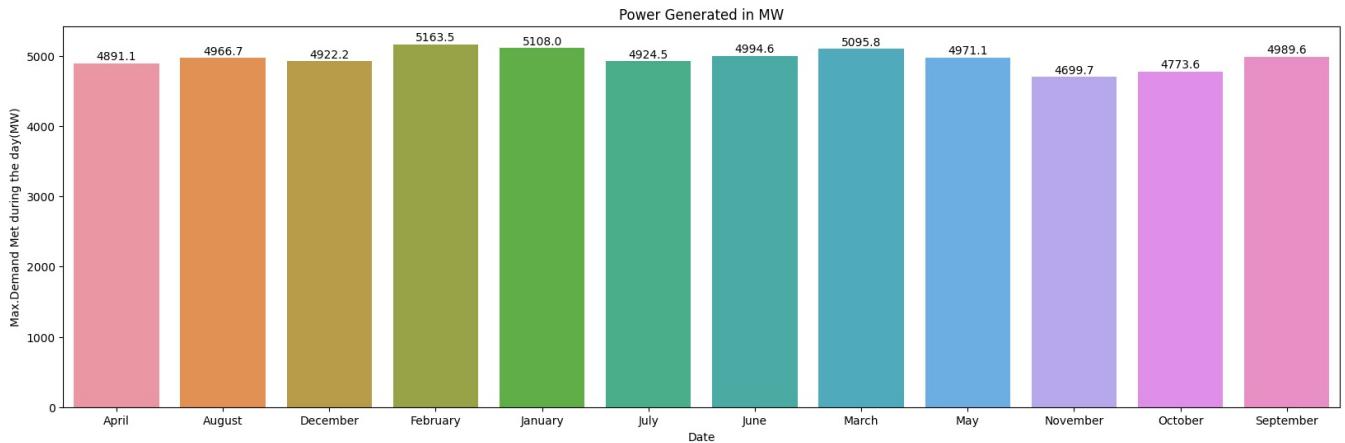
def show_values(axes, orient="v", space=.01):
    def _single(ax):
        if orient == "v":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() / 2
                _y = p.get_y() + p.get_height() + (p.get_height()*0.01)
                value = '{:.1f}'.format(p.get_height())
                ax.text(_x, _y, value, ha="center")
        elif orient == "h":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() + float(space)
                _y = p.get_y() + p.get_height() - (p.get_height()*0.5)
                value = '{:.1f}'.format(p.get_width())
                ax.text(_x, _y, value, ha="left")

        if isinstance(axes, np.ndarray):
            for idx, ax in np.ndenumerate(axes):
                _single(ax)
        else:
            _single(axes)

plt.figure(figsize=(20,6))
a=sns.barplot(x=df.groupby(["MonthName"])["Max.Demand Met during the day(MW)"].mean().index.to_list(),
               y=df.groupby(["MonthName"])["Max.Demand Met during the day(MW)"].mean())

show_values(a)
plt.xlabel("Date")
plt.ylabel("Max.Demand Met during the day(MW)")
plt.xticks(rotation=0)
plt.title("Power Generated in MW")
plt.show()

```



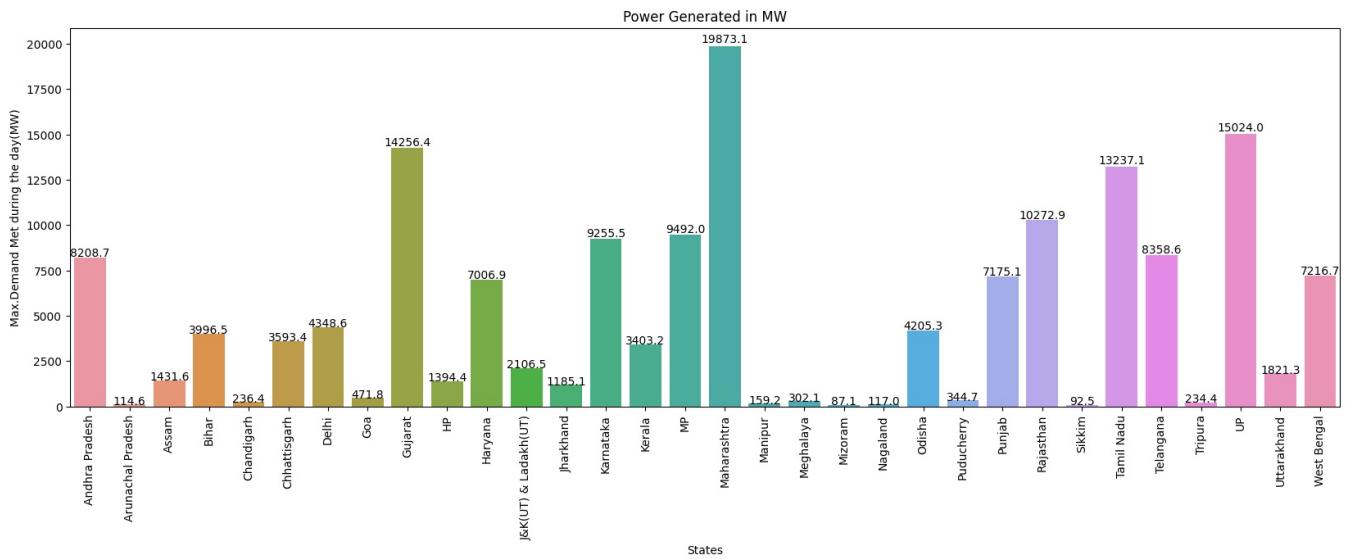
In [17]:

```

plt.figure(figsize=(20,6))
a=sns.barplot(x=df.groupby(["States"])["Max.Demand Met during the day(MW)"].mean().index.to_list(),
               y=df.groupby(["States"])["Max.Demand Met during the day(MW)"].mean())

