

Solar Power Generation Prediction

Import Libraries

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

import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
pd.set_option('precision',3)
```

Import Data

```
In [3]: generation_data = pd.read_csv('../input/solar-power/Plant_2_Generation_Data.csv')
```

```
In [4]: weather_data = pd.read_csv('../input/solar-power/Plant_2_Weather_Sensor_Data.csv')
```

```
In [5]: generation_data.sample(5).style.set_properties(
    **{
        'background-color': 'OliveDrab',
        'color': 'white',
        'border-color': 'darkblack'
    })
```

```
Out[5]:
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
35624	2020-06-02 19:15:00	4136001	9kRcWv60rDACzjR	0.000	0.000	7073.000	2247850037.000
29650	2020-05-30 23:00:00	4136001	V94E5Ben1TlhndV	0.000	0.000	5288.000	1412205527.000
47823	2020-06-08 14:00:00	4136001	WcxssY2VbP4hApt	0.000	0.000	2143.000	181863802.000
29053	2020-05-30 16:15:00	4136001	PeE6FRyGXUgsRhN	588.813	576.993	3477.533	1348451247.533
53667	2020-06-11 08:30:00	4136001	LYwnQax7tkwH5Cb	227.713	223.280	228.067	1795079587.067

```
In [6]: weather_data.sample(5).style.set_properties(
    **{
        'background-color': 'pink',
        'color': 'Black',
        'border-color': 'darkblack'
    })
```

```
Out[6]:
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
2683	2020-06-12 00:00:00	4136001	iq8k7ZNt4Mwm3w0	23.096	21.819	0.000
3137	2020-06-16 17:30:00	4136001	iq8k7ZNt4Mwm3w0	28.318	29.484	0.068
2451	2020-06-09 14:00:00	4136001	iq8k7ZNt4Mwm3w0	33.631	47.321	0.542
56	2020-05-15 14:00:00	4136001	iq8k7ZNt4Mwm3w0	36.843	54.810	0.838
83	2020-05-15 20:45:00	4136001	iq8k7ZNt4Mwm3w0	31.047	29.371	0.000

Adjust datetime format

```
In [7]: generation_data['DATE_TIME'] = pd.to_datetime(generation_data['DATE_TIME'],format = '%Y-%m-%d %H:%M')
weather_data['DATE_TIME'] = pd.to_datetime(weather_data['DATE_TIME'],format = '%Y-%m-%d %H:%M:%S')
```

Merging generation data and weather sensor data

```
In [8]: df_solar = pd.merge(generation_data.drop(columns = ['PLANT_ID']), weather_data.drop(columns = ['PLANT_ID', 'SOURCE_KEY'])
df_solar.sample(5).style.background_gradient(cmap='cool')
```

Out[8]:

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IF
35583	2020-06-02 18:45:00	LYwnQax7tkwH5Cb	1.080	1.040	4787.000	1795055485.000	27.959	27.0	
10255	2020-05-19 21:00:00	Et9kgGMDI729KT4	0.000	0.000	2589.000	1724399.000	24.108	23.0	
960	2020-05-15 10:45:00	mqwcsP2rE7J0TFp	1245.057	1213.921	3132.929	593583157.929	33.192	50.2	
10608	2020-05-20 03:15:00	WcxssY2VbP4hApt	0.000	0.000	0.000	181734740.000	23.884	22.1	
17938	2020-05-24 11:00:00	Mx2yZCDsyf6DPfv	1159.587	1130.593	3203.333	2529233.333	33.418	55.3	

Adding separate time and date columns

In [9]:

```
# adding separate time and date columns
df_solar['DATE'] = pd.to_datetime(df_solar['DATE_TIME']).dt.date
df_solar['TIME'] = pd.to_datetime(df_solar['DATE_TIME']).dt.time
df_solar['DAY'] = pd.to_datetime(df_solar['DATE_TIME']).dt.day
df_solar['MONTH'] = pd.to_datetime(df_solar['DATE_TIME']).dt.month
df_solar['WEEK'] = pd.to_datetime(df_solar['DATE_TIME']).dt.week

# add hours and minutes for ml models
df_solar['HOURS'] = pd.to_datetime(df_solar['TIME'], format='%H:%M:%S').dt.hour
df_solar['MINUTES'] = pd.to_datetime(df_solar['TIME'], format='%H:%M:%S').dt.minute
df_solar['TOTAL MINUTES PASS'] = df_solar['MINUTES'] + df_solar['HOURS']*60

# add date as string column
df_solar['DATE_STRING'] = df_solar['DATE'].astype(str) # add column with date as string
df_solar['HOURS'] = df_solar['HOURS'].astype(str)
df_solar['TIME'] = df_solar['TIME'].astype(str)

df_solar.head(2)
```

Out[9]:

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IF
0	2020-05-15	4UPUqMRk7TRMgml	0.0	0.0	9425.0	2.429e+06	27.005	25.061	
1	2020-05-15	81aHJ1q11NBPMrL	0.0	0.0	0.0	1.215e+09	27.005	25.061	

In [10]:

```
df_solar.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 67698 entries, 0 to 67697
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DATE_TIME        67698 non-null   datetime64[ns]
 1   SOURCE_KEY       67698 non-null   object 
 2   DC_POWER          67698 non-null   float64
 3   AC_POWER          67698 non-null   float64
 4   DAILY_YIELD      67698 non-null   float64
 5   TOTAL_YIELD      67698 non-null   float64
 6   AMBIENT_TEMPERATURE 67698 non-null   float64
 7   MODULE_TEMPERATURE 67698 non-null   float64
 8   IRRADIATION      67698 non-null   float64
 9   DATE              67698 non-null   object 
 10  TIME              67698 non-null   object 
 11  DAY               67698 non-null   int64  
 12  MONTH             67698 non-null   int64  
 13  WEEK              67698 non-null   int64  
 14  HOURS             67698 non-null   object 
 15  MINUTES           67698 non-null   int64  
 16  TOTAL MINUTES PASS 67698 non-null   int64  
 17  DATE_STRING       67698 non-null   object 
```

dtypes: datetime64[ns](1), float64(7), int64(5), object(5)
memory usage: 9.8+ MB

In [11]:

```
df_solar.isnull().sum()
```

```
Out[11]: DATE_TIME      0
SOURCE_KEY      0
DC_POWER        0
AC_POWER        0
DAILY_YIELD     0
TOTAL_YIELD     0
AMBIENT_TEMPERATURE 0
MODULE_TEMPERATURE 0
IRRADIATION     0
DATE            0
TIME            0
DAY             0
MONTH           0
WEEK            0
HOURS           0
MINUTES          0
TOTAL_MINUTES_PASS 0
DATE_STRING     0
dtype: int64
```

There is no Missing Values in the dataset

```
In [12]: df_solar.describe().style.background_gradient(cmap='rainbow')
```

Out[12]:

	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DAY	MONTH	WEEK	HOURS	MINUTES	TOTAL_MINUTES_PASS	DATE_STRING	dtype
count	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000	67698.000
mean	246.702	241.278	3294.890	658944788.424	27.987	32.607	0.229	15.531	15.531	15.531	15.531	15.531	15.531	15.531	15.531
std	370.570	362.112	2919.448	729667771.073	4.021	11.226	0.309	8.528	8.528	8.528	8.528	8.528	8.528	8.528	8.528
min	0.000	0.000	0.000	0.000	20.942	20.265	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
25%	0.000	0.000	272.750	19964944.867	24.570	23.686	0.000	9.000	9.000	9.000	9.000	9.000	9.000	9.000	9.000
50%	0.000	0.000	2911.000	282627587.000	26.910	27.434	0.019	16.000	16.000	16.000	16.000	16.000	16.000	16.000	16.000
75%	446.592	438.215	5534.000	1348495113.000	30.913	40.019	0.431	22.000	22.000	22.000	22.000	22.000	22.000	22.000	22.000
max	1420.933	1385.420	9873.000	2247916295.000	39.182	66.636	1.099	31.000	31.000	31.000	31.000	31.000	31.000	31.000	31.000

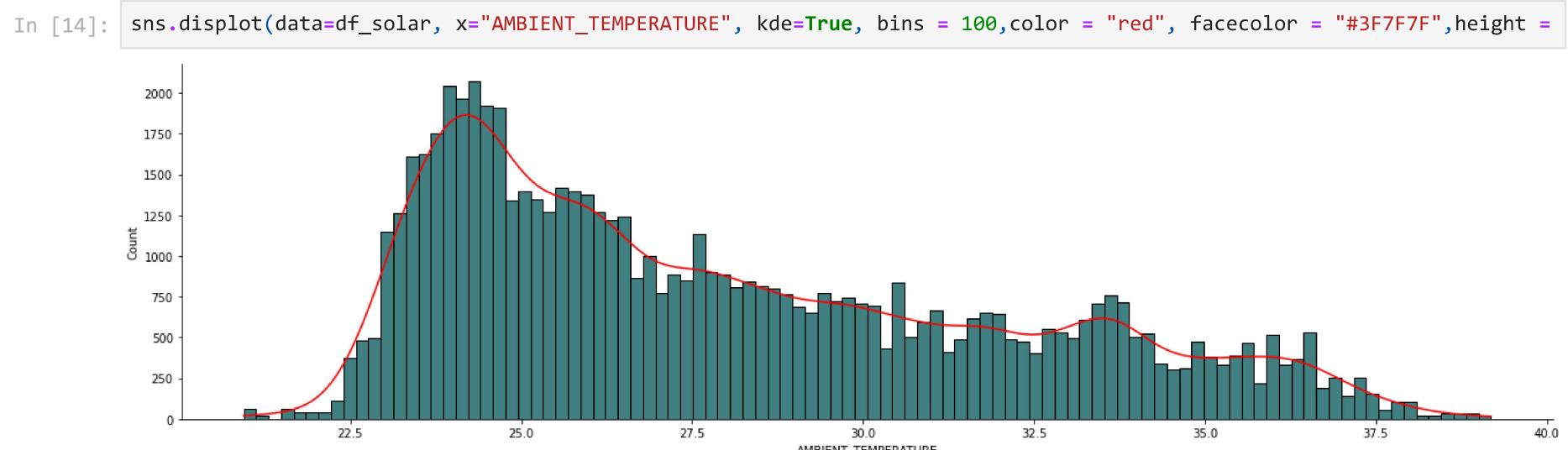
Converting 'SOURCE_KEY' from categorical form to numerical form

