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()
```

(126699, 6)

Out[2]:

	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()
```

(18251, 8)

Out[3]:

	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
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
```

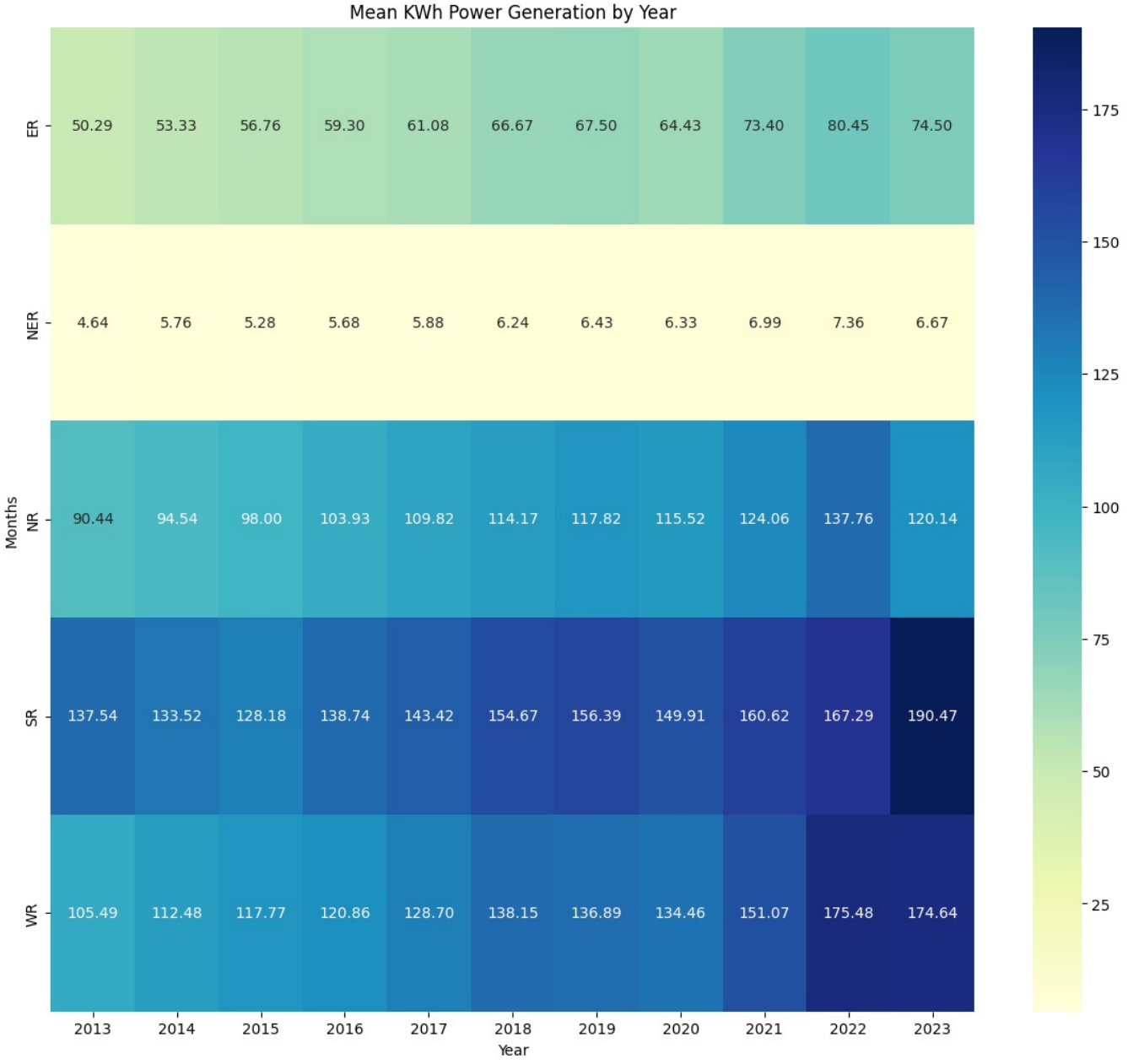
In [6]:
```
df['date'] = pd.to_datetime(df['date'])

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 "Autu


df.head()
```

Out[6]:

	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
```

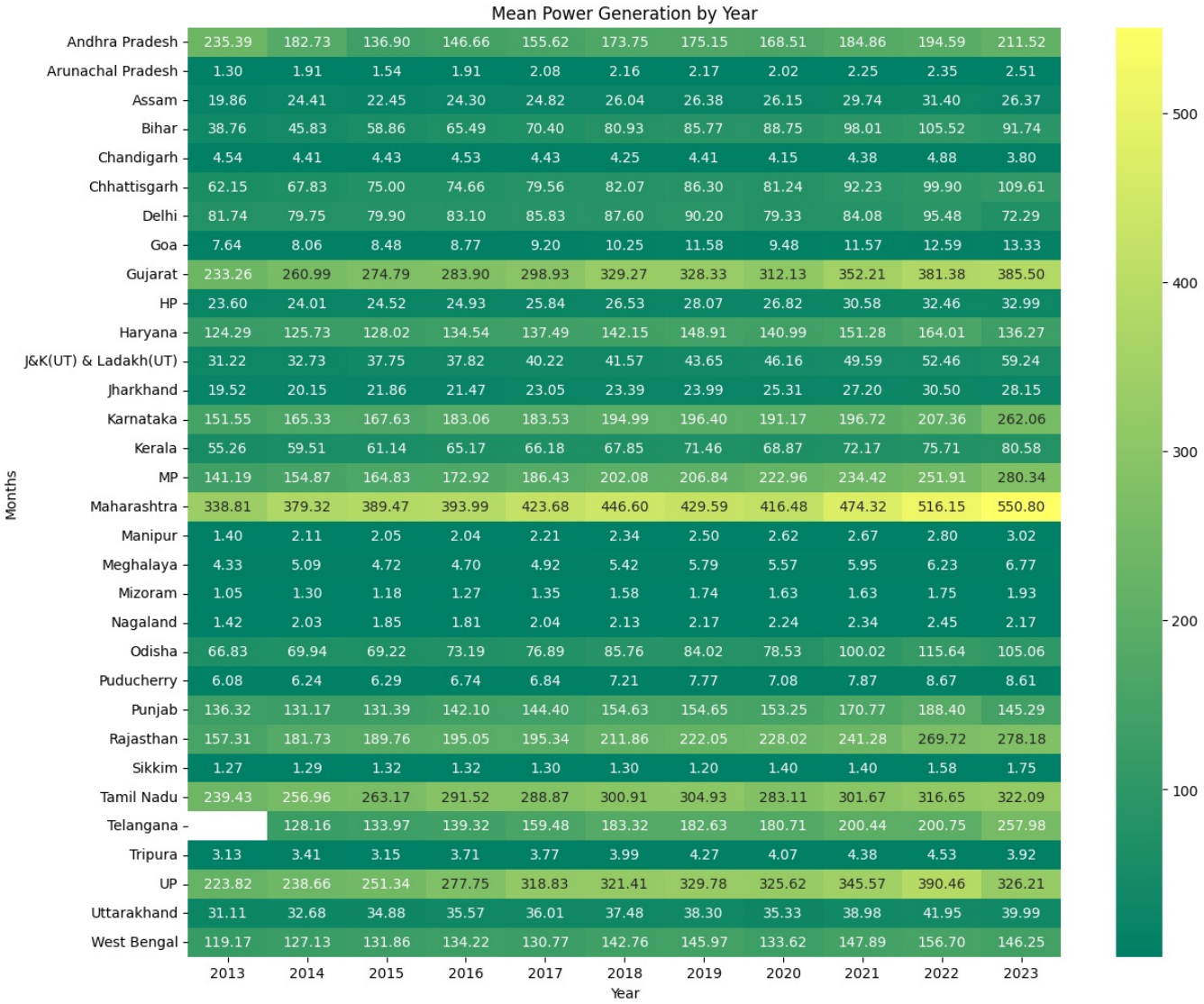
```
pivot_table = df1.pivot_table(values='Energy Met (MU)', index='Region', columns='Year', aggfunc='mean')
plt.figure(figsize=(14, 12))
sns.heatmap(pivot_table, annot=True, fmt=".2f", cmap="YlGnBu")
plt.title('Mean KWh Power Generation by Year')
plt.xlabel('Year')
plt.ylabel('Months')
plt.show()
```