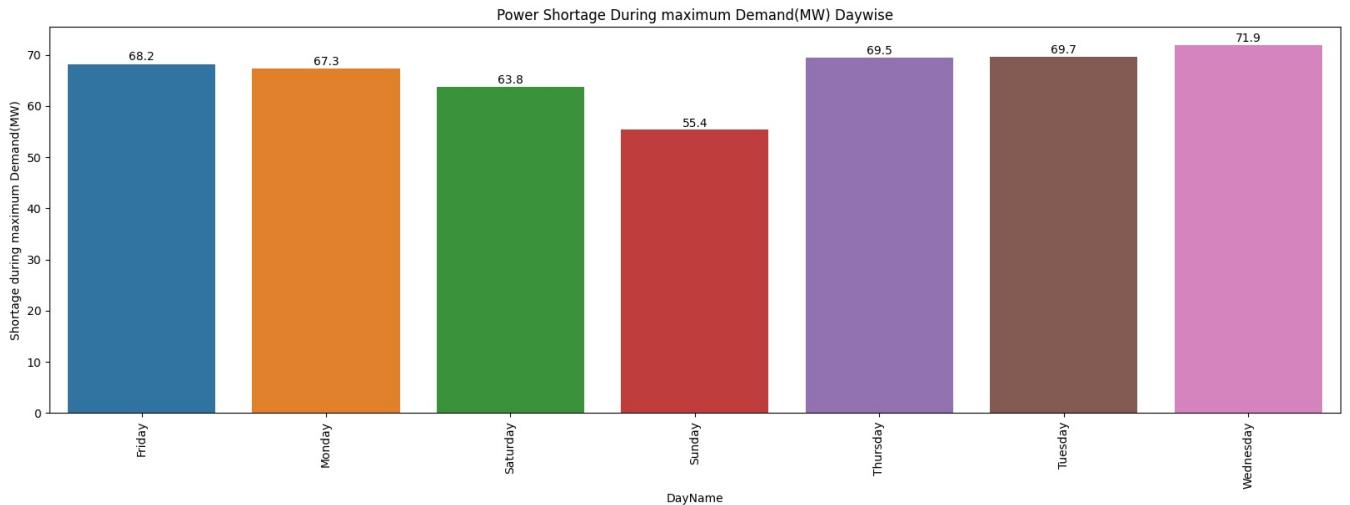
show_values(a)
plt.xlabel("States")
plt.ylabel("Max.Demand Met during the day(MW)")
plt.xticks(rotation=90)
plt.title("Power Generated in MW")
plt.show()

```



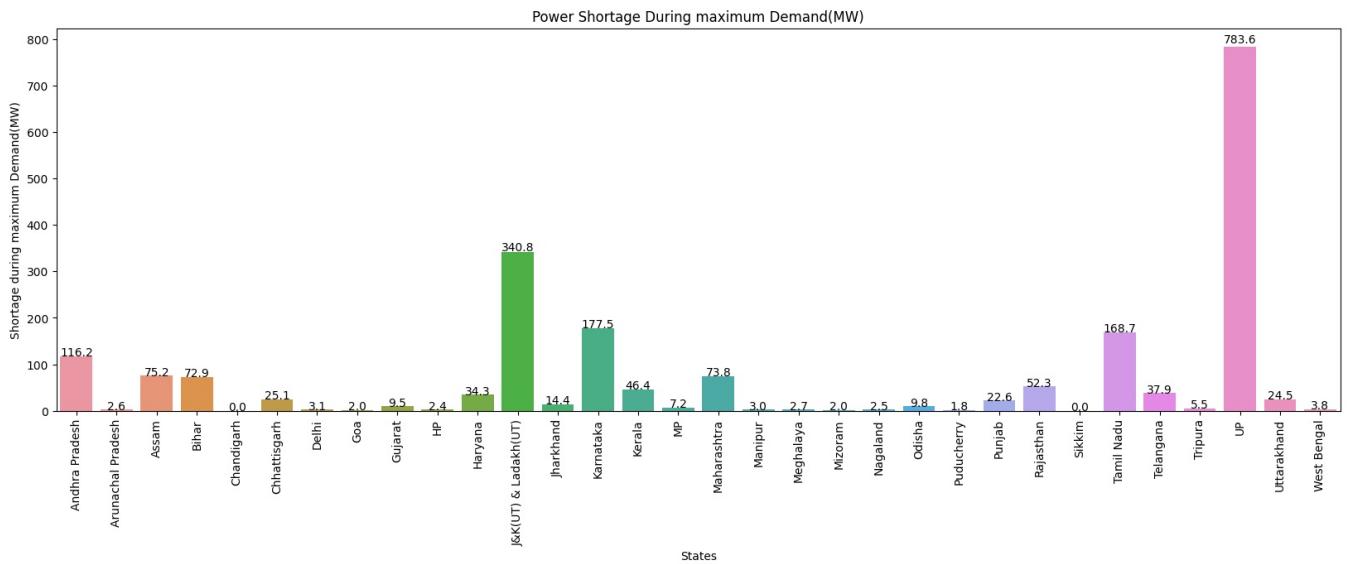
```
In [18]: plt.figure(figsize=(20,6))
a=sns.barplot(x=df.groupby(["DayName"])["Shortage during maximum Demand(MW)"].mean().index.to_list(),
               y=df.groupby(["DayName"])["Shortage during maximum Demand(MW)"].mean())

show_values(a)
plt.xlabel("DayName")
plt.ylabel("Shortage during maximum Demand(MW)")
plt.xticks(rotation=90)
plt.title("Power Shortage During maximum Demand(MW) Daywise")
plt.show()
```



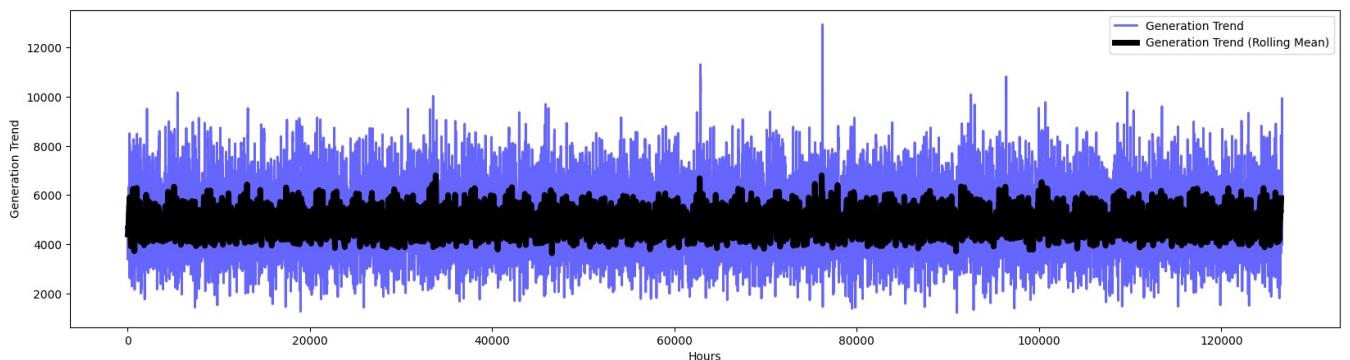
```
In [19]: plt.figure(figsize=(20,6))
a=sns.barplot(x=df.groupby(["States"])["Shortage during maximum Demand(MW)"].mean().index.to_list(),
               y=df.groupby(["States"])["Shortage during maximum Demand(MW)"].mean())

show_values(a)
plt.xlabel("States")
plt.ylabel("Shortage during maximum Demand(MW)")
plt.xticks(rotation=90)
plt.title("Power Shortage During maximum Demand(MW) ")
plt.show()
```