```
In [13]: from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df_solar['SOURCE_KEY_NUMBER'] = encoder.fit_transform(df_solar['SOURCE_KEY'])
df_solar.head()
```

Out[13]:

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DAY	MONTH	WEEK	HOURS	MINUTES	TOTAL_MINUTES_PASS	DATE_STRING	dtype
0	2020-05-15	4UPUqMRk7TRMgml	0.0	0.0	9425.000	2.429e+06	27.005	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061
1	2020-05-15	81aHJ1q11NBPMrl	0.0	0.0	0.000	1.215e+09	27.005	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061
2	2020-05-15	9kRcWv60rDACzjR	0.0	0.0	3075.333	2.248e+09	27.005	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061
3	2020-05-15	Et9kgGMDI729KT4	0.0	0.0	269.933	1.704e+06	27.005	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061
4	2020-05-15	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000	1.994e+07	27.005	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061	25.061

Data Visualization



```
In [15]: df_solar['DATE'].nunique()
```

Out[15]: 34

The data of solar power generation is of 34 days

Faults & Abnormalities detection in solar power plant generation

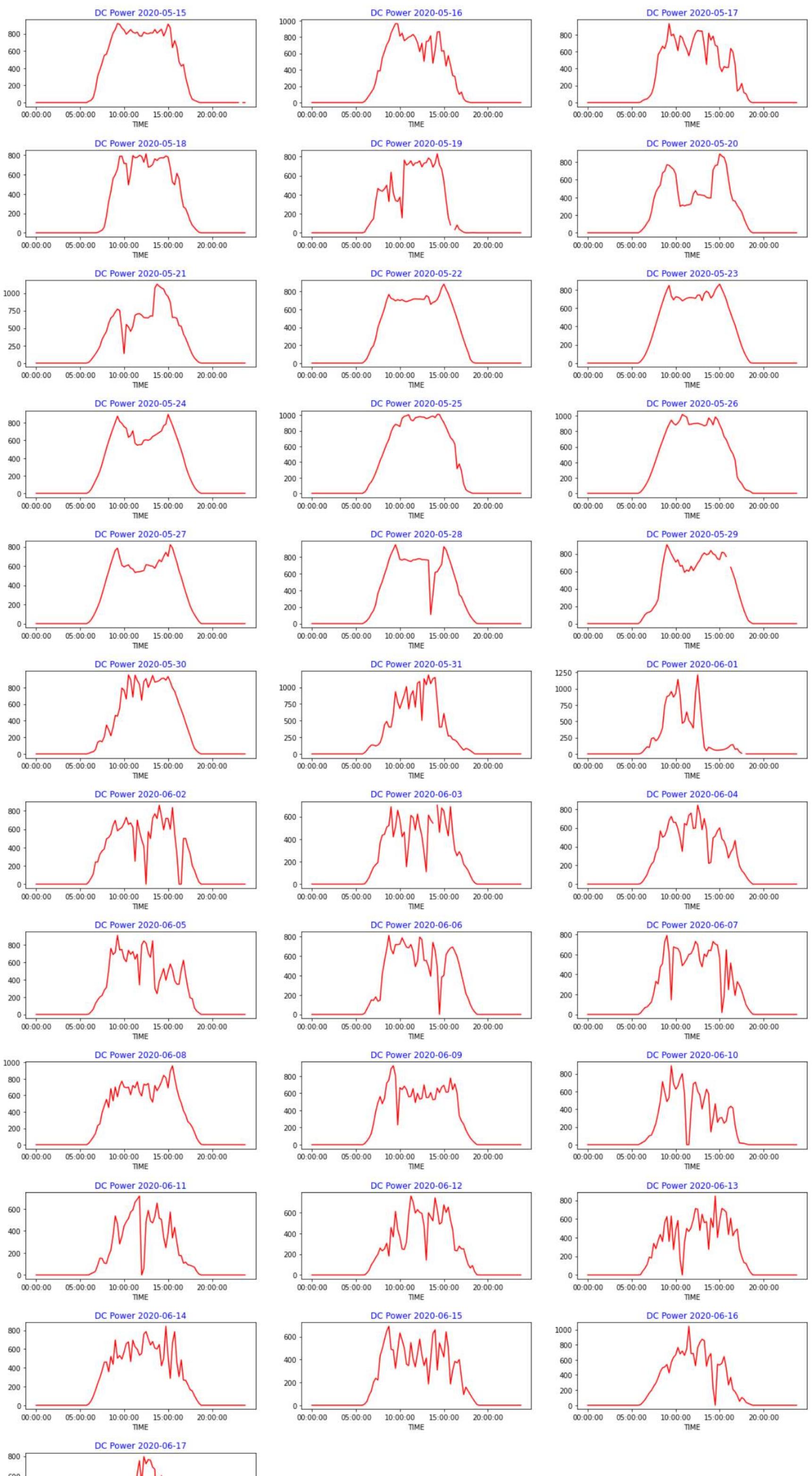
Multiple Plotting of DC_POWER generation on per day basis.

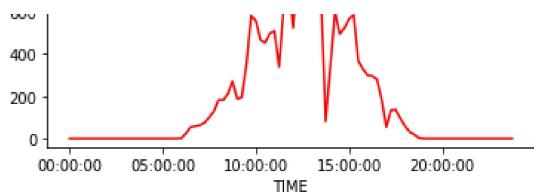
```
In [16]: solar_dc = df_solar.pivot_table(values='DC_POWER', index='TIME', columns='DATE')

def Daywise_plot(data= None, row = None, col = None, title='DC Power'):
    cols = data.columns # take all column
    gp = plt.figure(figsize=(20,40))

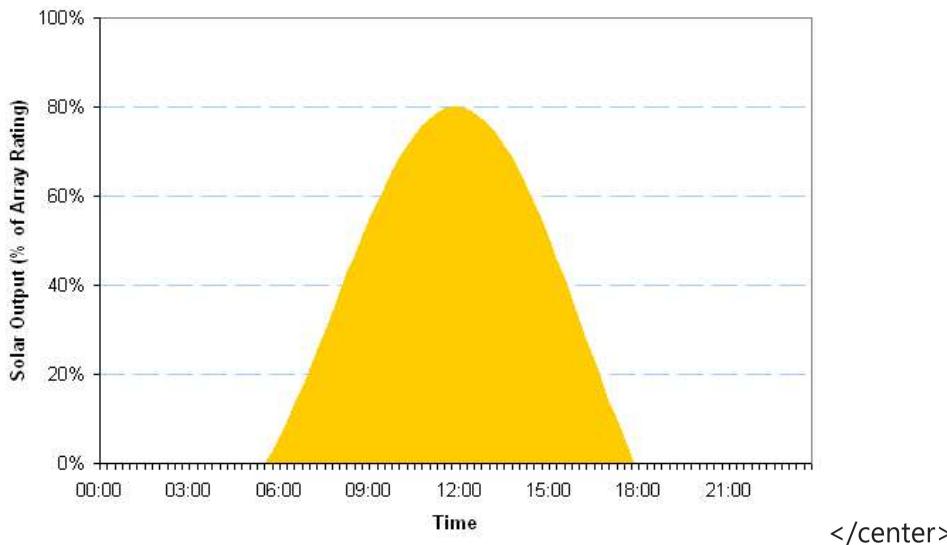
    gp.subplots_adjust(wspace=0.2, hspace=0.5)
    for i in range(1, len(cols)+1):
        ax = gp.add_subplot(row,col, i)
        data[cols[i-1]].plot(ax=ax, color='red')
        ax.set_title('{}_{}'.format(title, cols[i-1]),color='blue')