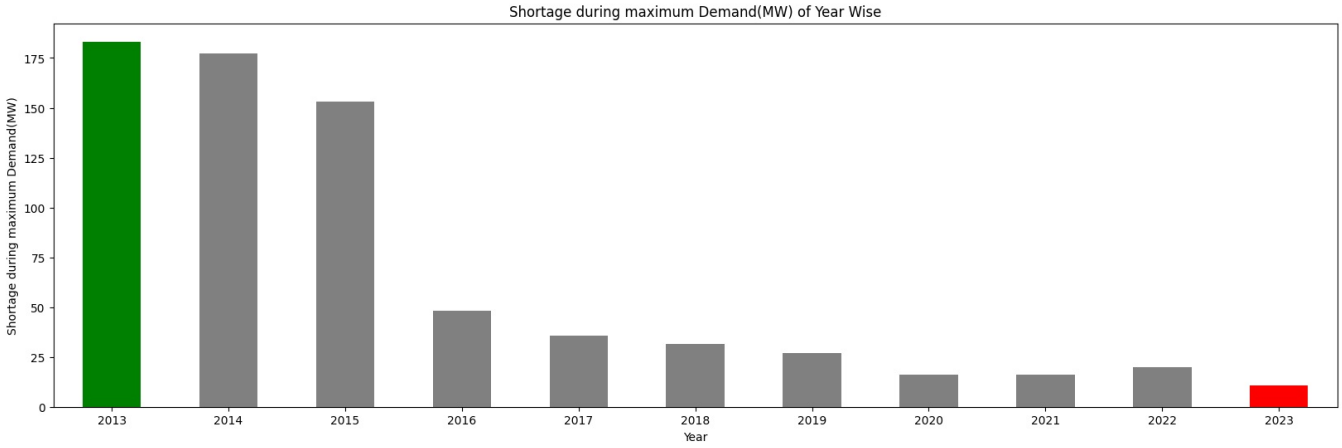
In [8]:
```
names_to_drop = ['AMNSIL', 'DNHDDPDCL', 'BALCO', 'Railways_NR ISTS','DNH','DD','DVC','Essar steel']
df.drop(df[df['States'].isin(names_to_drop)].index, inplace=True)
```

In [9]:
```
df["States"].replace({'J&K(UT) &' : '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'
```

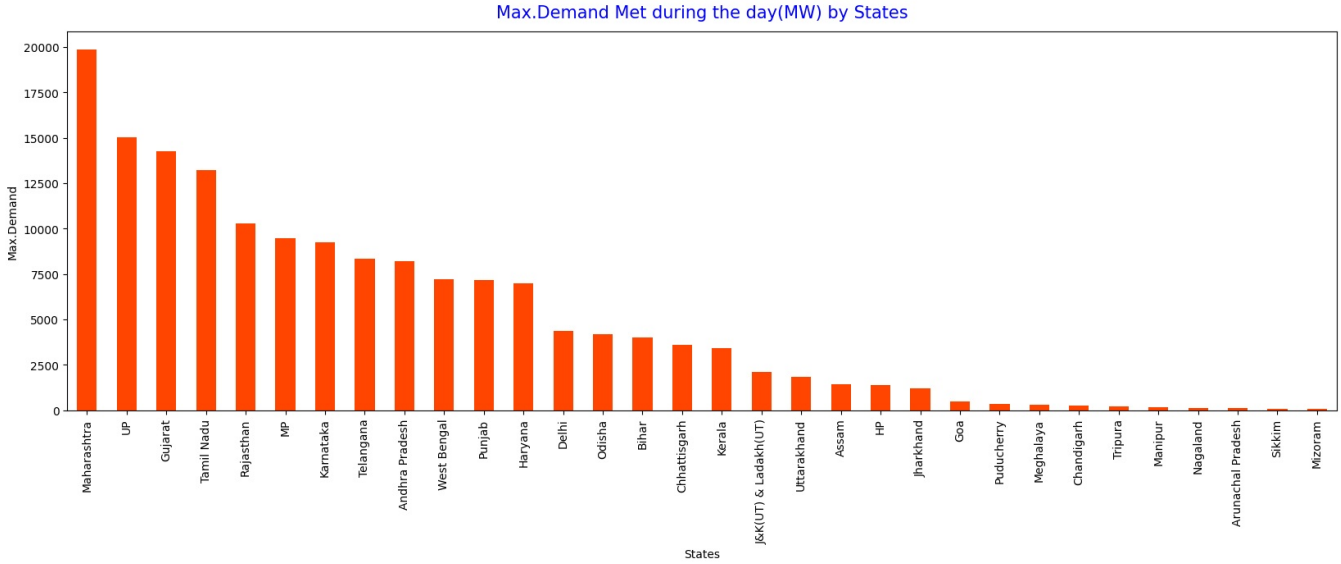
In [10]:
```
# Mean Price by Month

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()
```