```
In [20]: from statsmodels.tsa.seasonal import seasonal_decompose
decomposed_results = seasonal_decompose(df["Max.Demand Met during the day(MW)"], period=24)

fig, ax = plt.subplots(figsize=(20, 5))
ax.plot(decomposed_results.trend, alpha=0.6, color='blue', label='Generation Trend', linewidth = 2.0)
ax.plot(decomposed_results.trend.rolling(100).mean().shift(-100), alpha=1, color='black', label='Generation Trend (Rolling Mean)')
ax.legend(loc='best')
ax.set_ylabel("Generation Trend")
ax.set_xlabel("Hours")
plt.show()
```



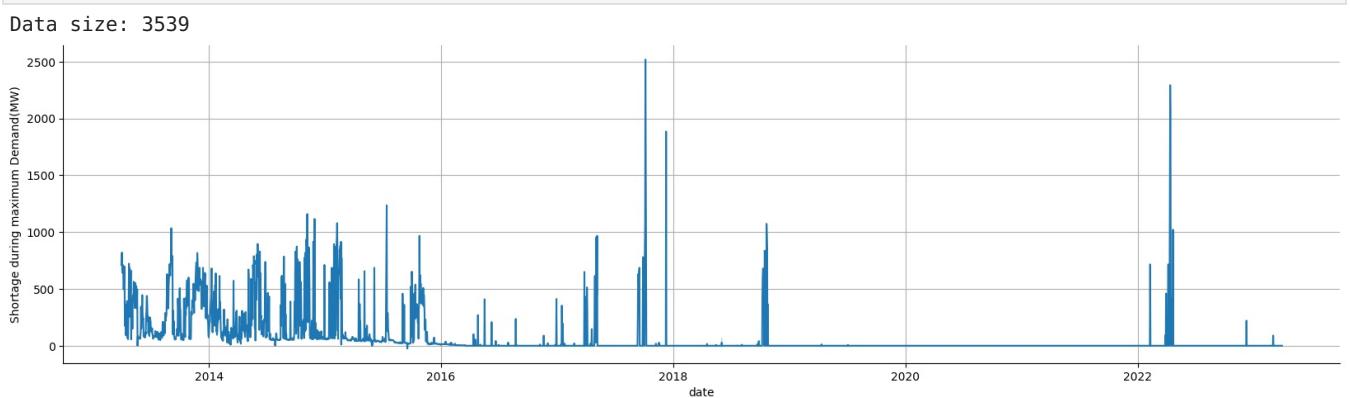
```
In [21]: maha = df[df['States'] == 'Maharashtra']
maha.head(2)
```

```
Out[21]:
```

Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
3	WR Maharashtra	14837	57.0	315.0	2015-01-01	Thursday	January	2015	1	1	Winter
38	WR Maharashtra	18950	0.0	381.7	2016-01-01	Friday	January	2016	1	1	Winter

```
In [22]: maha['date'] = pd.to_datetime(maha['date'])
print(f"Data size: {len(maha)}")

fig, ax = plt.subplots(figsize=(20,5))
sns.lineplot(data=maha, x="date", y="Shortage during maximum Demand(MW)", ci='sd')
sns.despine()
plt.grid()
```



In [23]:

```
# Convert to datetime format
df_train_sesional = df[['date','Max.Demand Met during the day(MW)', 'Shortage during maximum Demand(MW)']].copy()
df_train_sesional['Date'] = pd.to_datetime(df_train_sesional['date'])
# Set date column as index
df_train_sesional.set_index('Date', inplace=True)

# Resample Monthly average power and plot
plt.subplots(4,1,figsize=(20,20))

plt.subplot(411)
df_train_sesional.resample('Y').mean()['Max.Demand Met during the day(MW)'].plot(ylabel='Max.Demand Met during the day(MW)', fontsize=20,color='red');

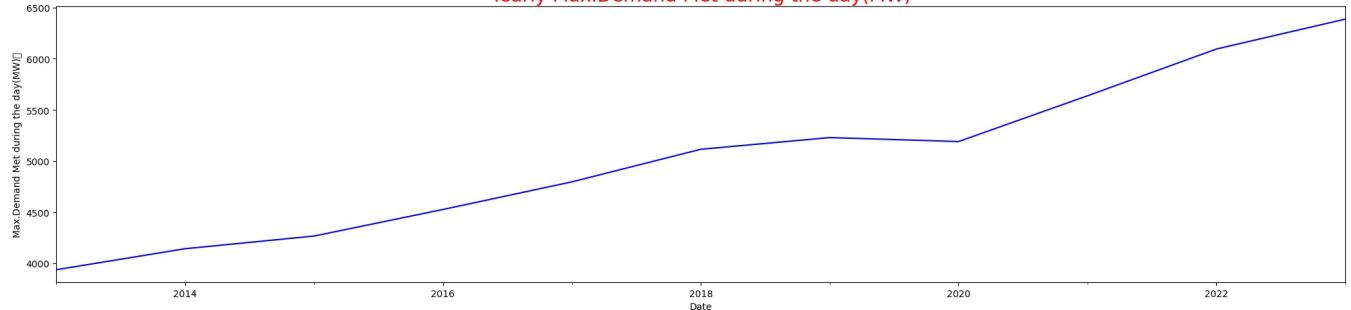
plt.subplot(412)
df_train_sesional.resample('M').mean()['Max.Demand Met during the day(MW)'].plot(ylabel='Max.Demand Met during the day(MW)', fontsize=20,color='red');

plt.subplot(413)
df_train_sesional.resample('W').mean()['Max.Demand Met during the day(MW)'].plot(ylabel='Max.Demand Met during the day(MW)', fontsize=20,color='Brown');

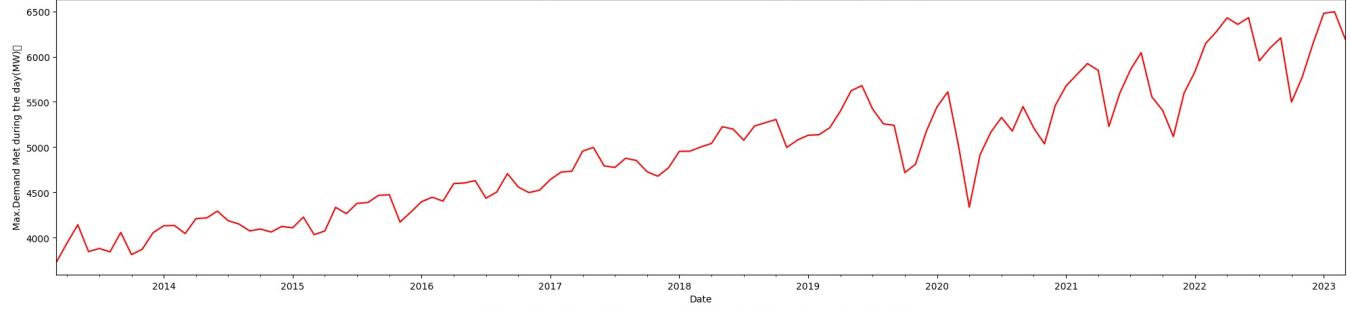
plt.subplot(414)
df_train_sesional.resample('D').mean()['Max.Demand Met during the day(MW)'].plot(ylabel='Max.Demand Met during the day(MW)', fontsize=20,color='DarkGreen');