Daywise_plot(data=solar_dc, row=12, col=3)
```





Ideal Graph of Solar Power Generation



The reason we get this shape is based on the angle of the sun to your panels. In the early morning, when the sun is still rising, it's essentially at its lowest point in the sky. In order for the sunlight to reach your panels at this time of the day, it has to cut through the most amount of atmosphere. This is important to note, because energy from the sun is absorbed when it travels through the atmosphere. So, this means that the more atmosphere it travels through, the more energy is absorbed and the less that is available for your panels to convert into electricity. At midday, when the sun is directly overhead, sunlight doesn't need to travel through as much atmosphere. Therefore, your panels will absorb more sunlight during this time.

- In winter, the same concept applies. However, on gloomier winter days there may be less sunlight for your panels to absorb.
- This can simply be explained by the sun being positioned lower in the sky than it is during summer.

Abnormalities in DC_POWER Generation

From the per day DC_POWER generation graph we can find that, most of the days there is a some fluctuation in the power generation.

Less Fluctuation in DC_POWER generation is observed in these days.

1. 2020-05-15
2. 2020-05-18
3. 2020-05-22
4. 2020-05-23
5. 2020-05-24
6. 2020-05-25
7. 2020-05-26

High Fluctuation in DC_POWER generation is observed in these days.

1. 2020-05-19
2. 2020-05-28
3. 2020-05-29
4. 2020-06-02
5. 2020-06-03
6. 2020-06-04
7. 2020-06-13
8. 2020-06-14
9. 2020-06-17

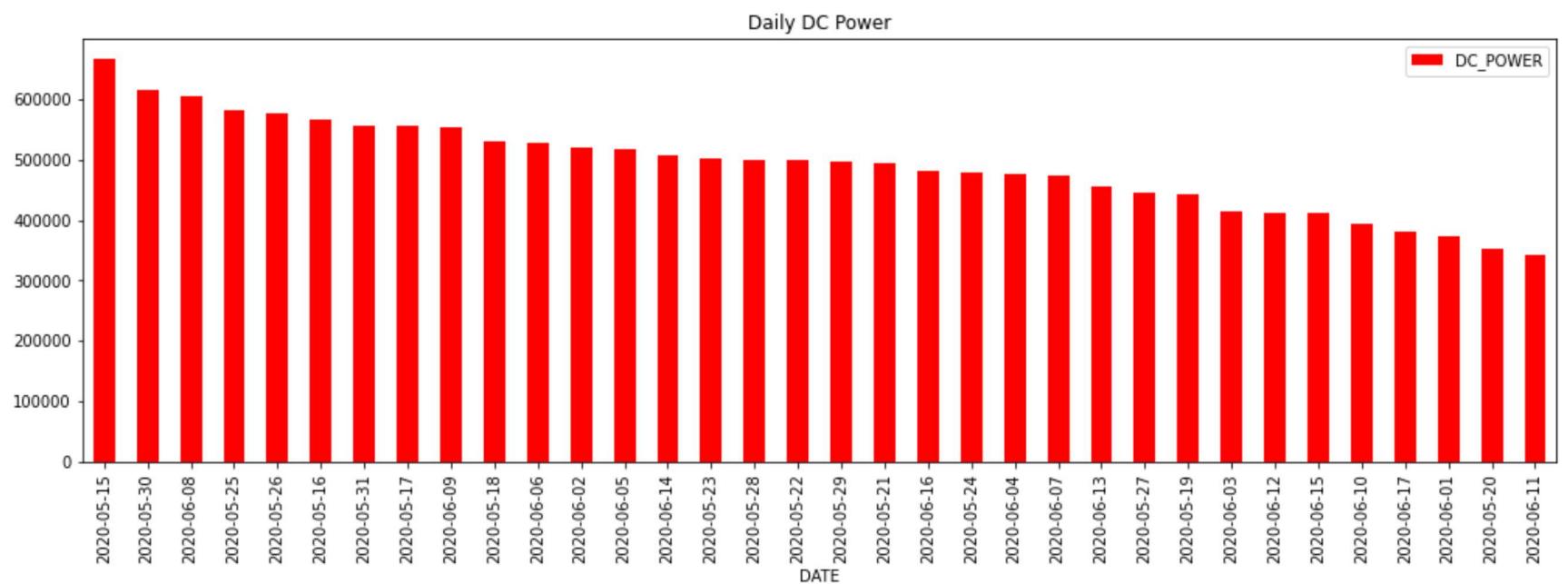
Very High Fluctuation & Reduction in DC_POWER generation is observed in these days.

1. 2020-06-03
2. 2020-06-11
3. 2020-06-12
4. 2020-06-15

Note: Reason for very high Fluctuation & Reduction in DC_POWER generation is due to fault in the system or may be fluctuation in weather or due to clouds etc. which need to be analyse further

```
In [17]: daily_dc = df_solar.groupby('DATE')['DC_POWER'].agg('sum')

ax = daily_dc.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='red')
plt.title('Daily DC Power')
plt.show()
```



From the per day DC_POWER generation graph we can find the average power generation per day.

Highest average DC_POWER Generation is on: 2020-05-15

Lowest average DC_POWER Generation is on : 2020-06-11

NOTE: This Large variation in the DC_POWER generation is due to the fault in the system or due to weather change, which needs to study further. But from this bar plot we find the day on which there is highest DC_POWER is generated and the day with the lowest DC_POWER generated.

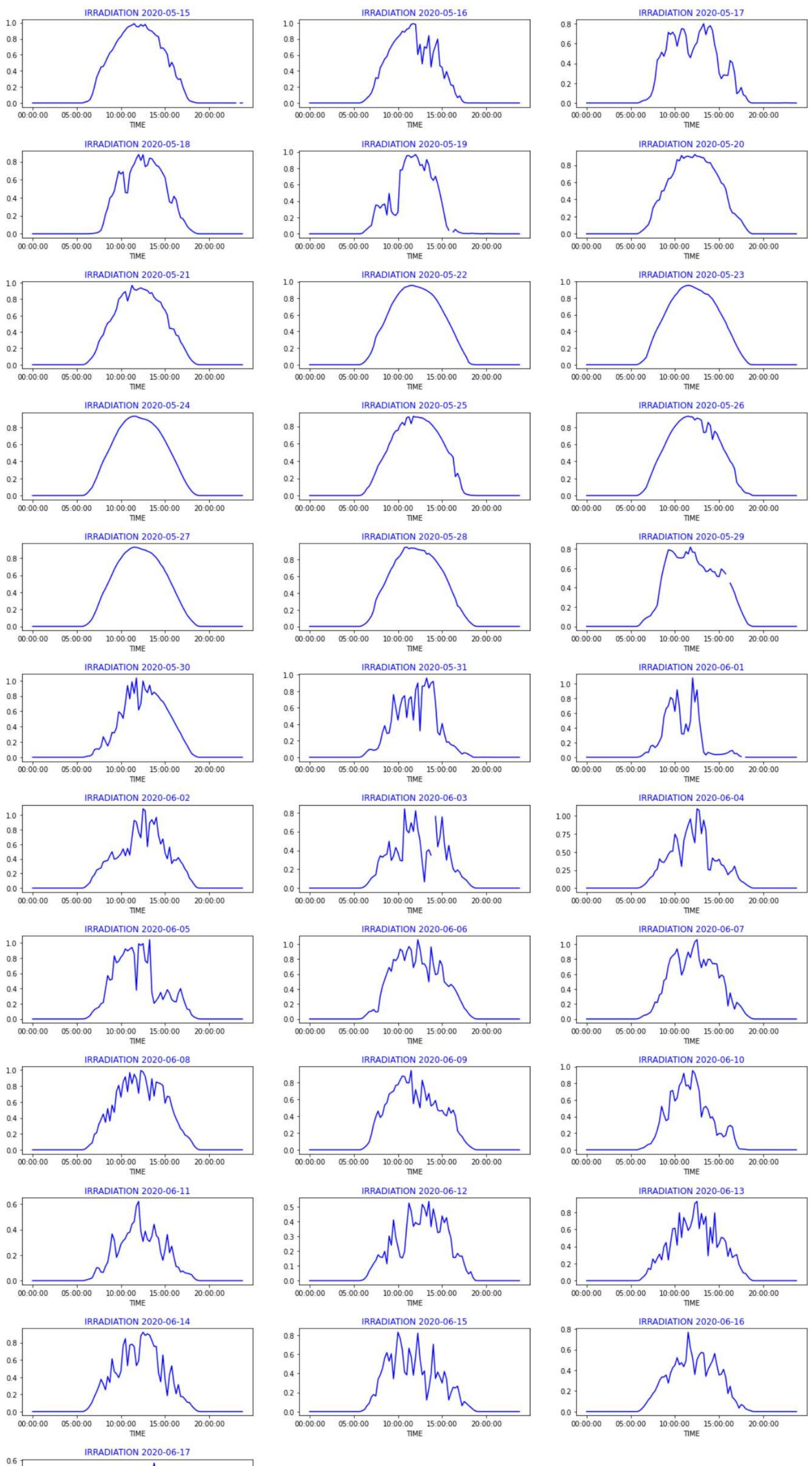
Multiple Plotting of IRRADIATION generation on per day basis.

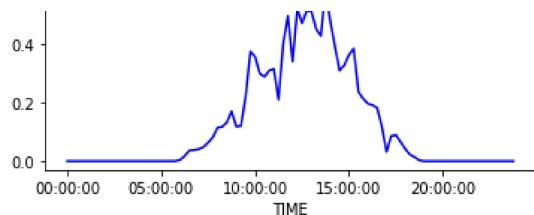
```
In [18]: solar_irradiation = df_solar.pivot_table(values='IRRADIATION', index='TIME', columns='DATE')