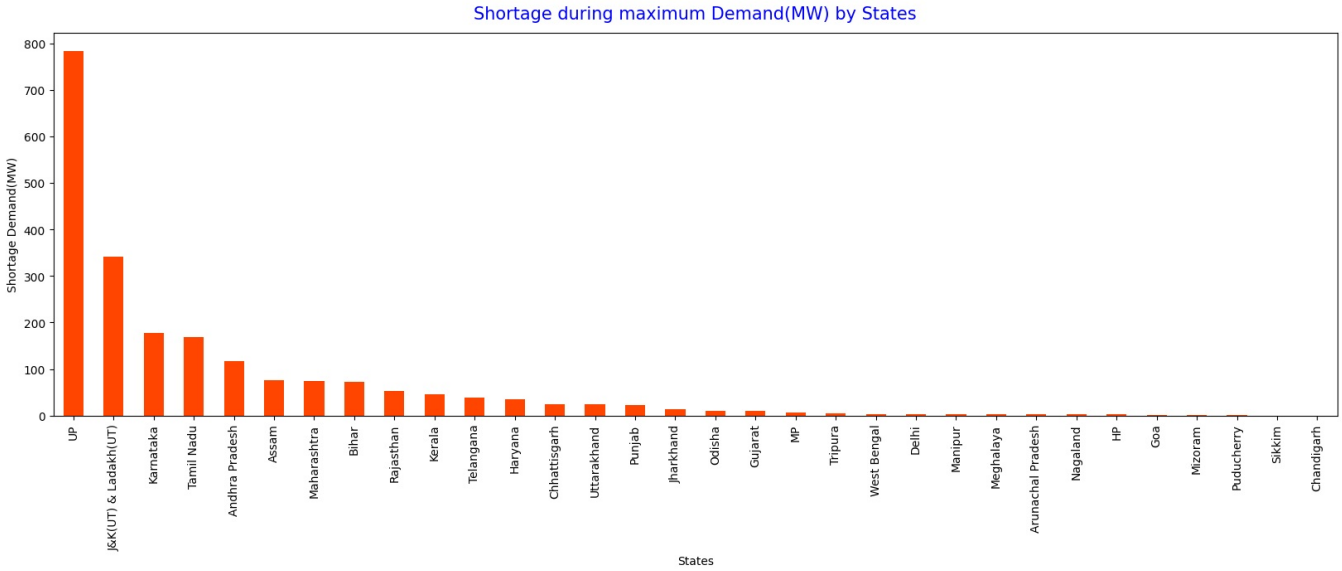
In [11]:
```
plt.figure(figsize=(20, 6))
data = df[['Year','Shortage during maximum Demand(MW)']].copy('deep')
color = list(np.full(12, 'grey'))
color[0], color[10] = 'Green', 'Red'
data.groupby('Year').mean()['Shortage during maximum Demand(MW)'].plot(kind='bar', title='Shortage during maxim
plt.ylabel('Shortage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



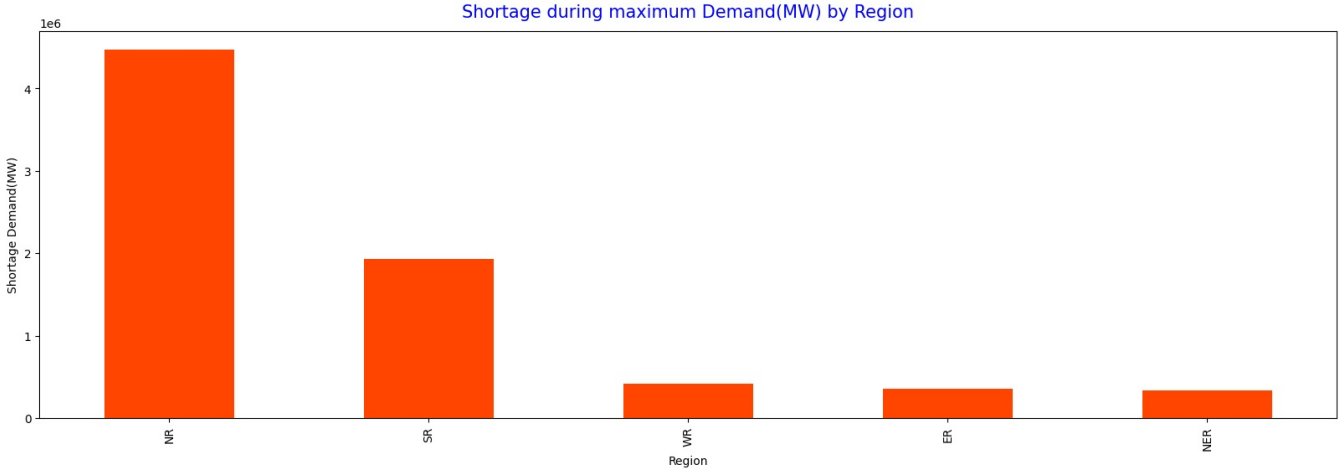
In [12]:
```
plt.figure(figsize=(20, 6))
chart = df.groupby('States')['Max.Demand Met during the day(MW)'].mean().sort_values(ascending = False).plot(ki
chart.set_xticklabels(chart.get_xticklabels(), rotation = 90)
plt.title('Max.Demand Met during the day(MW) by States', fontsize = 15, color = 'b', pad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



In [13]:
```
plt.figure(figsize=(20, 6))
chart = df.groupby('States')['Shortage during maximum Demand(MW)'].mean().sort_values(ascending = False).plot(k
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)');
```



In [14]:
```
plt.figure(figsize=(20, 6))
chart = df.groupby('Region')['Shortage during maximum Demand(MW)'].sum().sort_values(ascending = False).plot(ki
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)');
```



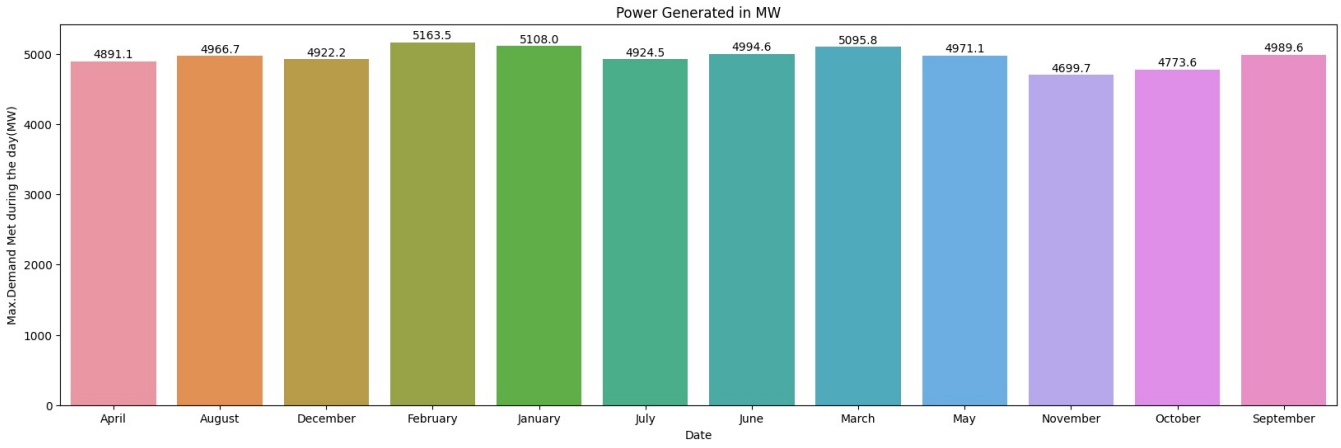
In [15]:
```
df.head(1)
```

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

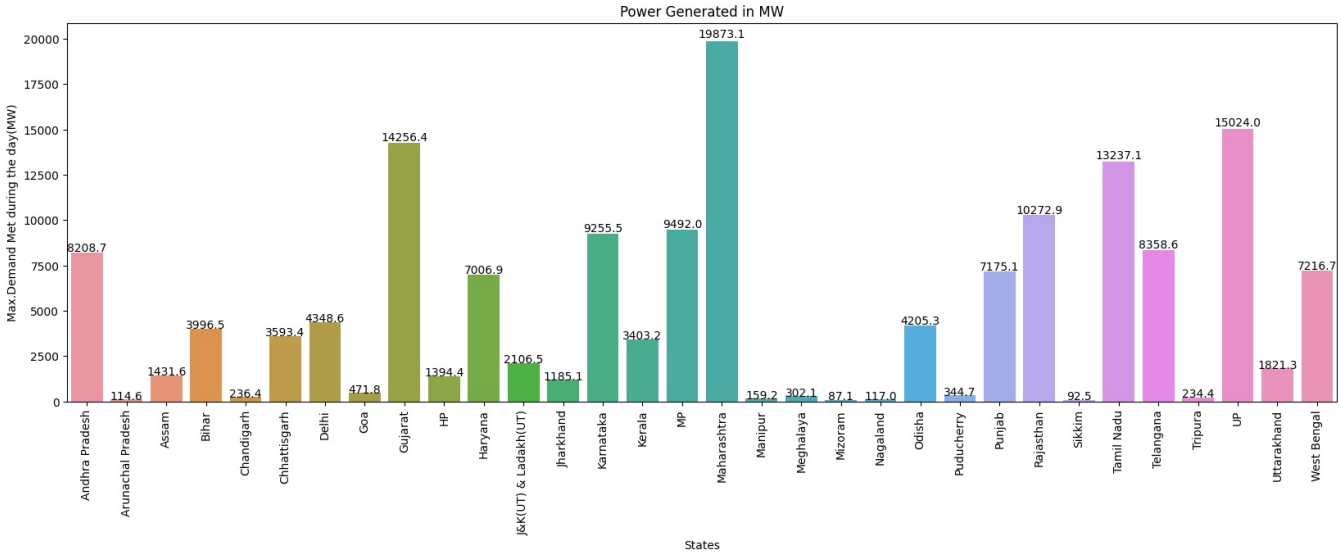
In [16]:

```
def show_values(axs, 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(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _single(ax)
    else:
        _single(axs)


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()
```