plt.tight_layout();
```

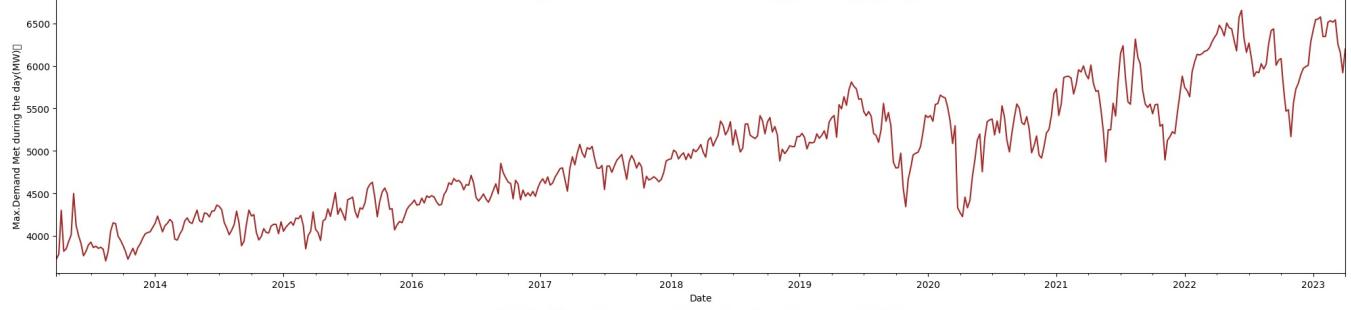
Yearly Max.Demand Met during the day(MW)



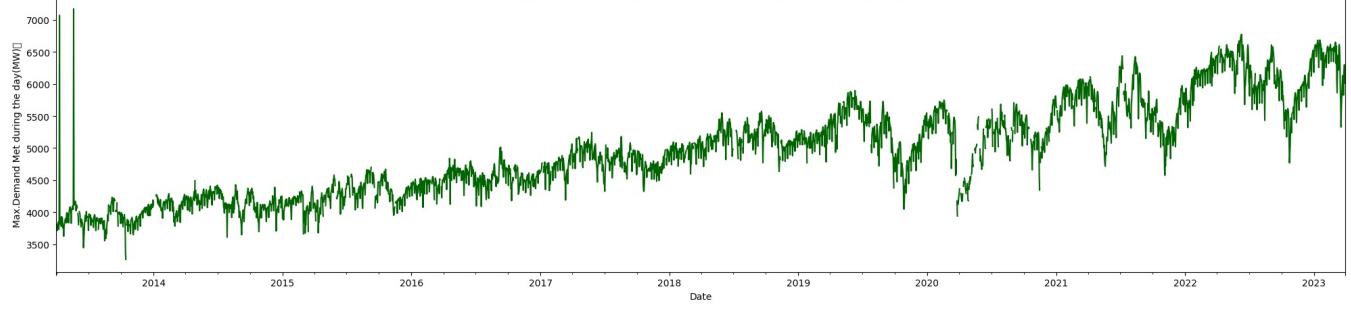
Monthly Max.Demand Met during the day(MW)



Weekly Max.Demand Met during the day(MW)



Daily Max.Demand Met during the day(MW)



In [24]:

```
# Resample Monthly average power and plot
plt.subplots(4,1,figsize=(20,20))

plt.subplot(411)
df_train_sesional.resample('Y').mean()['Shortage during maximum Demand(MW)'].plot(ylabel='Shortage during maximum Demand(MW)', fontsize=20,color='blue');
```

```

plt.title("Yearly Max.Demand Met during the day(MW)", fontsize=20,color='red');

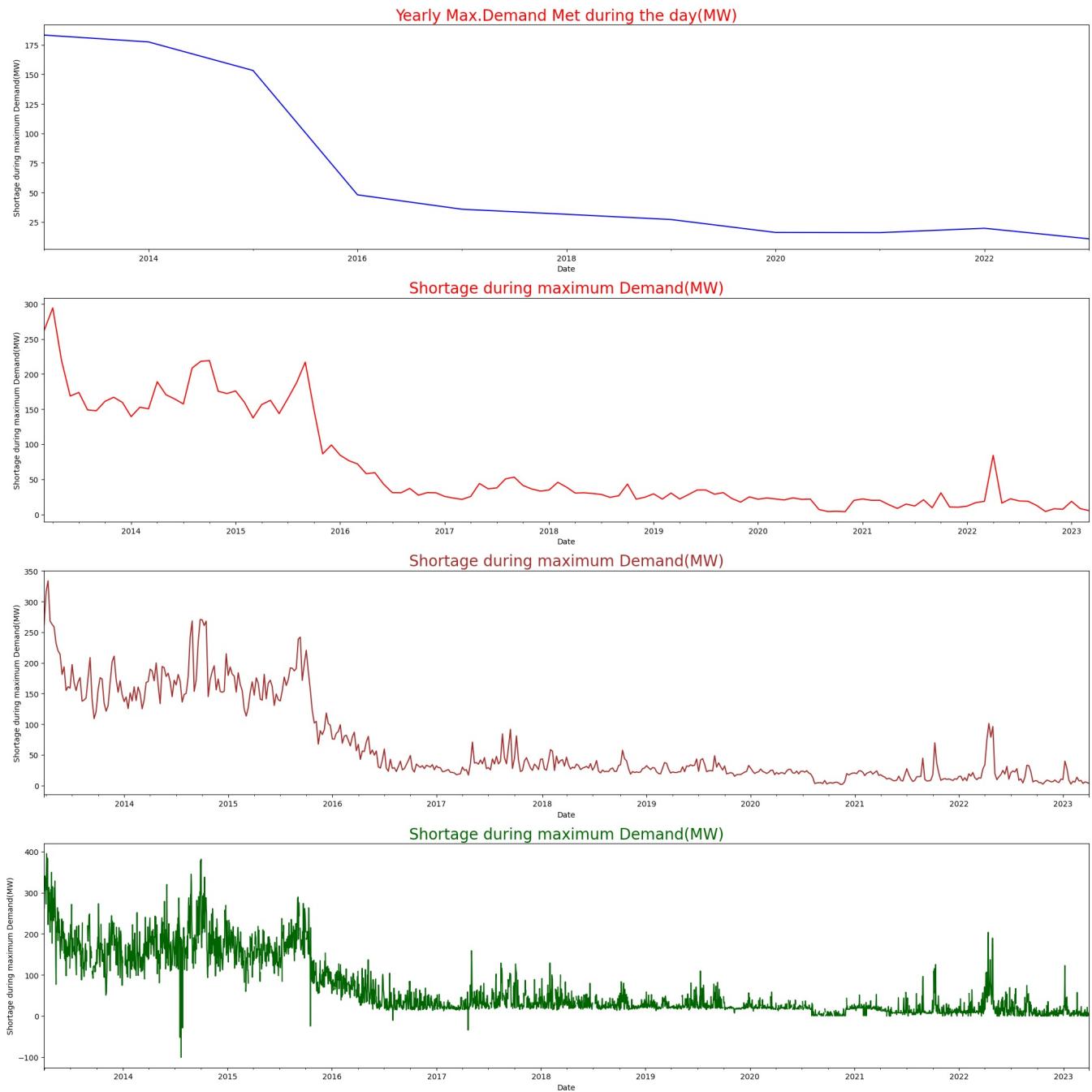
plt.subplot(412)
df_train_sesional.resample('M').mean()['Shortage during maximum Demand(MW)'].plot(ylabel='Shortage during maximum Demand(MW)', fontsize=20,color='red');