def Daywise_plot(data= None, row = None, col = None, title='IRRADIATION'):
    cols = data.columns # take all column
    gp = plt.figure(figsize=(20,40))

    gp.subplots_adjust(wspace=0.2, hspace=0.5)
    for i in range(1, len(cols)+1):
        ax = gp.add_subplot(row,col, i)
        data[cols[i-1]].plot(ax=ax, color='blue')
        ax.set_title('{}_{}'.format(title, cols[i-1]),color='blue')

Daywise_plot(data=solar_irradiation, row=12, col=3)
```



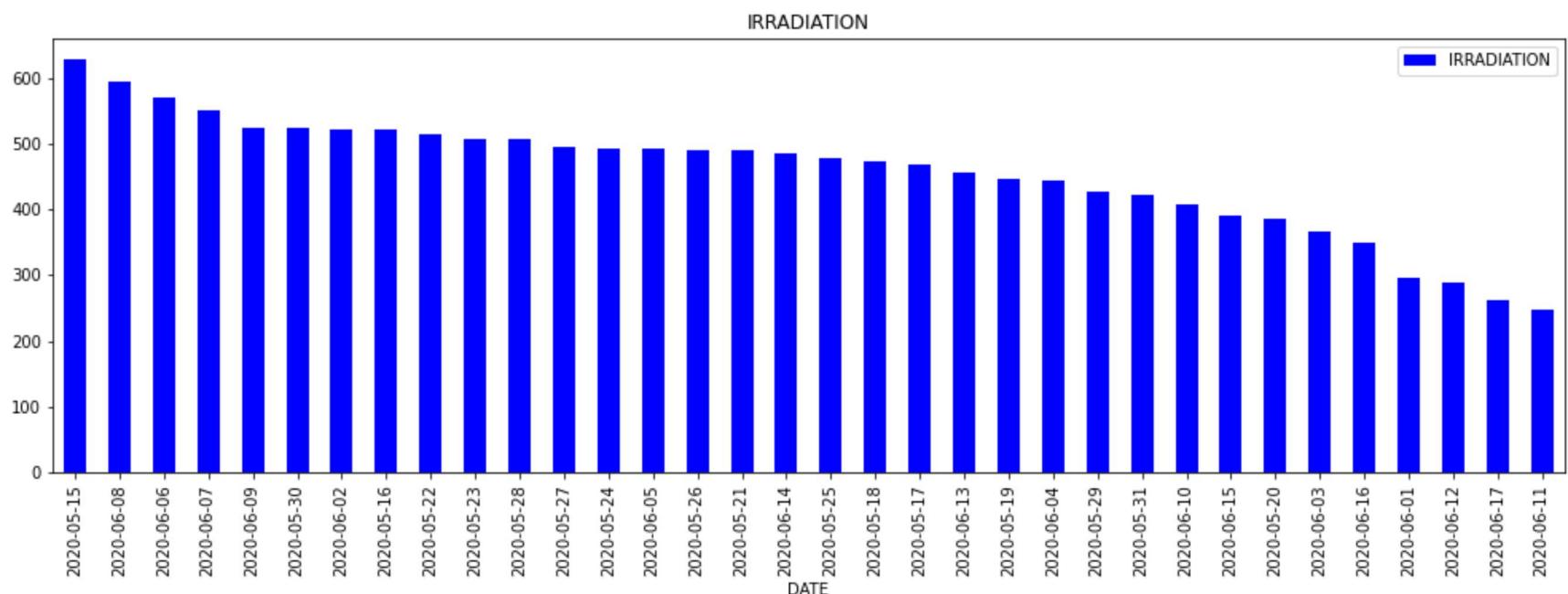


IRRADIATION graph pattern is looking very similar to the corresponding DC_POWER generation on per day basis.

- In solar power plant DC_POWER or Output power is mostly depends on IRRADIATION .Or it is not wrong to say that it's directly proportional.

```
In [19]: daily_irradiation = df_solar.groupby('DATE')[['IRRADIATION']].agg('sum')

daily_irradiation.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='blue')
plt.title('IRRADIATION')
plt.show()
```

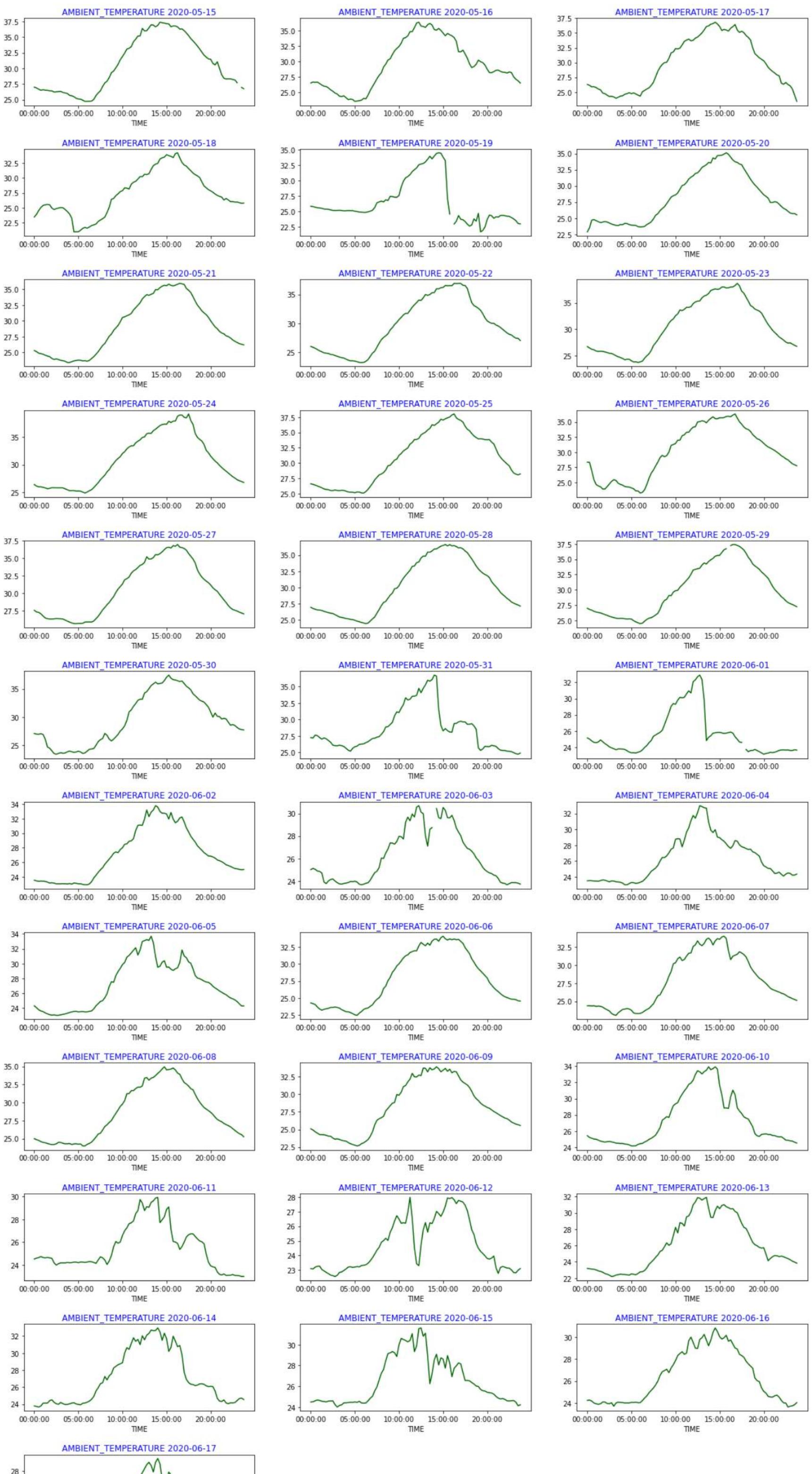


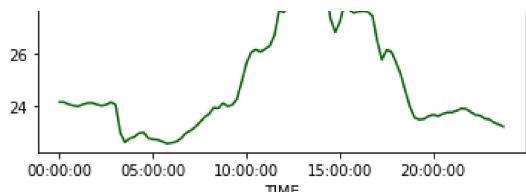
```
In [20]: solar_ambient_temp = df_solar.pivot_table(values='AMBIENT_TEMPERATURE', index='TIME', columns='DATE')