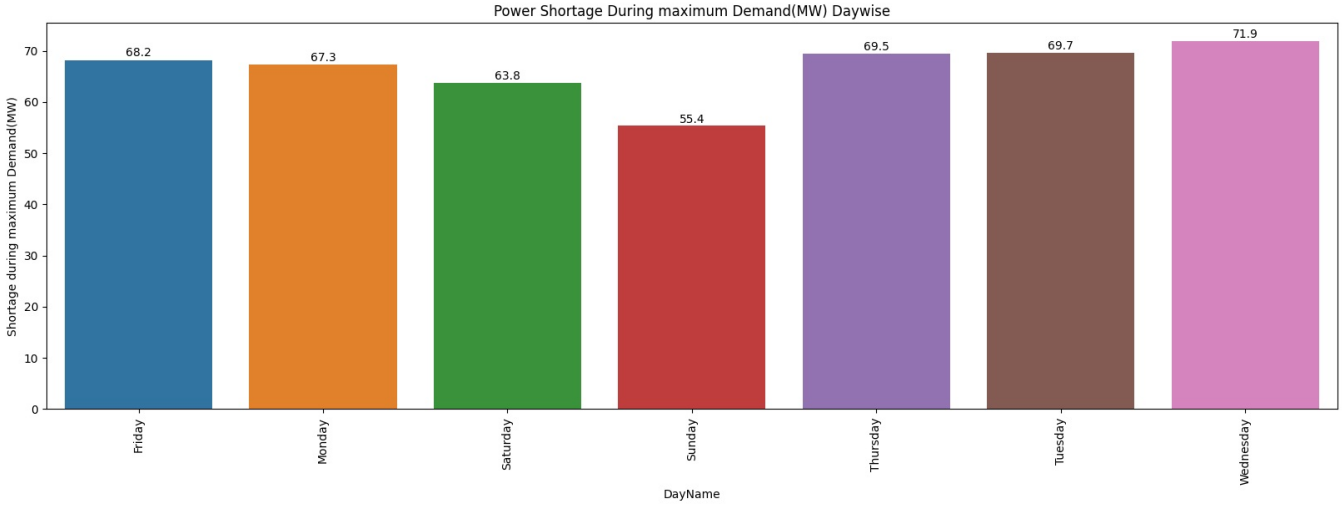


In [17]:

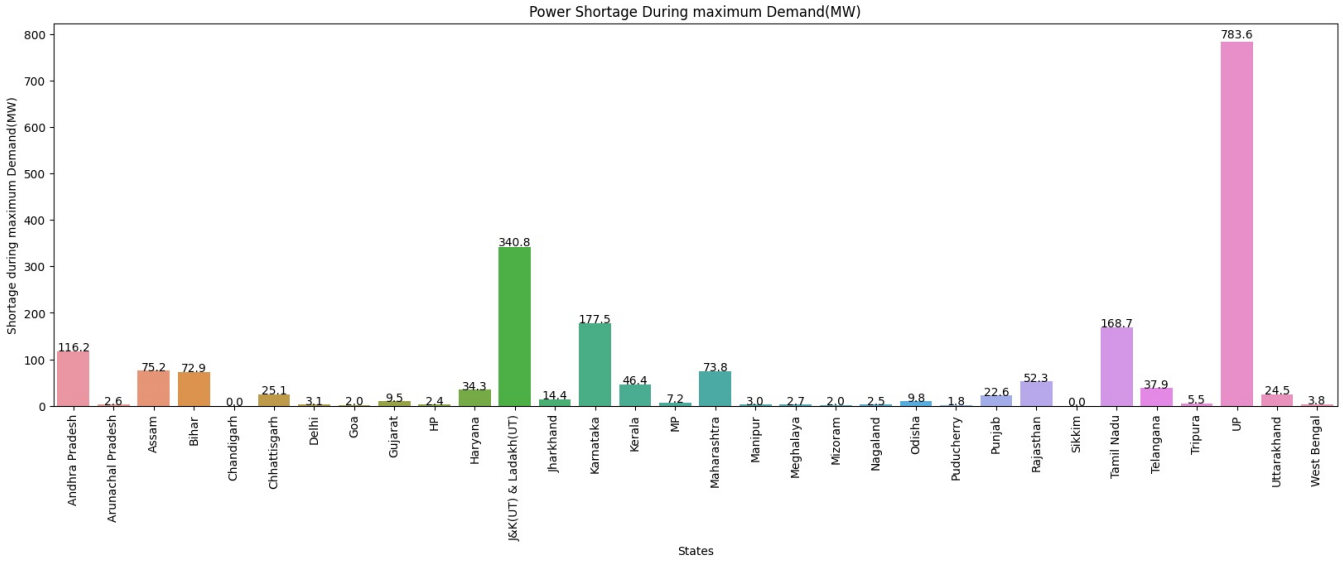
```
plt.figure(figsize=(20,6))
a=sns.barplot(x=df.groupby(["States"])["Max.Demand Met during the day(MW)"].mean().index.to_list(),
            y=df.groupby(["States"])["Max.Demand Met during the day(MW)"].mean())

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()
```



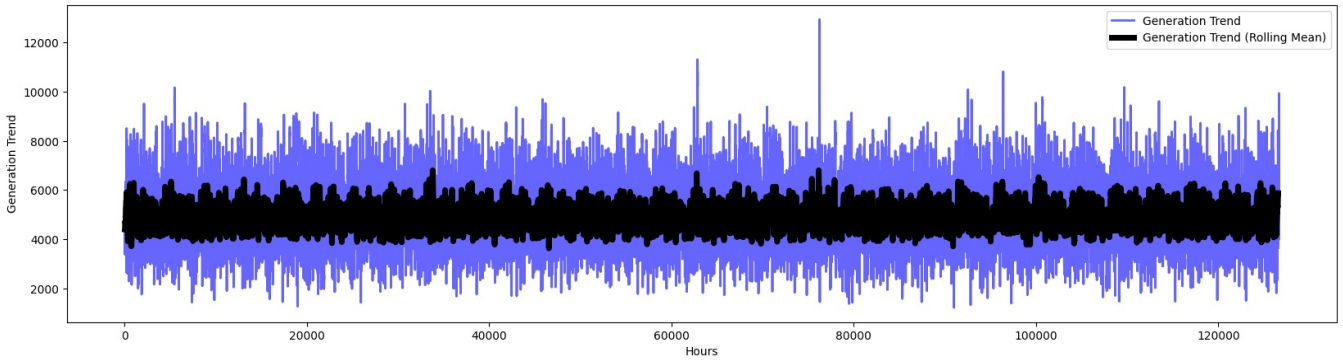
In [18]:
```
plt.figure(figsize=(20,6))
a=sns.barplot(x=df.groupby(["DayName"])["Shortage during maximum Demand(MW)"].mean().index.to_list(),
              y=df.groupby(["DayName"])["Shortage during maximum Demand(MW)"].mean())

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 Tre
ax.legend(loc='best')
ax.set_ylabel("Generation Trend")
ax.set_xlabel("Hours")
plt.show()
```