plt.subplot(413)
df_train_sesional.resample('W').mean()['Shortage during maximum Demand(MW)'].plot(ylabel='Shortage during maximum Demand(MW)', fontsize=20,color='Brown');

plt.subplot(414)
df_train_sesional.resample('D').mean()['Shortage during maximum Demand(MW)'].plot(ylabel='Shortage during maximum Demand(MW)', fontsize=20,color='DarkGreen');

plt.tight_layout();

```

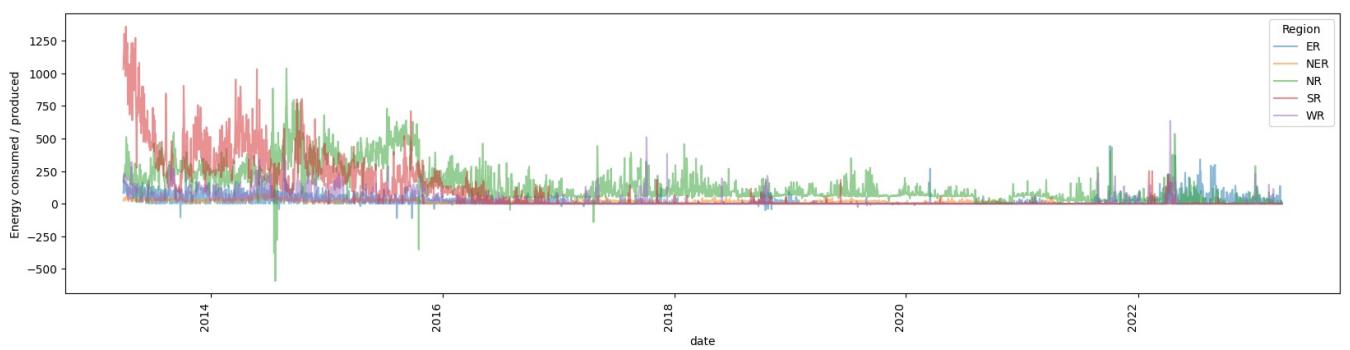


```

In [25]: train_avgd = (
    df.groupby(['date','Region'])['Shortage during maximum Demand(MW)'].mean().unstack()
)

fig, ax = plt.subplots(1, 1, figsize=(20, 5))
_ = train_avgd.plot(ax=ax, alpha=0.5)
_ = ax.set_ylabel('Energy consumed / produced')
plt.xticks(rotation=90);

```



In [26]: `df.head(2)`

Out[26]:

	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
0	NER	Mizoram	77	1.0	1.2	2015-01-01	Thursday	January	2015	1	1	Winter
2	WR	Goa	383	0.0	7.3	2015-01-01	Thursday	January	2015	1	1	Winter

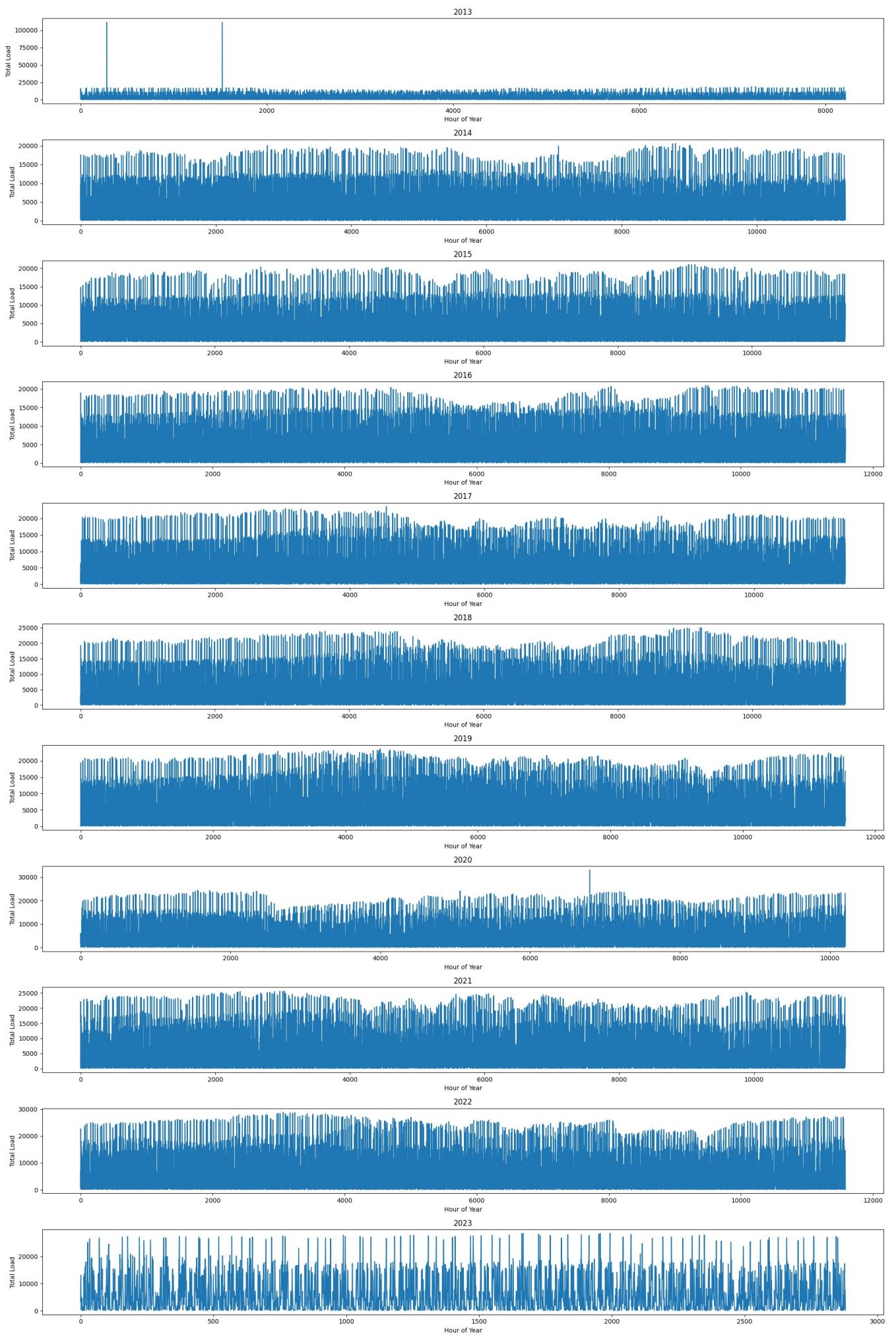
In [27]: `data = df[['Max.Demand Met during the day(MW)', 'date']]  
data['date'] = pd.to_datetime(data['date'])  
data = data.set_index('date')  
data.head()`

Out[27]:

Max.Demand Met during the day(MW)

date	Max.Demand Met during the day(MW)
2015-01-01	77
2015-01-01	383
2015-01-01	14837
2015-01-01	5740
2015-01-01	11383

In [28]: `groups = data['Max.Demand Met during the day(MW)'].groupby(pd.Grouper(freq='A'))  
fig, axs = plt.subplots(len(groups), 1, figsize=(20,30))  
  
for ax, (name, group) in zip(axs, groups):  
 ax.plot(pd.Series(group.values))  
 ax.set_xlabel('Hour of Year')  
 ax.set_ylabel('Total Load')  
 ax.set_title(name.year)  
 plt.subplots_adjust(hspace=0.5);  
 plt.tight_layout()`



## Time Series Analysis of Maharashtra Power Generation with Fb-Prophet