def Daywise_plot(data= None, row = None, col = None, title='AMBIENT_TEMPERATURE'):
    cols = data.columns # take all column
    gp = plt.figure(figsize=(20,40))

    gp.subplots_adjust(wspace=0.2, hspace=0.5)
    for i in range(1, len(cols)+1):
        ax = gp.add_subplot(row,col, i)
        data[cols[i-1]].plot(ax=ax, color='darkgreen')
        ax.set_title('{0} {1}'.format(title, cols[i-1]),color='blue')

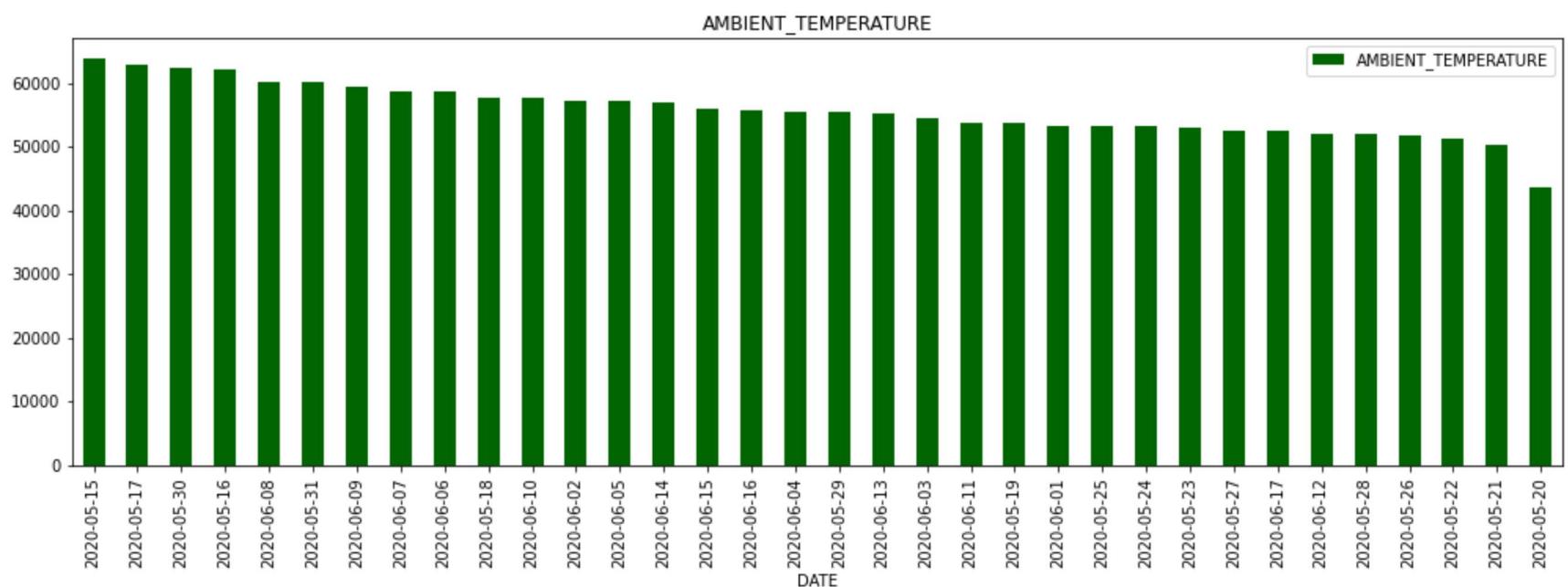
Daywise_plot(data=solar_ambient_temp, row=12, col=3)
```





```
In [21]: daily_ambient_temp = df_solar.groupby('DATE')[['AMBIENT_TEMPERATURE']].agg('sum')

daily_ambient_temp.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='darkgreen')
plt.title('AMBIENT_TEMPERATURE')
plt.show()
```



Best and Worst Power generation comparision

Major Environmental Factors affecting the solar power generation are:

1. The thickness of clouds is also a factor in how much sunlight your solar panels can soak up. We may see thicker clouds in winter too and this is something else to look out for. It's hard for sunlight to travel through thick clouds, which will affect your solar power system's output.
2. While we've looked at the sun's positioning and how it can affect output, there's another factor to consider when your system may not be performing at its maximum... even at midday.
3. Solar panel temperature is the number one reason behind your solar power system not achieving peak performance
4. Solar power generation is directly depends on Irradiation comming from the sun.

Highest average DC_POWER is generated on "2020-05-15"

```
In [22]: plt.figure(figsize=(16,16))

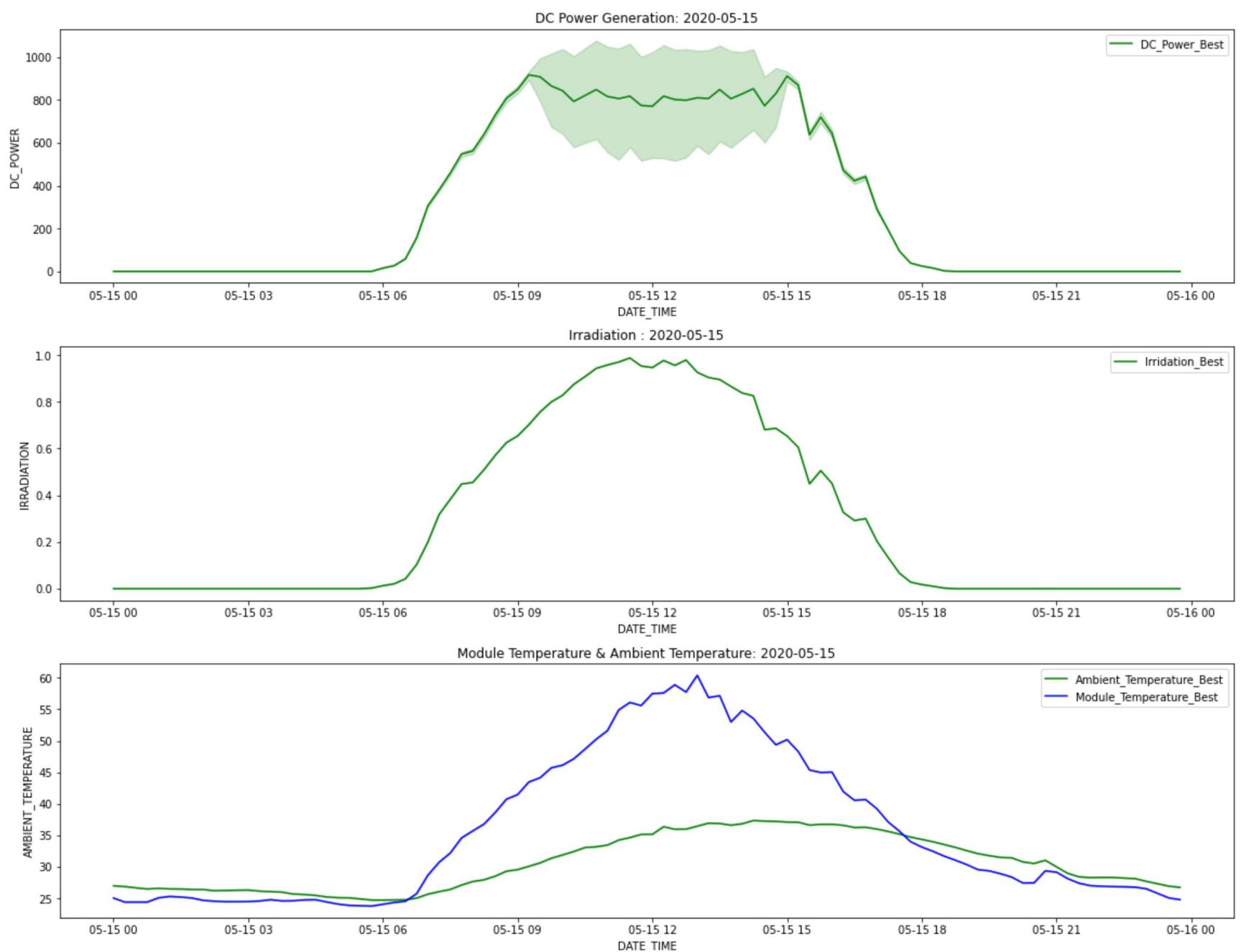
date=["2020-05-15"]