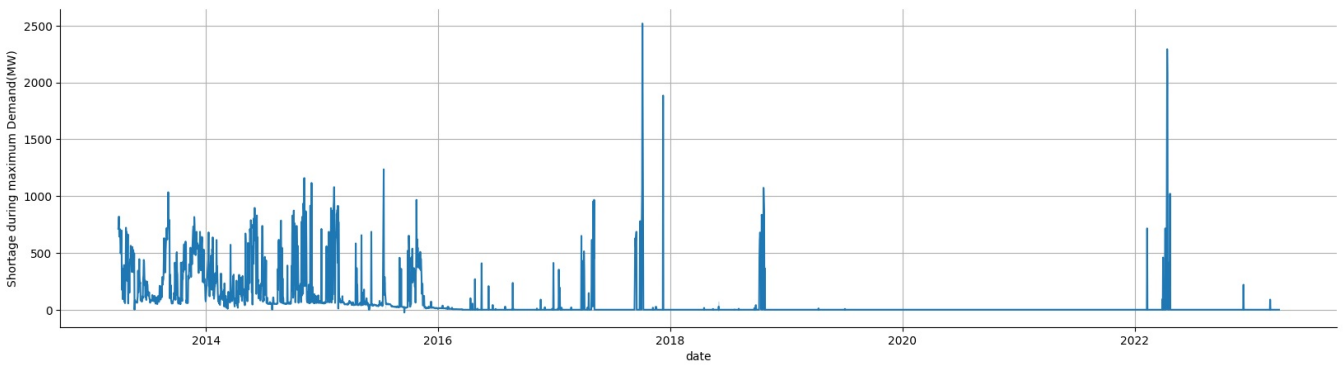
In [21]:
```
maha = df[df['States'] == 'Maharashtra']
maha.head(2)
```

Out[21]:

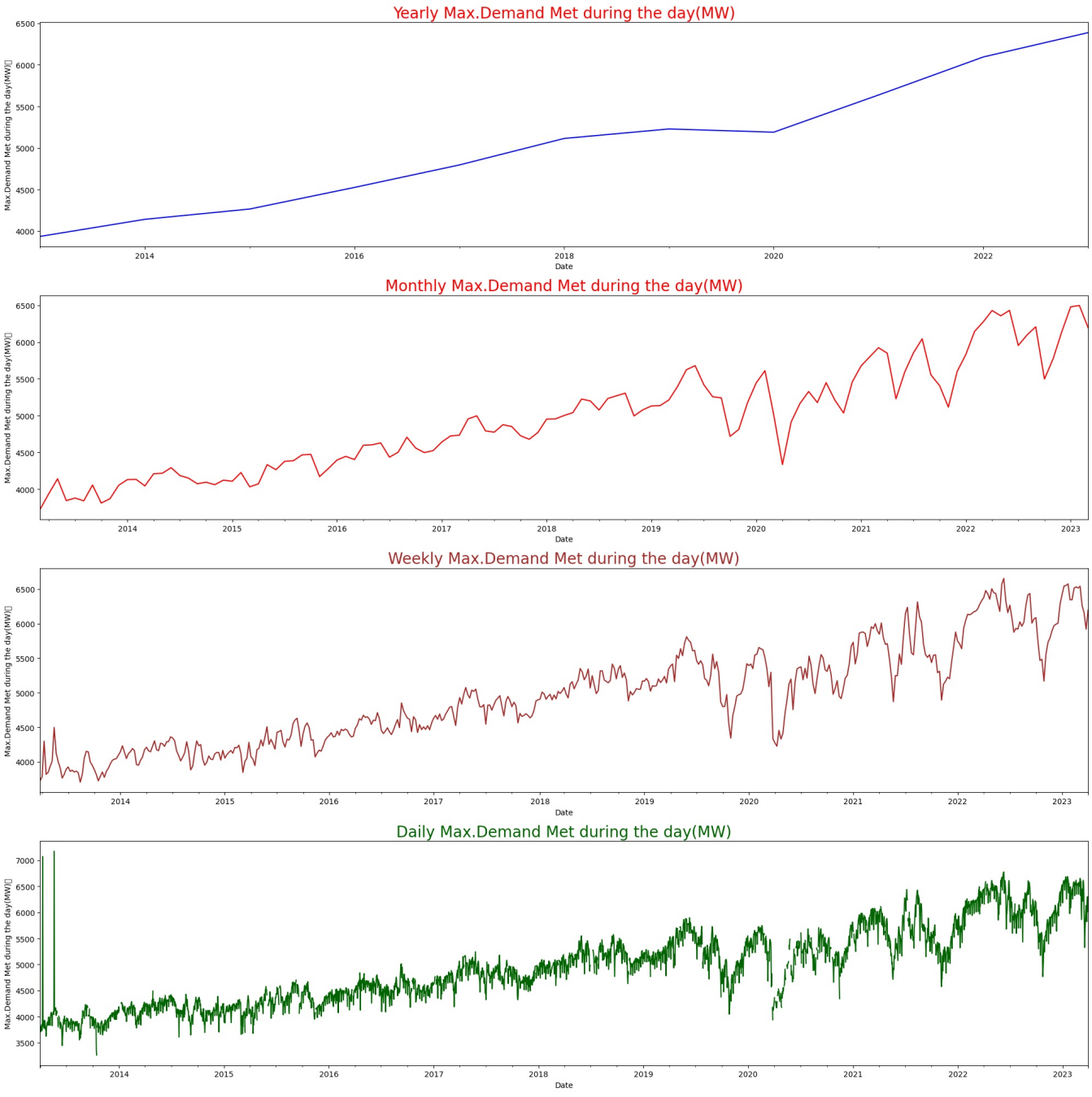
	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
3	WR	Maharashtra	14837	57.0	315.0	2015-01-01	Thursday	January	2015	1	1	Winter
38	WR	Maharashtra	18950	0.0	381.7	2016-01-01	Friday	January	2016	1	1	Winter