```
In [29]: maha = df[df['States'] == 'Maharashtra']
maha = maha.sort_values(by='date', ascending=True)
display(maha.shape)
maha.head()
```

(3539, 12)

Out[29]:

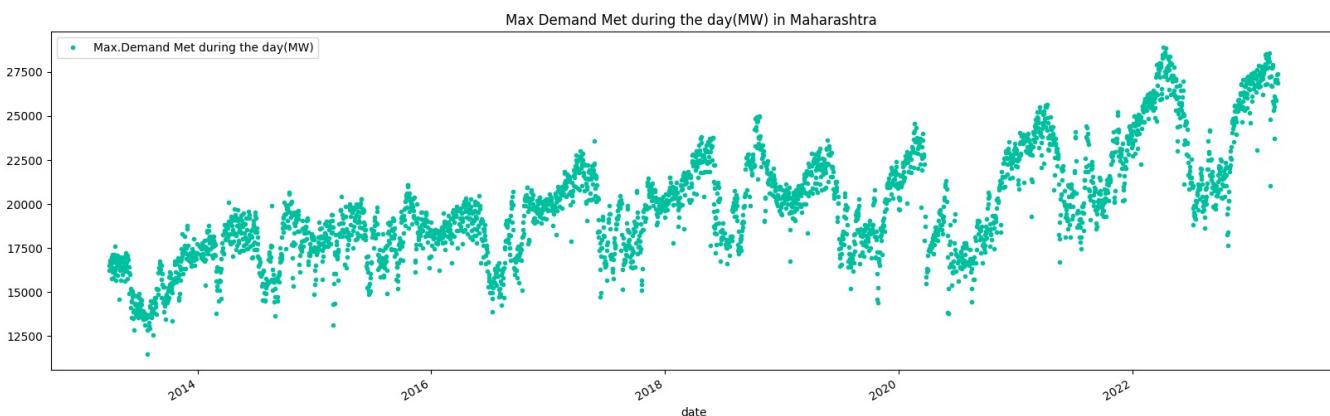
	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
124734	WR	Maharashtra	16196	712.0	337.8	2013-03-31	Sunday	March	2013	1	3	Spring
1140	WR	Maharashtra	16515	765.0	357.5	2013-04-01	Monday	April	2013	2	4	Spring
5193	WR	Maharashtra	16778	820.0	365.0	2013-04-02	Tuesday	April	2013	2	4	Spring
9230	WR	Maharashtra	16697	645.0	366.5	2013-04-03	Wednesday	April	2013	2	4	Spring
13235	WR	Maharashtra	16574	642.0	362.5	2013-04-04	Thursday	April	2013	2	4	Spring

In [30]:

```
# Find the indices of the outliers
outliers = maha[maha['Max.Demand Met during the day(MW)'] < 6000].index
# Remove the outliers from the dataframe
maha = maha.drop(outliers)

# Color palette for plotting
df3 = maha[['date', 'Max.Demand Met during the day(MW)']].copy('deep')
df3['date'] = pd.to_datetime(df3['date'])
df3 = df3.set_index('date')

color_pal = ['#F8766D', '#D39200', '#93AA00', '#00BA38', '#00C19F', '#00B9E3', '#619cff', '#DB72FB']
df3.plot(style='.', figsize=(20,6), color=color_pal[4], title='Max Demand Met during the day(MW) in Maharashtra')
plt.ylim();
plt.show()
```



In [31]:

```
from statsmodels.tsa.seasonal import seasonal_decompose

def seasonal_decompose_plotter(df: pd.DataFrame, period=12, title='', figsize=(20, 12)):
    """
    Perform and plot seasonal decomposition of a time series.

    Parameters:
        df: DataFrame with time series data.
        col: Column name for data to decompose. Default is 'sqrt(O3 AQI)'.
        date_col: Column name for datetime values. Default is 'Date'.
        period: Seasonality period. Default is 12.