plt.subplot(411)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].DC_POWER)
plt.title("DC Power Generation: {}" .format(date[0]))

plt.subplot(412)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].IRRADIATION)
plt.title("Irradiation : {}" .format(date[0]))

plt.subplot(413)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].AMBIENT_TEMPERATURE)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].MODULE_TEMPERATURE)
plt.title("Module Temperature & Ambient Temperature: {}" .format(date[0]));

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



NOTE: Both DC_POWER graph and IRRADIATION graph is almost looking like an ideal graph which is explained earlier. Weather is also looking good, and there is no cloud is in the sky because there is very less variation in IRRADIATION and temperature of the solar panel and ambient temperature.

Lowest average DC_POWER is generated on "2020-06-11"

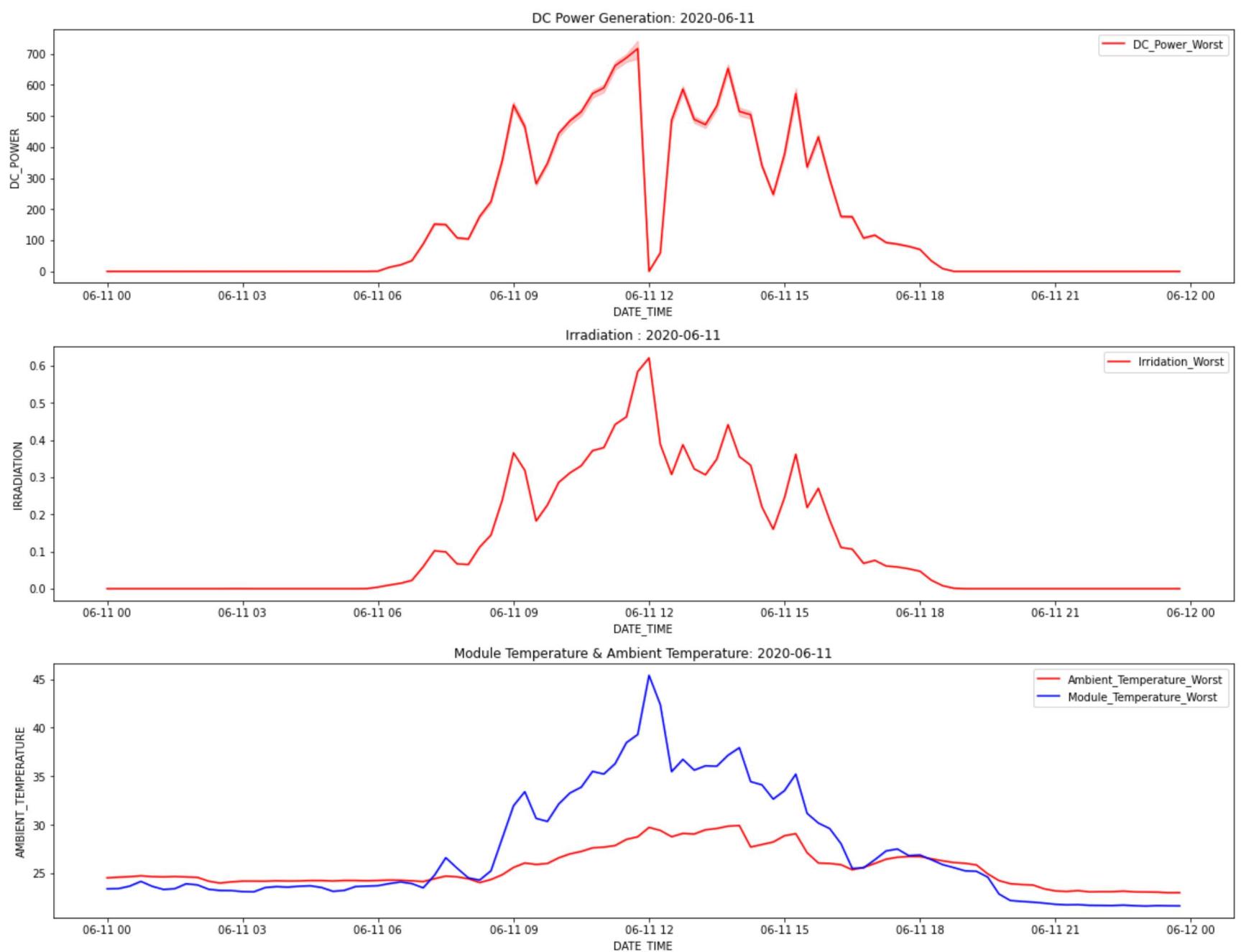
```
In [23]: date=["2020-06-11"]
plt.figure(figsize=(16,16))

plt.subplot(411)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].DC_POWER)
plt.title("DC Power Generation: {}" .format(date[0]))

plt.subplot(412)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].IRRADIATION)
plt.title("Irradiation : {}" .format(date[0]))

plt.subplot(413)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].AMBIENT_TEMPERATURE)
sns.lineplot(df_solar[df_solar["DATE_STRING"].isin(date)].DATE_TIME, df_solar[df_solar["DATE_STRING"].isin(date)].MODULE_TEMPERATURE)
plt.title("Module Temperature & Ambient Temperature: {}" .format(date[0]));

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



NOTE: There are very large fluctuations in both DC_POWER graph and IRRADIATION graph

Possible Reasons for these large fluctuation in the DC_POWER, IRRADIATION, Ambient temperature, Module temperature:

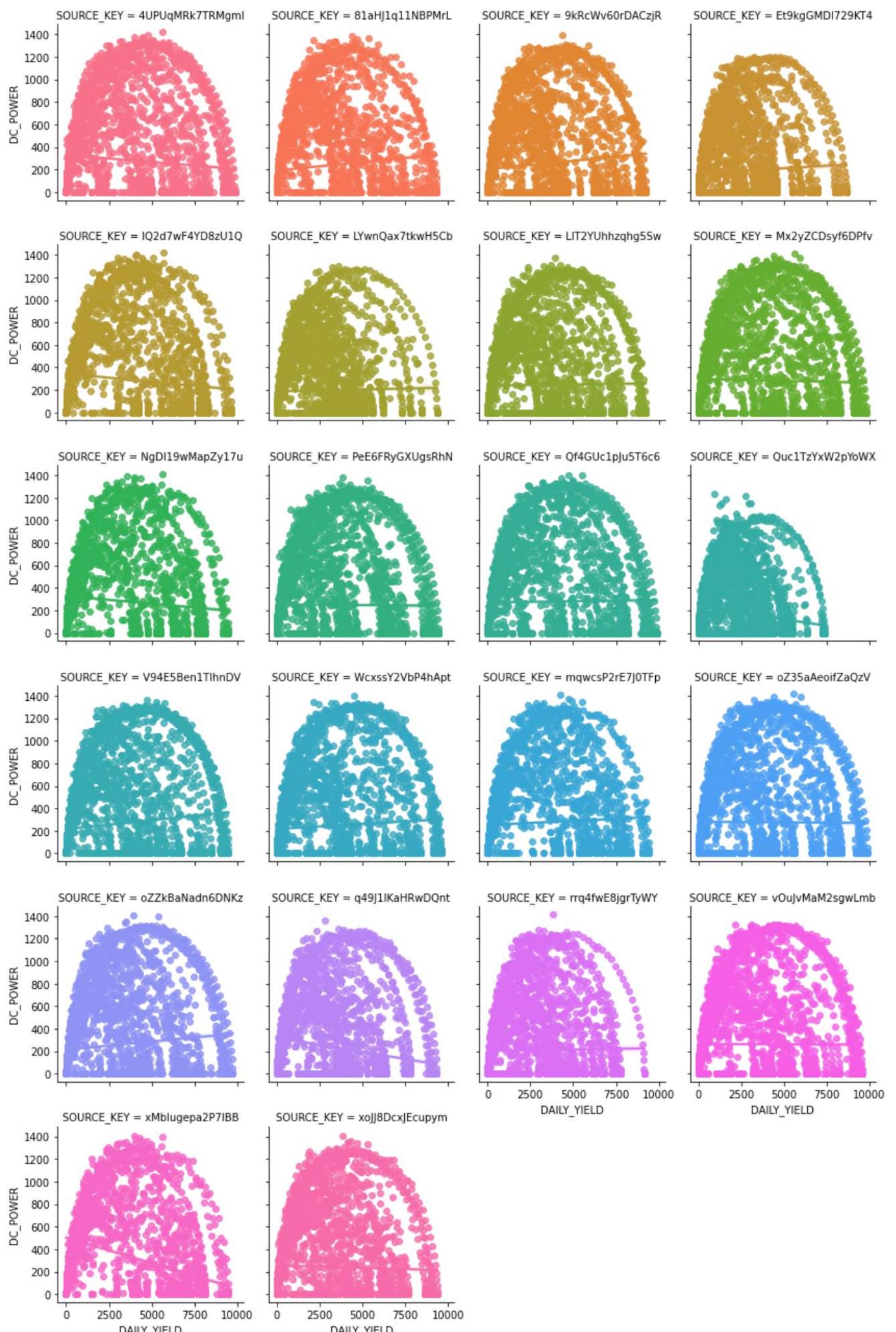
At about 12 O'clock there is a sharp decline in the DC_POWER generation from 700 to almost 20 KWatt.

And at the same time IRRADIATION fall from 0.6 to 0.3 almost half.

Ambient temperature and Module temperature also fall drastically. Module temperature from 45 C to 35 C & Ambient temperature is also reduced.

The possible reason for this reduction is due to may be heavy rain and heavily clouded sky and bad weather. There is almost very less possibility of any fault in the system