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()
```

```
Data size: 3539
```

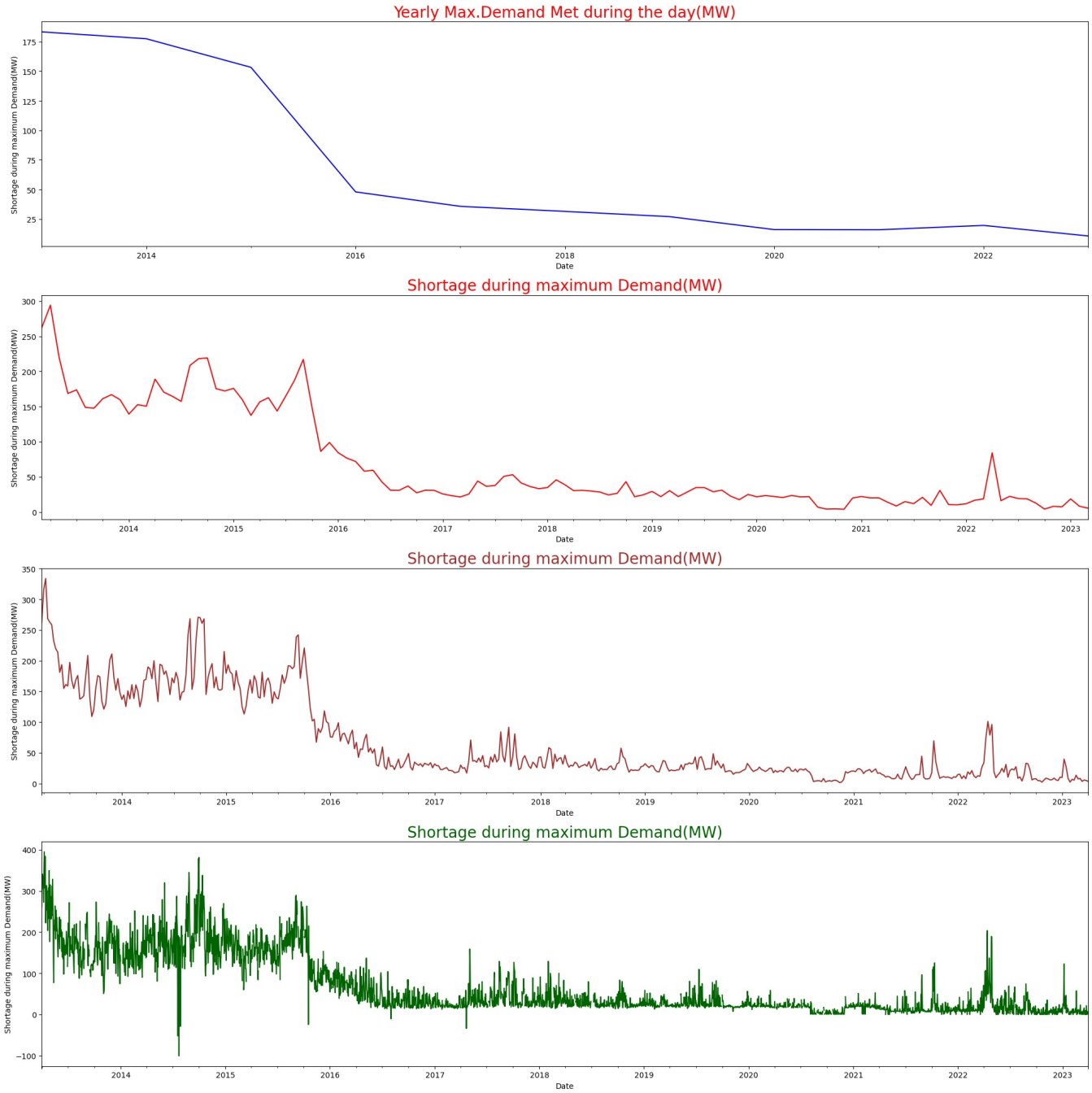


In [23]:
```
# Convert to datetime format
df_train_sesional = df[['date','Max.Demand Met during the day(MW)','Shortage during maximum Demand(MW)']].copy(
df_train_sesional['Date'] = pd.to_datetime(df_train_sesional['date'])
# Set date column as index
df_train_sesional.set_index('Date', inplace=True)

# 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
plt.title("Yearly 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
plt.title("Monthly 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
plt.title("Weekly 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
plt.title("Daily Max.Demand Met during the day(MW)", fontsize=20,color='DarkGreen');


plt.tight_layout();
```



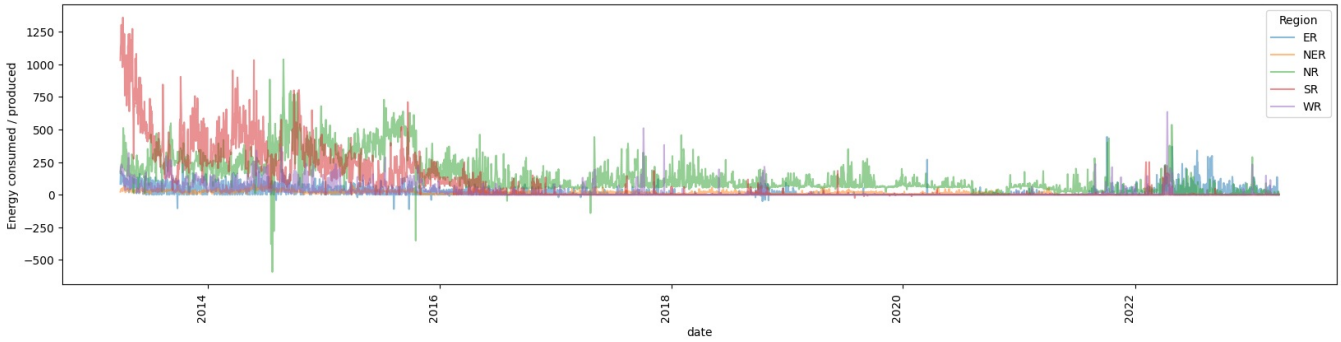
In [24]:
```
# Resample Monthly average power and plot
plt.subplots(4,1,figsize=(20,20))

plt.subplot(411)
df_train_sesional.resample('Y').mean()['Shortage during maximum Demand(MW)'].plot(ylabel='Shortage during maxim
```

```
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 maxim
plt.title("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 maxim
plt.title("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 maxim
plt.title("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]:

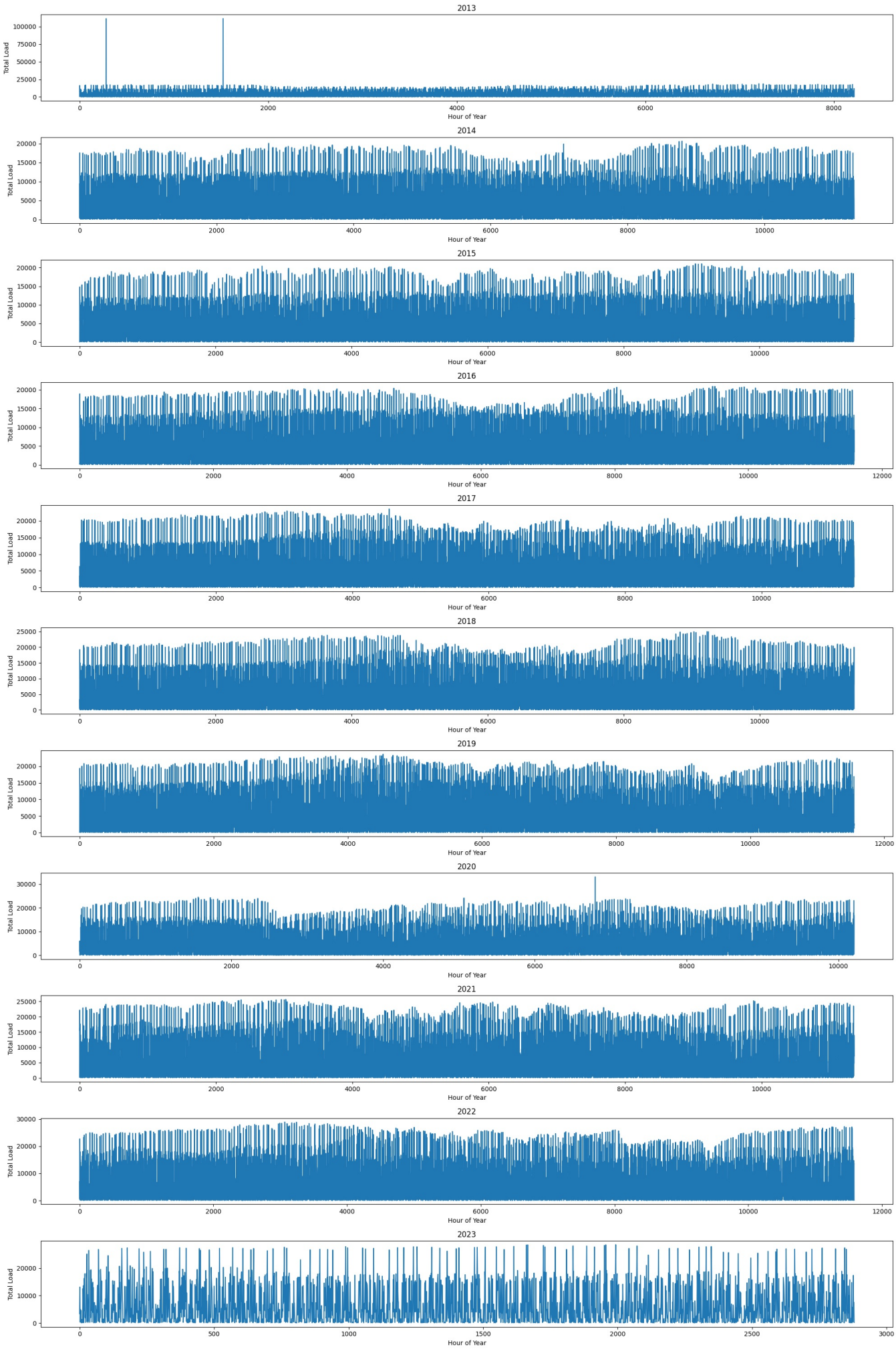
	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
0	NER	Mizoram	77	1.0	1.2	2015-01-01	Thursday	January	2015	1	1	Winter
2	WR	Goa	383	0.0	7.3	2015-01-01	Thursday	January	2015	1	1	Winter