    Returns:
        A DecomposeResult object with seasonal, trend, and residual components.
    """

    # Decomposition
    decomposition = seasonal_decompose(df.values, period=period)
    de_season = decomposition.seasonal
    de_resid = decomposition.resid
    de_trend = decomposition.trend

    fig, ax = plt.subplots(4, sharex=True, figsize=figsize)

    ax[0].set_title(title)
    ax[0].plot(df.index, df.values, color='C3')
    ax[0].set_ylabel(df.keys()[0])
    ax[0].grid(alpha=0.25)

    ax[1].plot(df.index, de_trend, color='C1')
    ax[1].set_ylabel('Trend')
    ax[1].grid(alpha=0.25)

    ax[2].plot(df.index, de_season, color='C2')
    ax[2].set_ylabel('Seasonal')
    ax[2].grid(alpha=0.25)

    ax[3].plot(df.index, de_resid, color='C4')
    ax[3].set_ylabel('Residual')
    ax[3].grid(alpha=0.25)
```

```

        ax[2].plot(df.index, de_season, color='C2')
        ax[2].set_ylabel('Seasonal')
        ax[2].grid(alpha=0.25)

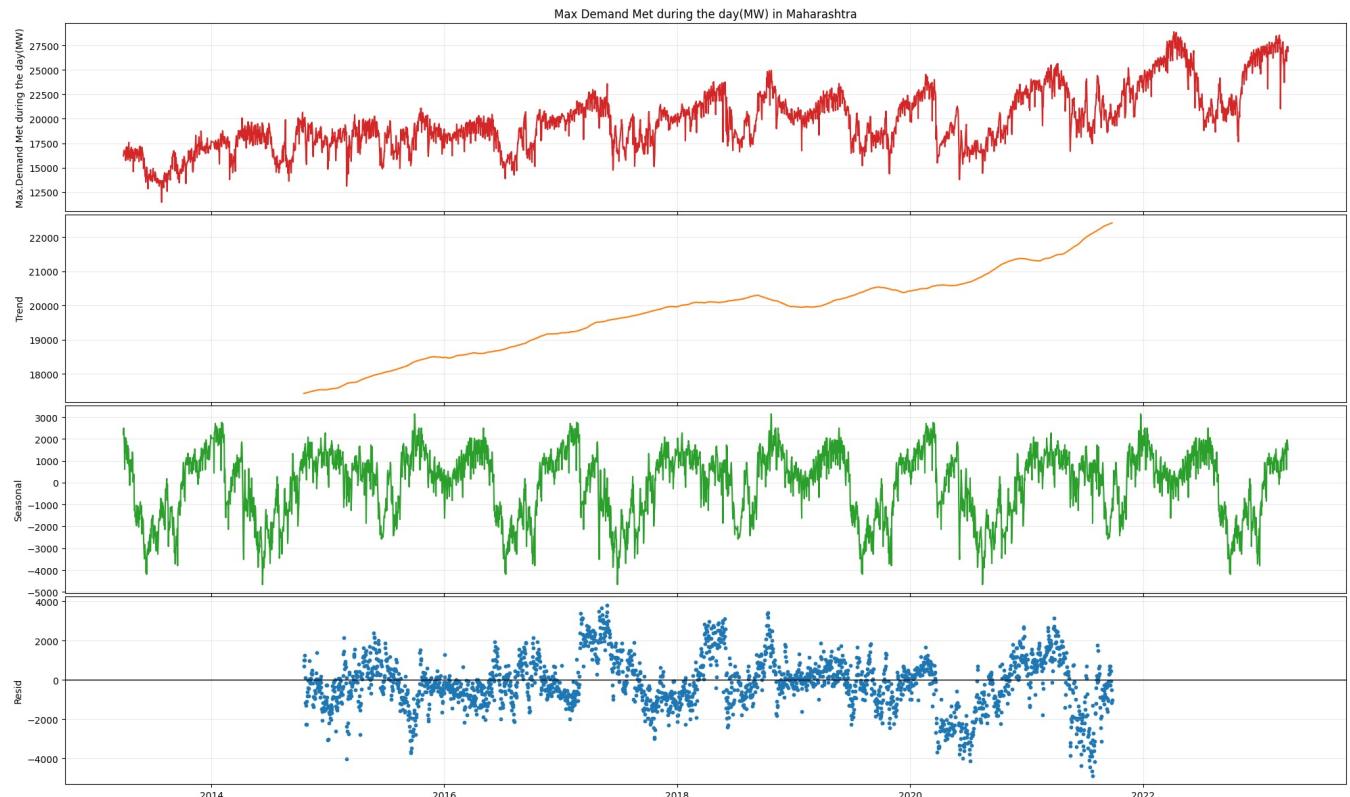
        ax[3].axhline(y=0, color='k', linewidth=1)
        ax[3].scatter(df.index, de_resid, color='C0', s=10)
        ax[3].set_ylabel('Resid')
        ax[3].grid(alpha=0.25)

    plt.tight_layout(h_pad=0)
    plt.show()

    return decomposition

```

```
_ = seasonal_decompose_plotter(df3, period=365*3, title='Max Demand Met during the day(MW) in Maharashtra', fig
```



In [32]: # Format data for prophet model using ds and y

```

df4 = maha.copy('deep')
df4 = df4[['date','Max.Demand Met during the day(MW)']]
df_final = df4.reset_index().rename(columns={'date':'ds','Max.Demand Met during the day(MW)':'y'})
df_final = df_final.drop('index',axis=1)
df_final['ds'] = pd.to_datetime(df_final['ds'])
df_final['ds'] = df_final['ds'].dt.tz_localize(None)
df_final.head()

```

Out[32]:

	ds	y
0	2013-03-31	16196
1	2013-04-01	16515
2	2013-04-02	16778
3	2013-04-03	16697
4	2013-04-04	16574

In [33]:

```

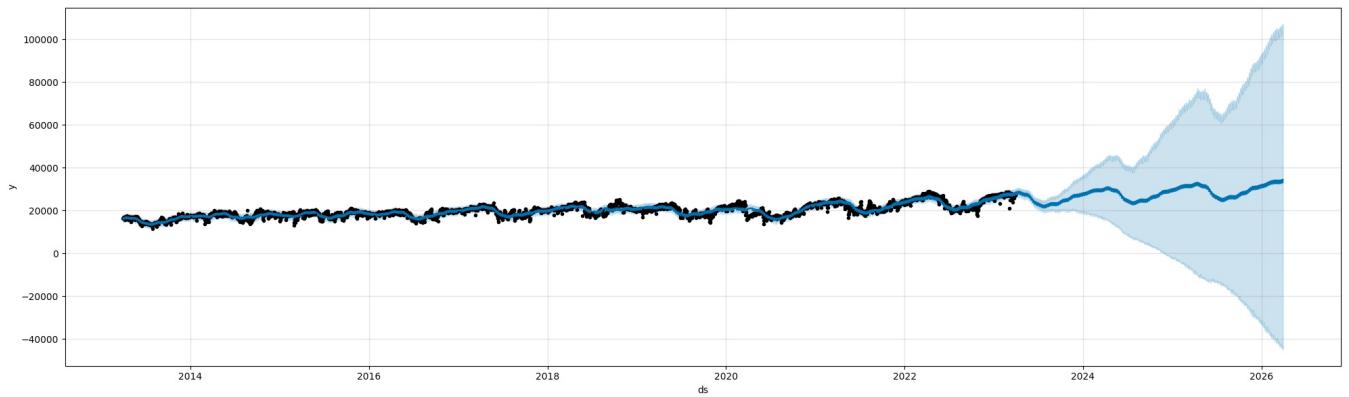
from prophet import Prophet
model = Prophet( daily_seasonality=True,
                 weekly_seasonality=True,
                 yearly_seasonality=True,
                 seasonality_mode='multiplicative',
                 changepoint_prior_scale=0.5,
                 holidays_prior_scale=0.1)

model.fit(df_final)

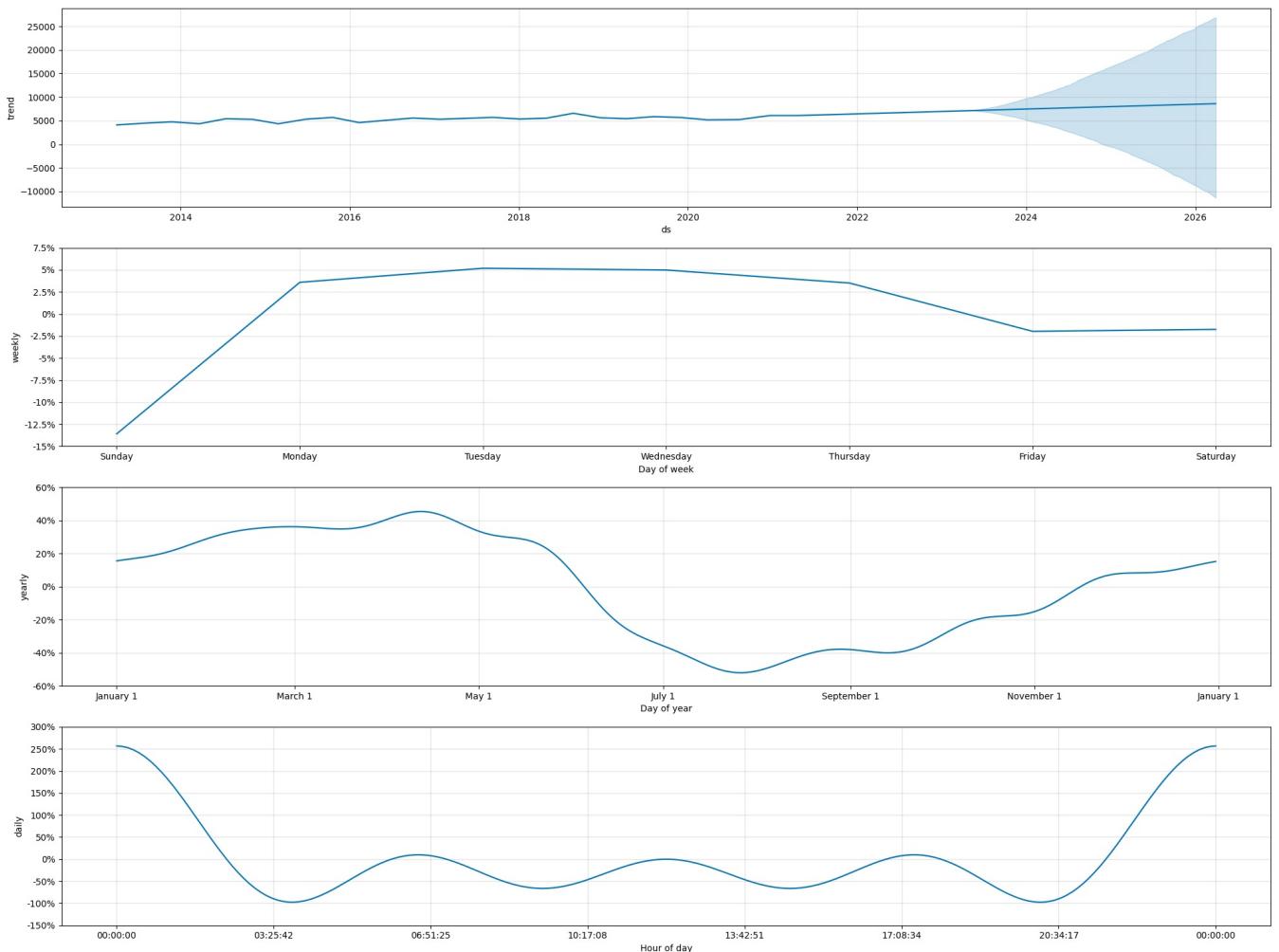
future = model.make_future_dataframe(periods=365*3,freq='D')
forecast = model.predict(future)
fig = model.plot(forecast,figsize=(20, 6))
plt.show()

```

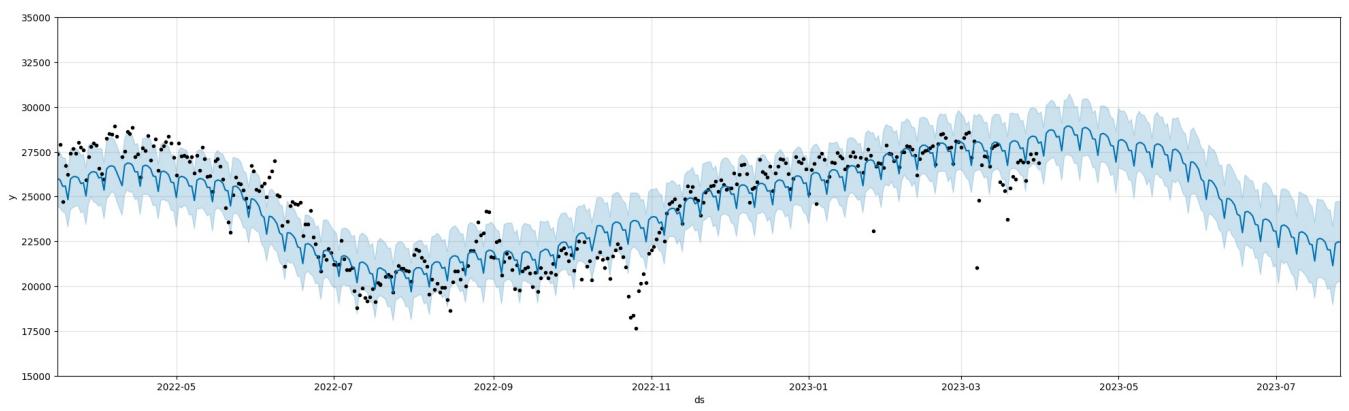
```
14:00:07 - cmdstanpy - INFO - Chain [1] start processing
14:00:11 - cmdstanpy - INFO - Chain [1] done processing
```



```
In [34]: fig3 = model.plot_components(forecast, uncertainty=True, figsize=(20, 15))
plt.show()
```



```
In [35]: fig = model.plot(forecast, figsize=(20, 6))
plt.xlim(pd.to_datetime(['2022-03-16', '2023-07-26']))
plt.ylim(15000, 35000)
plt.show()
```



Black dots are actual values and blue line is the forecast

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js