```
In [24]: sns.lmplot(y="DC_POWER", x="DAILY_YIELD", hue="SOURCE_KEY", col="SOURCE_KEY", height=3, col_wrap=4, data=df_solar, fit_reg=True)
```



Solar Power Plant Inverter Efficiency Calculation

```
In [25]: solar_dc_power = df_solar[df_solar['DC_POWER'] > 0]['DC_POWER'].values
solar_ac_power = df_solar[df_solar['AC_POWER'] > 0]['AC_POWER'].values
```

```
In [26]: solar_plant_eff = (np.max(solar_ac_power)/np.max(solar_dc_power ))*100
print(f"Power ratio AC/DC (Efficiency) of Solar Power Plant: {solar_plant_eff:0.3f} %")
```

Power ratio AC/DC (Efficiency) of Solar Power Plant: 97.501 %

```
In [27]: AC_list=[]
for i in df_solar['AC_POWER']:
    if i>0:
        AC_list.append(i)
```

```
AC_list
#AC_List.sort()
#AC_List.reverse()
len(AC_list)
```

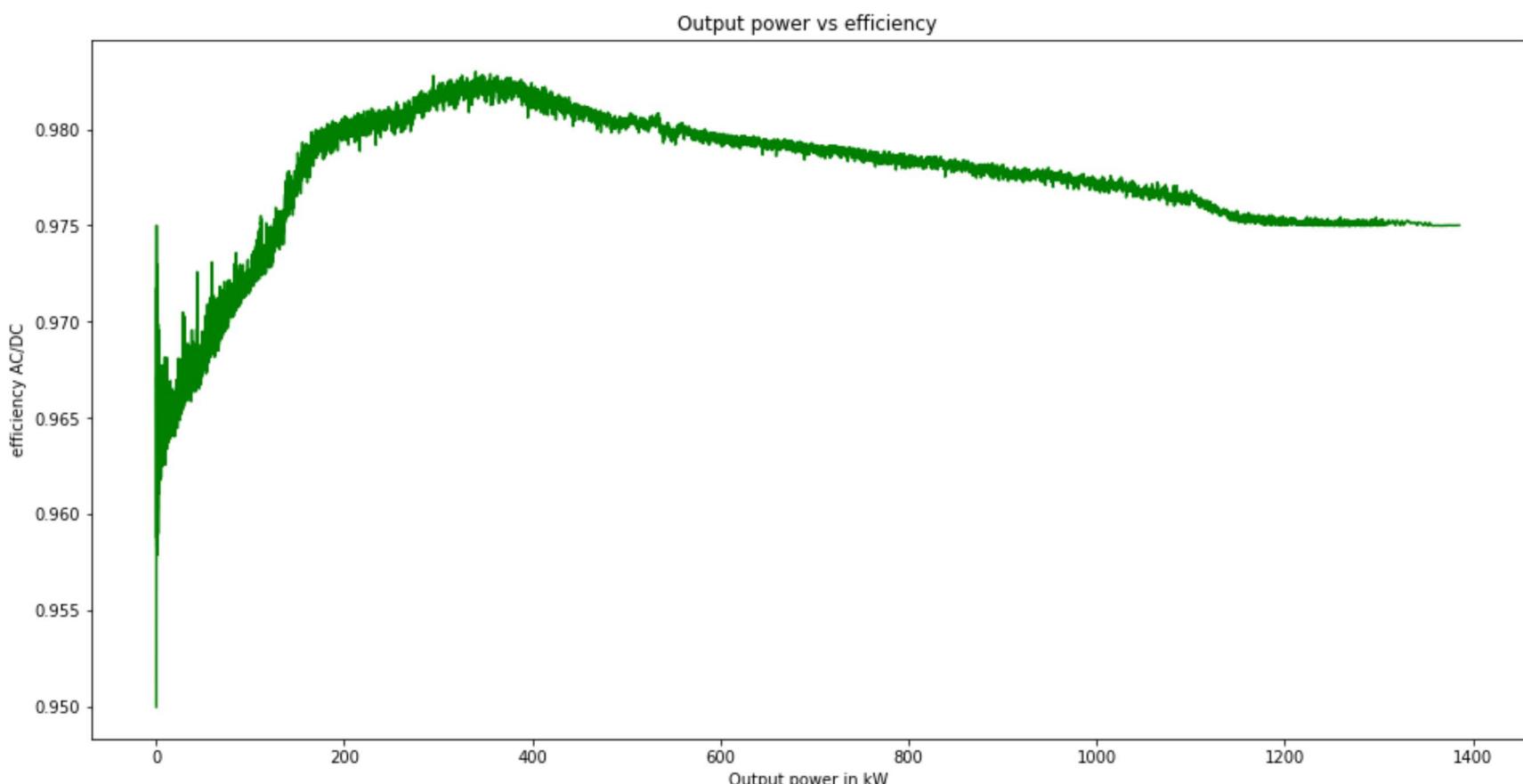
Out[27]: 32036

```
In [28]: #Here we take all nonzero DC values and plot them on histogram
DC_list=[]
for i in df_solar['DC_POWER']:
    if i>0:
        DC_list.append(i)
DC_list
DC_list.sort()
DC_list.reverse()
len(DC_list)
```

Out[28]: 32036

```
In [29]: plt.figure(figsize=(16,8))
AC_list.sort()
DC_list.sort()
#print(DC_list)
#DC_list.sort
#res = [i / 10 for i in AC_list]
eff = [i/j for i,j in zip(AC_list,DC_list)]

plt.plot(AC_list,eff,color='green')
plt.xlabel('Output power in kW')
plt.ylabel('efficiency AC/DC')
plt.title('Output power vs efficiency');
```

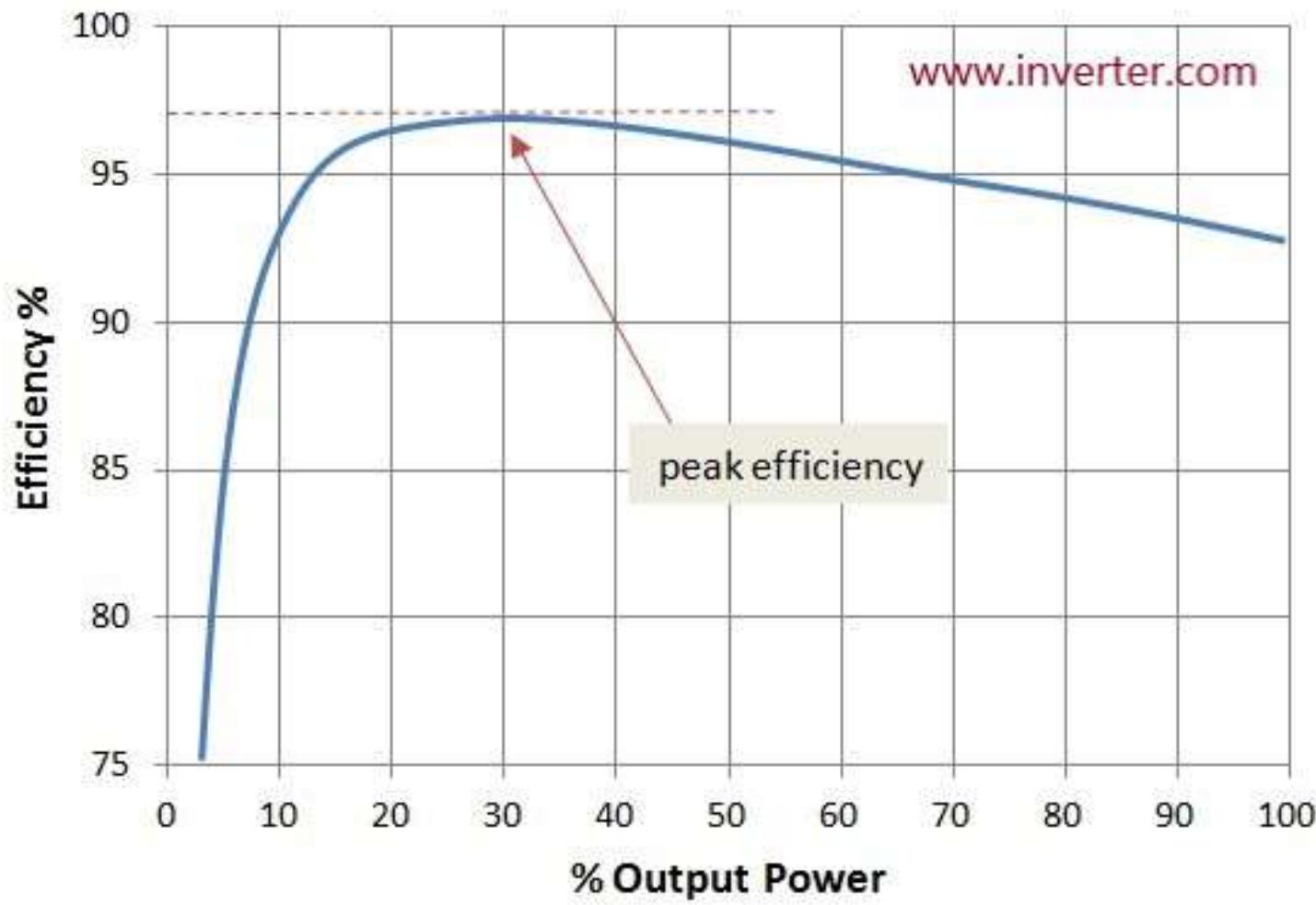


What does inverter efficiency mean?

- In fact, we shall discuss here the general power inverter efficiency whether it's solar inverter or pure sine wave inverter or even modified sine wave inverter.
- The inverter efficiency refers to how much dc power will be converted to ac power, as some of power will be lost during this transition in two forms:

Heat loss.

- Stand-by power which consumed just to keep the inverter in power mode. Also, we can refer to it as inverter power consumption at no load condition.
- Hence, inverter efficiency = pac/pdc where pac refers to ac output power in watt and pdc refers to dc input power in watts.



</center>

For the two basic inverters types in the market, the typical efficiency of high-quality pure sine wave inverter varied from 90% to 95% and for low quality modified sine wave inverter, it varied from 75% to 85%.

This power inverter efficiency value depends on inverter load power capacity variation, as the efficiency increases and may reach to its max value at higher load power capacity in compare to lower loading power capacity, and in condition that not going above inverter output power capacity limit. Generally, below 15% inverter loading, the efficiency will be quite low. Consequently, good matching between inverter capacity and its load capacity will enable us harvest larger efficiency, which means larger inverter ac output power for the same dc input power.

Solar Power Prediction

```
In [30]: df2 = df_solar.copy()
X = df2[['DAILY_YIELD', 'TOTAL_YIELD', 'AMBIENT_TEMPERATURE', 'MODULE_TEMPERATURE', 'IRRADIATION', 'DC_POWER']]
y = df2['AC_POWER']
```

```
In [31]: X.head()
```

```
Out[31]: DAILY_YIELD  TOTAL_YIELD  AMBIENT_TEMPERATURE  MODULE_TEMPERATURE  IRRADIATION  DC_POWER
0    9425.000    2.429e+06          27.005          25.061        0.0        0.0
1      0.000    1.215e+09          27.005          25.061        0.0        0.0
2   3075.333    2.248e+09          27.005          25.061        0.0        0.0
3    269.933    1.704e+06          27.005          25.061        0.0        0.0
4   3177.000    1.994e+07          27.005          25.061        0.0        0.0
```

```
In [32]: y.head()
```

```
Out[32]: 0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: AC_POWER, dtype: float64
```

```
In [33]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.2,random_state=21)
```

Linear Regression

```
In [34]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
score_lr = 100*lr_clf.score(X_test,y_test)
print(f'LR Model score = {score_lr:.4f}%')
```

LR Model score = 99.9994%

```
In [35]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

lr = LinearRegression()
lr.fit(X_train,y_train)
y_pred_lr = lr.predict(X_test)
```

```
R2_Score_lr = round(r2_score(y_pred_lr,y_test) * 100, 2)
print("R2 Score : ",R2_Score_lr,"%")
```

R2 Score : 100.0 %

Random Forest Regressor

```
In [36]: from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)
y_pred_rfr = lr.predict(X_test)
R2_Score_rfr = round(r2_score(y_pred_rfr,y_test) * 100, 2)

print("R2 Score : ",R2_Score_rfr,"%")
```

R2 Score : 100.0 %

Decision Tree Regressor

```
In [37]: from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
dtr.fit(X_train,y_train)

y_pred_dtr = lr.predict(X_test)
R2_Score_dtr = round(r2_score(y_pred_dtr,y_test) * 100, 2)

print("R2 Score : ",R2_Score_dtr,"%")
```

R2 Score : 100.0 %

Result Prediction

```
In [38]: prediction = rfr.predict(X_test)
print(prediction)

[ 0.          0.          684.7520619 ...  0.        1007.14337619
 0.          ]
```

```
In [39]: cross_checking = pd.DataFrame({'Actual' : y_test , 'Predicted' : prediction})
cross_checking.head()
```

```
Out[39]:
```

	Actual	Predicted
40426	0.000	0.000
50974	0.000	0.000
53919	684.913	684.752
2384	0.000	0.000
22014	0.000	0.000

```
In [40]: cross_checking['Error'] = cross_checking['Actual'] - cross_checking['Predicted']
cross_checking.head()
```

```
Out[40]:
```

	Actual	Predicted	Error
40426	0.000	0.000	0.000
50974	0.000	0.000	0.000
53919	684.913	684.752	0.161
2384	0.000	0.000	0.000
22014	0.000	0.000	0.000

```
In [41]: cross_checking_final = cross_checking[cross_checking['Error'] <= 20]
cross_checking_final.sample(25).style.background_gradient(
    cmap='coolwarm').set_properties(**{
        'font-family': 'Lucida Calligraphy',
        'color': 'LightGreen',
        'font-size': '15px'
    })
```

Out[41]:

	Actual	Predicted	Error
14845	533.993	536.231	-2.238
28528	672.353	672.078	0.275
18557	0.000	0.000	0.000
26191	0.000	0.000	0.000
39010	1100.893	1100.792	0.101
1149	314.713	316.744	-2.031
28056	0.000	0.000	0.000
38899	525.771	525.781	-0.009
62563	394.853	394.649	0.204
36805	491.086	491.262	-0.177
67422	0.000	0.000	0.000
60295	652.140	653.336	-1.196
35249	502.186	502.576	-0.391
32399	25.387	25.386	0.001
55466	0.000	0.000	0.000
47387	669.421	669.760	-0.339
54605	0.000	0.000	0.000
22604	0.000	0.000	0.000
41249	0.000	0.000	0.000
28809	1173.093	1172.785	0.309
56261	641.520	641.439	0.081
54549	9.593	9.589	0.004
55212	0.000	0.000	0.000
48315	0.000	0.000	0.000
63586	0.000	0.000	0.000