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]:

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

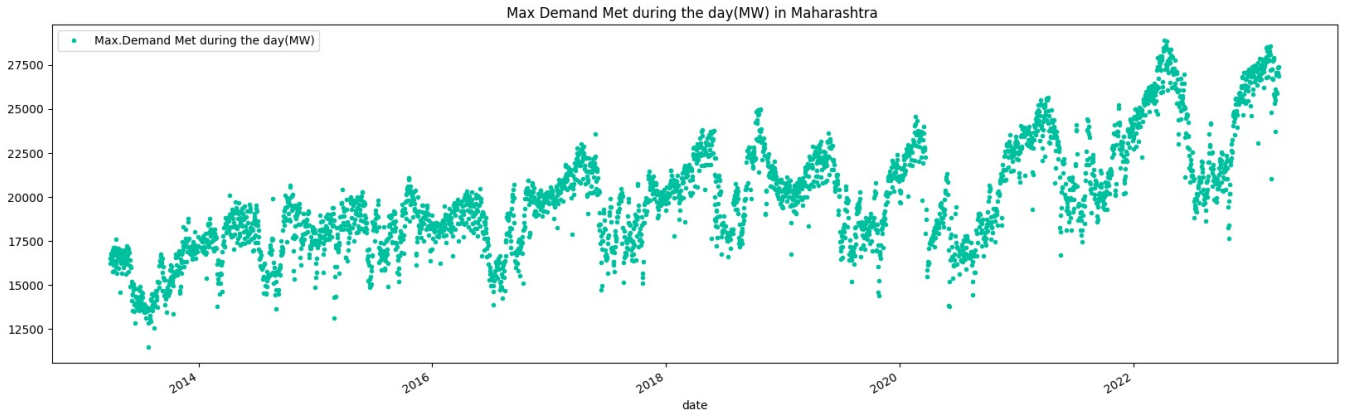
In [29]:
```
maha = df[df['States'] == 'Maharashtra']
maha = maha.sort_values(by='date',ascending=True)
display(maha.shape)
maha.head()
```

(3539, 12)

Out[29]:

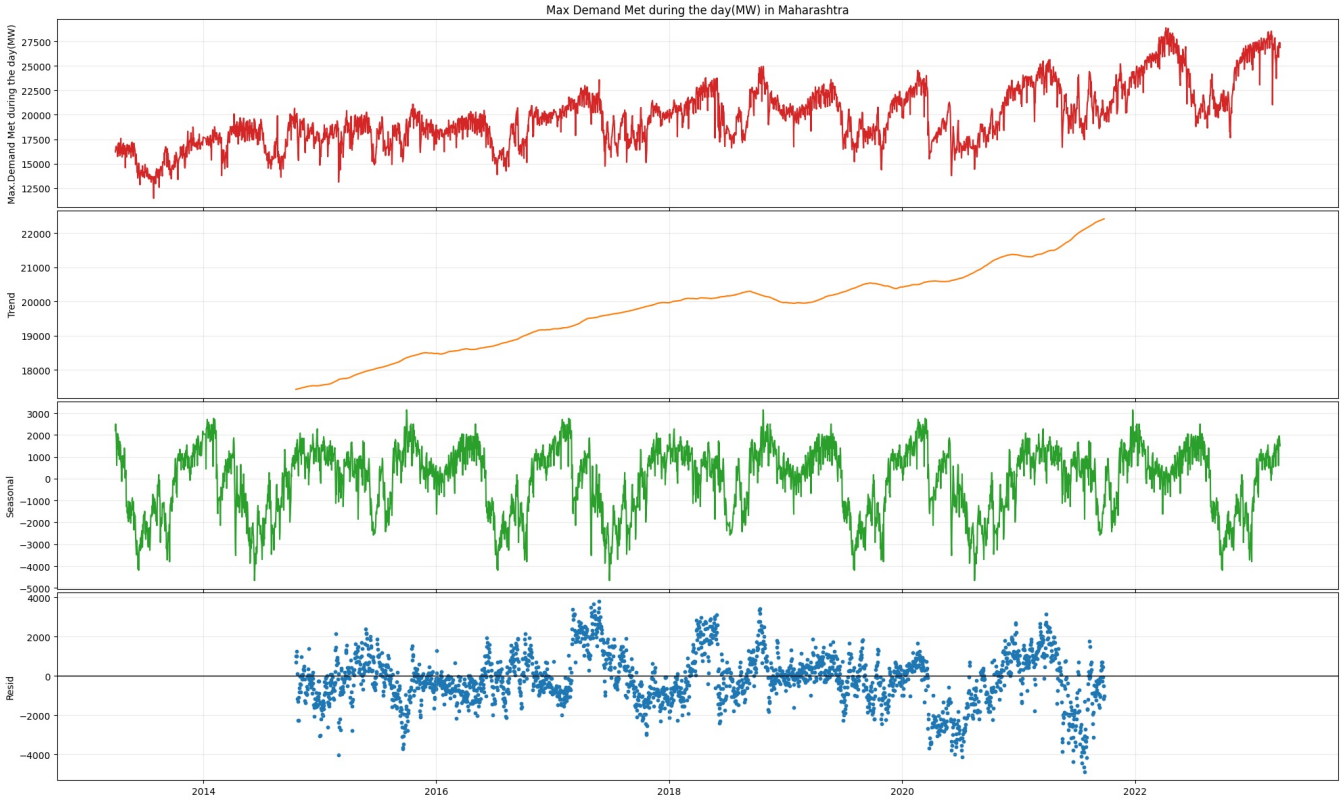
	Region	States	Max.Demand Met during the day(MW)	Shortage during maximum Demand(MW)	Energy Met (MU)	date	DayName	MonthName	Year	Quarter	Month	Season
124734	WR	Maharashtra	16196	712.0	337.8	2013-03-31	Sunday	March	2013	1	3	Spring
1140	WR	Maharashtra	16515	765.0	357.5	2013-04-01	Monday	April	2013	2	4	Spring
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 pallete 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.
    """
    # Decompostion
    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].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
```



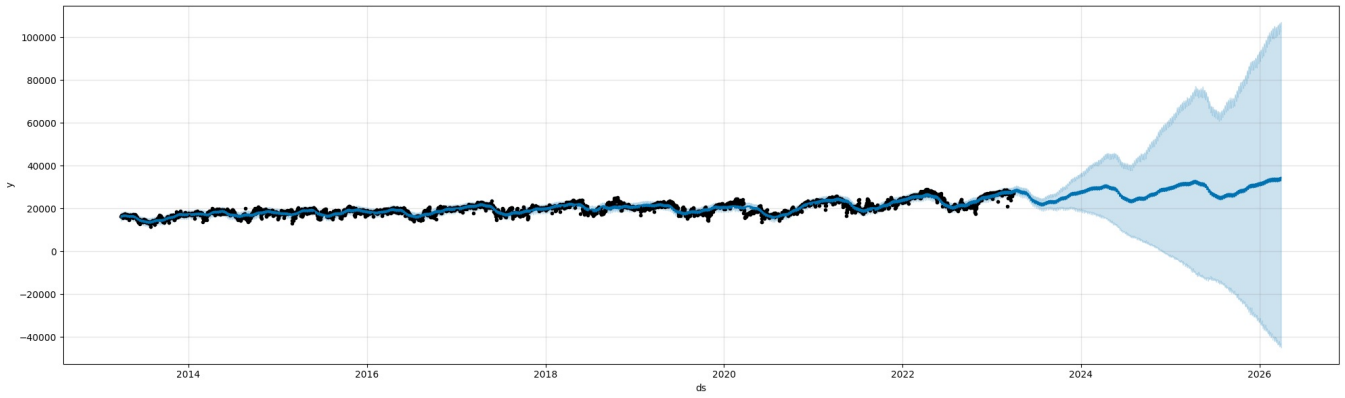
In [32]:
```
# Format data for prophet model using ds and y
df4 = maha.copy('deep')
df4 = df4[['date','Max.Demand Met during the day(MW)']]
df_final  = df4.reset_index().rename(columns={'date':'ds','Max.Demand Met during the day(MW)':'y'})
df_final = df_final.drop('index',axis=1)
df_final['ds'] = pd.to_datetime(df_final['ds'])
df_final['ds'] = df_final['ds'].dt.tz_localize(None)
df_final.head()
```

Out[32]:

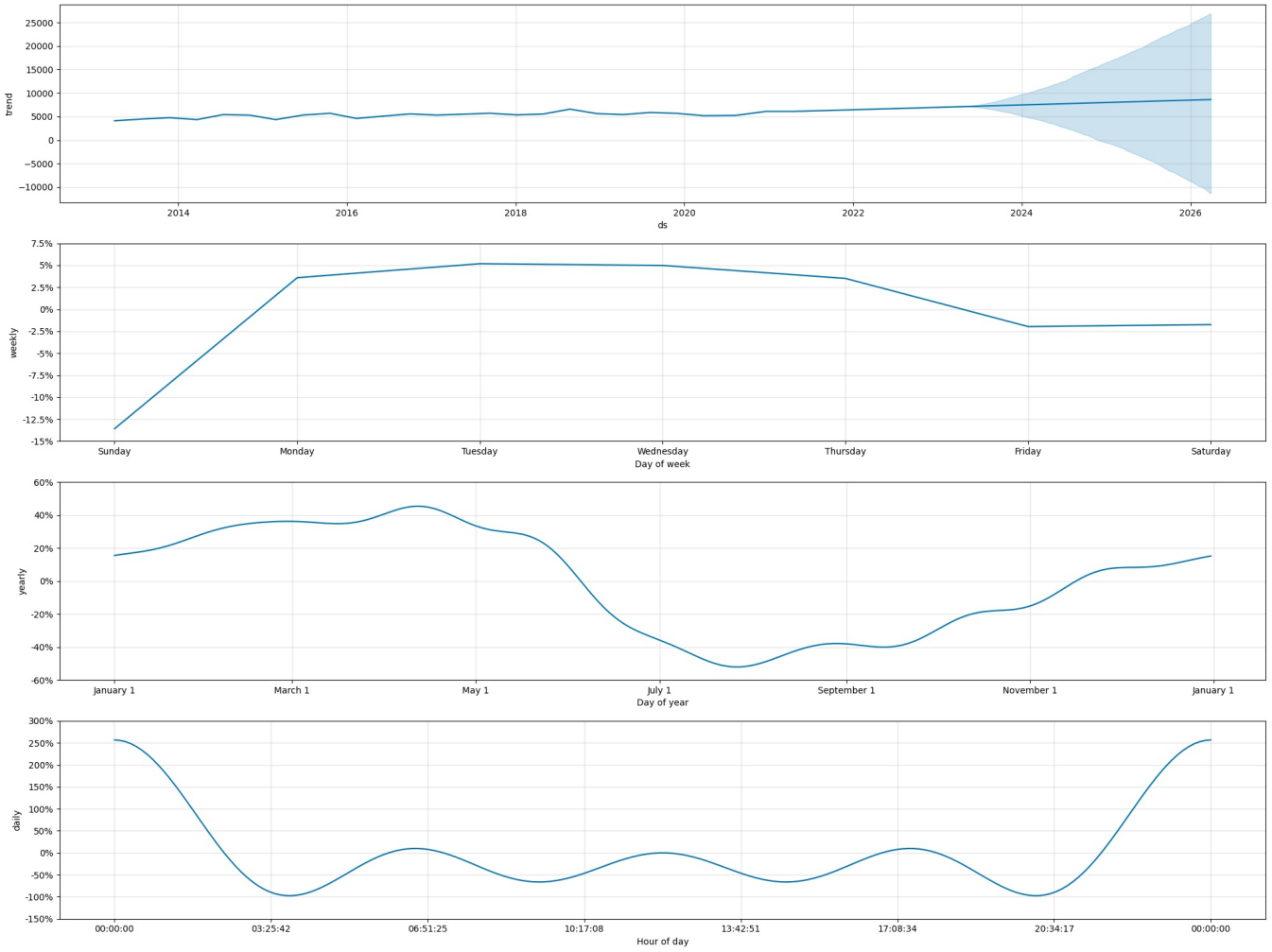
	ds	y
0	2013-03-31	16196
1	2013-04-01	16515
2	2013-04-02	16778
3	2013-04-03	16697
4	2013-04-04	16574

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()
```

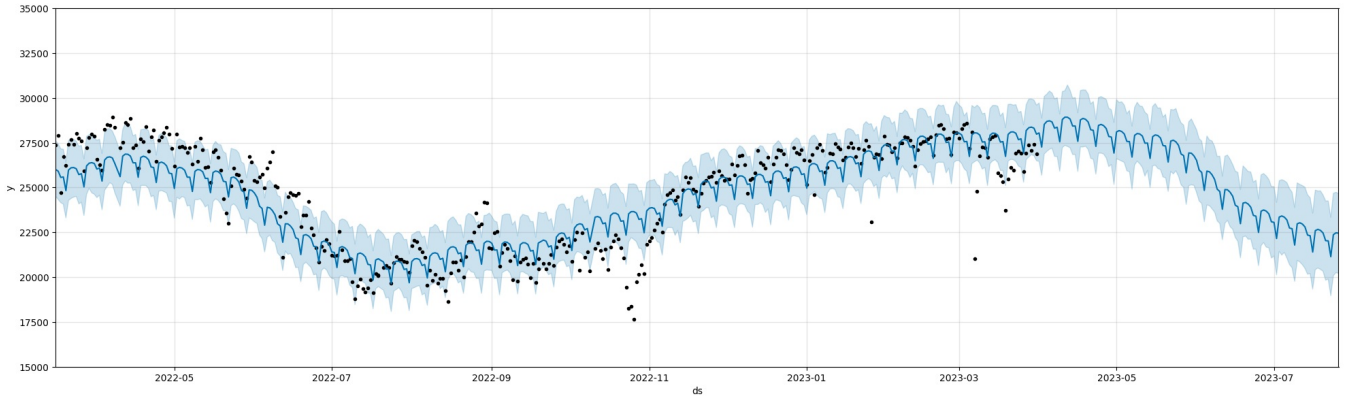
```
14:00:07 - cmdstanpy - INFO - Chain [1] start processing
14:00:11 - cmdstanpy - INFO - Chain [1] done processing
```



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

https://www.kaggle.com/code/pythonafroz/time-series-analysis-of-indian-power-generation

In [ ]: