Solar Power generation of two plants and there climate data analytics project

Data Set ->

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

pd.set_option('display.max_columns', None) sets the maximum number of columns to display to None, which means all columns will be displayed. This is useful when working with datasets that have a large number of columns, as it allows you to see all of the data without truncation.

pd.set_option('display.max_rows', None) sets the maximum number of rows to display to None, which means all rows will be displayed. This is useful when working with datasets that have a large number of rows, as it allows you to see all of the data without truncation.

pd.set_option('precision', 3) sets the precision for floating-point numbers to 3 decimal places. This means that any floating-point numbers in the dataset will be displayed with 3 decimal places.

```
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
#pd.set_option('precision',3)
```

```
generation_plant1 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Sloar power generation vs weather/Solar Power vs Weather/Plant_1_Gene
generation_plant2 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Sloar power generation vs weather/Solar Power vs Weather/Plant_2_Gene
weather_plant1 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Sloar power generation vs weather/Solar Power vs Weather/Plant_1_Weather
weather_plant2 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Sloar power generation vs weather/Solar Power vs Weather/Plant_2_Weather
```

```
generation_plant1.sample(5).style.set_properties(
 **{
 'background-color': 'OliveDrab',
 'color': 'white',
 'border-color': 'darkblack'
 })
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
26132	28-05-2020 08:00	4135001	McdE0feGgRqW7Ca	5048.750000	494.950000	422.125000	7256811.125000
22488	26-05-2020 13:30	4135001	1BY6WEcLGh8j5v7	11568.250000	1128.625000	5244.250000	6341452.250000
42998	05-06-2020 17:30	4135001	1BY6WEcLGh8j5v7	3253.625000	319.700000	7035.375000	6412415.375000
62007	14-06-2020 18:30	4135001	VHMLBKoKgIrUVDU	66.500000	6.412500	7488.750000	7437054.750000
24657	27-05-2020 14:30	4135001	ZoEaEvLYb1n2sOq	9484.428571	926.414286	5139.285714	7191829.286000

```
weather_plant1.sample(5).style.set_properties(
 **{
 'background-color': 'OliveDrab',
 'color': 'white',
 'border-color': 'darkblack'
 })
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
613	2020-05-21 23:45:00	4135001	HmiyD2TTLFNqkNe	23.856805	21.155121	0.000000
252	2020-05-17 17:45:00	4135001	HmiyD2TTLFNqkNe	26.728900	27.917517	0.061723
583	2020-05-21 16:15:00	4135001	HmiyD2TTLFNqkNe	31.034047	37.628568	0.239683
2416	2020-06-10 00:30:00	4135001	HmiyD2TTLFNqkNe	22.709959	20.856700	0.000000
2090	2020-06-06 15:00:00	4135001	HmiyD2TTLFNqkNe	25.105676	26.625590	0.254935

```
generation_plant2.sample(5).style.set_properties(
 **{
 'background-color': 'cadetblue',
 'color': 'white',
 'border-color': 'darkblack'
 })
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
18842	2020-05-24 23:30:00	4136001	V94E5Ben1TIhnDV	0.000000	0.000000	9263.000000	1412158846.000000
24660	2020-05-28 08:15:00	4136001	q49J1IKaHRwDQnt	605.173333	592.973333	562.666667	415485.666667
67186	2020-06-17 18:00:00	4136001	oZZkBaNadn6DNKz	62.013333	60.046667	4368.600000	1708287703.600000
10750	2020-05-20 06:15:00	4136001	PeE6FRyGXUgsRhN	43.378571	41.942857	9.000000	1348378356.000000
58234	2020-06-13 12:15:00	4136001	rrq4fwE8jgrTyWY	1164.466667	1136.573333	2532.600000	121107692.600000

```
weather_plant2.sample(5).style.set_properties(
 **{
 'background-color': 'cadetblue',
 'color': 'white',
 'border-color': 'darkblack'
 })
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
3019	2020-06-15 12:00:00	4136001	iq8k7ZNt4Mwm3w0	29.900189	38.873928	0.574809
2803	2020-06-13 06:00:00	4136001	iq8k7ZNt4Mwm3w0	22.766307	22.510003	0.004778
2258	2020-06-07 13:45:00	4136001	iq8k7ZNt4Mwm3w0	33.778215	54.161322	0.798034
3176	2020-06-17 03:15:00	4136001	iq8k7ZNt4Mwm3w0	22.963709	22.606509	0.000000
10	2020-05-15 02:30:00	4136001	iq8k7ZNt4Mwm3w0	26.260399	24.482407	0.000000

```
generation_plant1['DATE_TIME'] = pd.to_datetime(generation_plant1['DATE_TIME'])
generation_plant2['DATE_TIME'] = pd.to_datetime(generation_plant2['DATE_TIME'])
weather_plant1['DATE_TIME'] = pd.to_datetime(weather_plant1['DATE_TIME'])
weather_plant2['DATE_TIME'] = pd.to_datetime(weather_plant2['DATE_TIME'])
```

```
df_solar_plant1 = pd.merge(generation_plant1.drop(columns = ['PLANT_ID']), weather_plant1.drop(columns = ['PLANT_ID', 'SOURCE_KEY']), on='DAT
df_solar_plant1.sample(5).style.background_gradient(cmap='cool')
```

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRAD
24963	2020-05-27 18:00:00	YxYtjZvoooNbGkE	255.750000	24.700000	6695.250000	7276028.250000	29.454140	28.887065	0.
9399	2020-05-19 15:15:00	bvBOhCH3iADSZry	5476.857143	537.000000	5429.000000	6345967.000000	30.035451	43.317071	0.
1076	2020-05-15 12:15:00	wCURE6d3bPkepu2	9891.000000	966.314286	2961.428571	6785559.429000	31.507298	49.844493	0.
21220	2020-05-25	ZnxXDlPa8U1GXgE	0.000000	0.000000	8910.000000	6604666.000000	22.538633	20.395795	0

```
if len(df_solar_plant1) >= 5:
    df_solar_plant1.sample(5)
else:
    print("Not enough rows to sample")
```

```
df_solar_plant2 = pd.merge(generation_plant2.drop(columns = ['PLANT_ID']), weather_plant2.drop(columns = ['PLANT_ID', 'SOURCE_KEY']), on='DAT
df_solar_plant2.sample(5).style.background_gradient(cmap='cool')
```

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IR
41609	2020-06-05 15:30:00	Et9kgGMDl729KT4	492.340000	482.866667	1903.800000	1813068.800000	29.233108	36.759183	
3889	2020-05-16 20:15:00	q49J1IKaHRwDQnt	0.000000	0.000000	3533.000000	352552.000000	28.205945	27.102159	
65614	2020-06-17 00:15:00	LIT2YUhhzqhg5Sw	0.000000	0.000000	0.000000	282784844.000000	24.140737	23.967333	
47976	2020-06-08	V94E5Ben1TlhnDV	843.946667	825.746667	8141.333333	1412250613.333333	34.780828	48.181005	

```
# adding separate time and date columns
df_solar_plant1["DATE"] = pd.to_datetime(df_solar_plant1["DATE_TIME"]).dt.date
df_solar_plant1["TIME"] = pd.to_datetime(df_solar_plant1["DATE_TIME"]).dt.time
df_solar_plant1['DAY'] = pd.to_datetime(df_solar_plant1['DATE_TIME']).dt.day
df_solar_plant1['MONTH'] = pd.to_datetime(df_solar_plant1['DATE_TIME']).dt.month
df_solar_plant1['WEEK'] = pd.to_datetime(df_solar_plant1['DATE_TIME']).dt.week


# add hours and minutes for ml models
df_solar_plant1['HOURS'] = pd.to_datetime(df_solar_plant1['TIME'],format='%H:%M:%S').dt.hour
df_solar_plant1['MINUTES'] = pd.to_datetime(df_solar_plant1['TIME'],format='%H:%M:%S').dt.minute
df_solar_plant1['TOTAL MINUTES PASS'] = df_solar_plant1['MINUTES'] + df_solar_plant1['HOURS']*60

# add date as string column
df_solar_plant1["DATE_STRING"] = df_solar_plant1["DATE"].astype(str) # add column with date as string
df_solar_plant1["HOURS"] = df_solar_plant1["HOURS"].astype(str)
df_solar_plant1["TIME"] = df_solar_plant1["TIME"].astype(str)
df_solar_plant1.head()
```

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DA
0	2020-05-15	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0	25.184316	22.857507	0.0	202 05-
1	2020-05-15	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0	25.184316	22.857507	0.0	202 05-
2	2020-05-15	3PZuoBAID5Wc2HD	0.0	0.0	0.0	6987759.0	25.184316	22.857507	0.0	202 05-
3	2020-05-15	7JYdWkrLSPkdwr4	0.0	0.0	0.0	7602960.0	25.184316	22.857507	0.0	202 05-
4	2020-05-15	McdE0feGgRqW7Ca	0.0	0.0	0.0	7158964.0	25.184316	22.857507	0.0	202 05-

```
# adding separate time and date columns
df_solar_plant2["DATE"] = pd.to_datetime(df_solar_plant2["DATE_TIME"]).dt.date
df_solar_plant2["TIME"] = pd.to_datetime(df_solar_plant2["DATE_TIME"]).dt.time
df_solar_plant2['DAY'] = pd.to_datetime(df_solar_plant2['DATE_TIME']).dt.day
df_solar_plant2['MONTH'] = pd.to_datetime(df_solar_plant2['DATE_TIME']).dt.month
df_solar_plant2['WEEK'] = pd.to_datetime(df_solar_plant2['DATE_TIME']).dt.week

# add hours and minutes for ml models
df_solar_plant2['HOURS'] = pd.to_datetime(df_solar_plant2['TIME'],format='%H:%M:%S').dt.hour
df_solar_plant2['MINUTES'] = pd.to_datetime(df_solar_plant2['TIME'],format='%H:%M:%S').dt.minute
df_solar_plant2['TOTAL MINUTES PASS'] = df_solar_plant2['MINUTES'] + df_solar_plant2['HOURS']*60

# add date as string column
df_solar_plant2["DATE_STRING"] = df_solar_plant2["DATE"].astype(str) # add column with date as string
df_solar_plant2["HOURS"] = df_solar_plant2["HOURS"].astype(str)
df_solar_plant2["TIME"] = df_solar_plant2["TIME"].astype(str)
df_solar_plant2.head()
```

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	D
0	2020-05-15	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06	27.004764	25.060789	0.0	20 05
1	2020-05-15	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09	27.004764	25.060789	0.0	20 05
2	2020-05-15	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09	27.004764	25.060789	0.0	20 05
3	2020-05-15	Et9kgGMDl729KT4	0.0	0.0	269.933333	1.704250e+06	27.004764	25.060789	0.0	20 05
4	2020-05-15	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07	27.004764	25.060789	0.0	20 05

```
df_solar_plant1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45680 entries, 0 to 45679
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   DATE_TIME            45680 non-null  datetime64[ns]
 1   SOURCE_KEY           45680 non-null  object
 2   DC_POWER             45680 non-null  float64
 3   AC_POWER             45680 non-null  float64
 4   DAILY_YIELD          45680 non-null  float64
 5   TOTAL_YIELD          45680 non-null  float64
 6   AMBIENT_TEMPERATURE  45680 non-null  float64
 7   MODULE_TEMPERATURE   45680 non-null  float64
 8   IRRADIATION          45680 non-null  float64
 9   DATE                 45680 non-null  object
 10  TIME                 45680 non-null  object
 11  DAY                  45680 non-null  int64
 12  MONTH                45680 non-null  int64
 13  WEEK                 45680 non-null  int64
 14  HOURS                45680 non-null  object
 15  MINUTES              45680 non-null  int64
 16  TOTAL MINUTES PASS   45680 non-null  int64
 17  DATE_STRING          45680 non-null  object
dtypes: datetime64[ns](1), float64(7), int64(5), object(5)
memory usage: 6.6+ MB
```

```
df_solar_plant2.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
```

```
df_solar_plant1.describe().style.background_gradient(cmap='rainbow')
```

	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DAY	
count	45680.000000	45680.000000	45680.000000	45680.000000	45680.000000	45680.000000	45680.000000	45680.000000	4568
mean	3197.175971	312.652679	3313.146538	6957007.021147	25.917168	31.877975	0.236834	20.414317	
std	4080.448523	398.668968	3156.100252	417238.643557	3.556550	12.638448	0.306316	6.258661	
min	0.000000	0.000000	0.000000	6183645.000000	20.398505	18.140415	0.000000	6.000000	
25%	0.000000	0.000000	0.000000	6512357.875000	22.930031	21.406390	0.000000	16.000000	
50%	464.196429	44.912500	2653.633928	7115710.714000	24.993020	25.379072	0.035266	20.000000	
75%	6478.424107	634.481250	6318.000000	7244521.410750	28.379008	42.757119	0.459503	26.000000	
max	14471.125000	1410.950000	9163.000000	7846821.000000	35.252486	65.545714	1.221652	31.000000	

```
df_solar_plant2.describe().style.background_gradient(cmap='rainbow')
```

	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DAY	
count	67698.000000	67698.000000	67698.000000	67698.000000	67698.000000	67698.000000	67698.000000	67698.000000	6
mean	246.701961	241.277825	3294.890295	658944788.423766	27.986756	32.607233	0.229204	15.530680	
std	370.569597	362.112118	2919.448386	729667771.073241	4.021294	11.226446	0.309365	8.527546	
min	0.000000	0.000000	0.000000	0.000000	20.942385	20.265123	0.000000	1.000000	
25%	0.000000	0.000000	272.750000	19964944.866667	24.570349	23.685627	0.000000	9.000000	
50%	0.000000	0.000000	2911.000000	282627587.000000	26.910352	27.433723	0.018554	16.000000	
75%	446.591667	438.215000	5534.000000	1348495113.000000	30.912601	40.019036	0.431027	22.000000	
max	1420.933333	1385.420000	9873.000000	2247916295.000000	39.181638	66.635953	1.098766	31.000000	

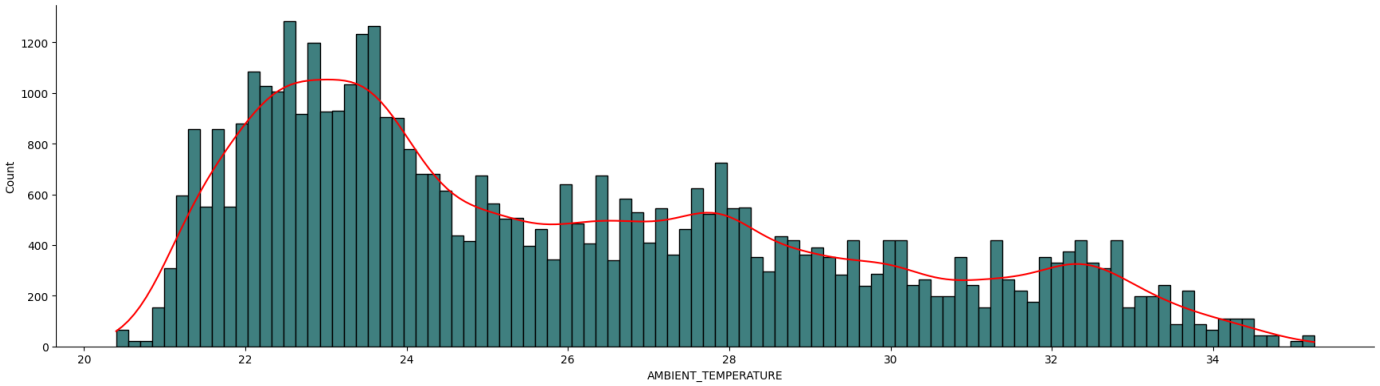
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df_solar_plant1['SOURCE_KEY_NUMBER'] = encoder.fit_transform(df_solar_plant1['SOURCE_KEY'])
df_solar_plant1.head()
```

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DA
0	2020-05-15	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0	25.184316	22.857507	0.0	202 05-
1	2020-05-15	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0	25.184316	22.857507	0.0	202 05-
2	2020-05-15	3PZuoBAID5Wc2HD	0.0	0.0	0.0	6987759.0	25.184316	22.857507	0.0	202 05-
3	2020-05-15	7JYdWkrLSPkdwr4	0.0	0.0	0.0	7602960.0	25.184316	22.857507	0.0	202 05-
4	2020-05-15	McdE0feGgRqW7Ca	0.0	0.0	0.0	7158964.0	25.184316	22.857507	0.0	202 05-

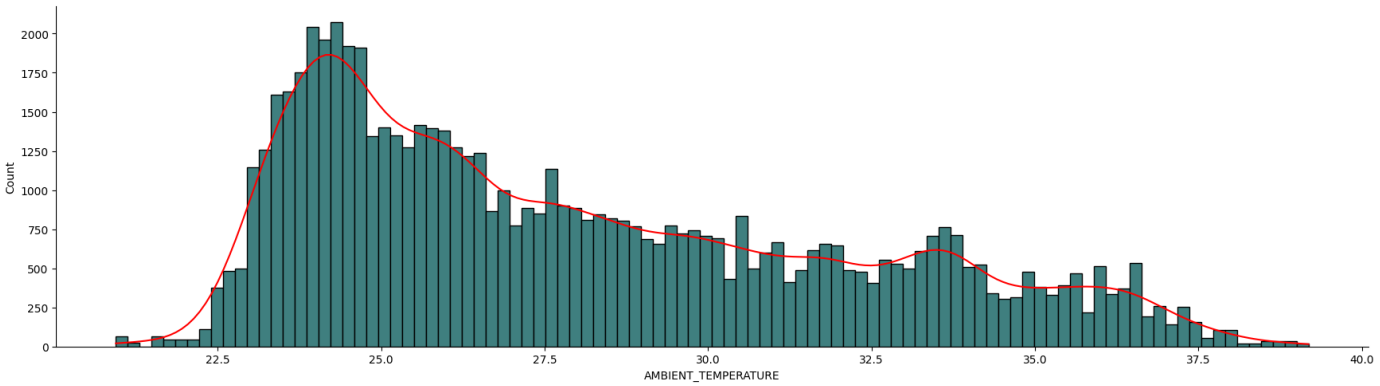
```
df_solar_plant2['SOURCE_KEY_NUMBER'] = encoder.fit_transform(df_solar_plant2['SOURCE_KEY'])
df_solar_plant2.head(200)
```

	DATE_TIME	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATIO
0	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06	27.004764	25.060789	0
1	2020-05-15 00:00:00	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09	27.004764	25.060789	0
2	2020-05-15 00:00:00	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09	27.004764	25.060789	0
3	2020-05-15 00:00:00	Et9kgGMDl729KT4	0.0	0.0	269.933333	1.704250e+06	27.004764	25.060789	0
4	2020-05-15 00:00:00	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07	27.004764	25.060789	0
5	2020-05-15 00:00:00	LYwnQax7tkwH5Cb	0.0	0.0	1872.500000	1.794959e+09	27.004764	25.060789	0
6	2020-05-15 00:00:00	LIT2YUhhzqhg5Sw	0.0	0.0	1094.357143	2.825928e+08	27.004764	25.060789	0
7	2020-05-15 00:00:00	Mx2yZCDsyf6DPfv	0.0	0.0	5692.200000	2.453646e+06	27.004764	25.060789	0
8	2020-05-15 00:00:00	NgDl19wMapZy17u	0.0	0.0	1866.200000	1.115126e+08	27.004764	25.060789	0
9	2020-05-15 00:00:00	PeE6FRyGXUgsRhN	0.0	0.0	651.200000	1.348351e+09	27.004764	25.060789	0
10	2020-05-15 00:00:00	Qf4GUc1pJu5T6c6	0.0	0.0	0.000000	8.384214e+08	27.004764	25.060789	0
11	2020-05-15 00:00:00	Quc1TzYxW2pYoWX	0.0	0.0	5495.000000	3.295091e+08	27.004764	25.060789	0
12	2020-05-15 00:00:00	V94E5Ben1TlhnDV	0.0	0.0	0.000000	1.412083e+09	27.004764	25.060789	0
13	2020-05-15 00:00:00	WcxssY2VbP4hApt	0.0	0.0	0.000000	1.816953e+08	27.004764	25.060789	0
14	2020-05-15 00:00:00	mqwcsP2rE7J0TFp	0.0	0.0	1238.533333	5.935800e+08	27.004764	25.060789	0
15	2020-05-15 00:00:00	oZ35aAeoifZaQzV	0.0	0.0	1281.466667	1.659965e+09	27.004764	25.060789	0
16	2020-05-15 00:00:00	oZZkBaNadn6DNKz	0.0	0.0	0.000000	1.708083e+09	27.004764	25.060789	0
17	2020-05-15 00:00:00	q49J1IKaHRwDQnt	0.0	0.0	4315.000000	3.399230e+05	27.004764	25.060789	0
18	2020-05-15 00:00:00	rrq4fwE8jgrTyWY	0.0	0.0	280.214286	1.209641e+08	27.004764	25.060789	0
19	2020-05-15 00:00:00	vOuJvMaM2sgwLmb	0.0	0.0	0.000000	2.211962e+06	27.004764	25.060789	0
20	2020-05-15 00:00:00	xMbIugepa2P7IBB	0.0	0.0	9166.000000	1.066566e+08	27.004764	25.060789	0
	2020-05								

```
sns.displot(data=df_solar_plant1, x="AMBIENT_TEMPERATURE", kde=True, bins = 100,color = "red", facecolor = "#3F7F7F",height = 5, aspect = 3.5
```




```
sns.displot(data=df_solar_plant2, x="AMBIENT_TEMPERATURE", kde=True, bins = 100,color = "red", facecolor = "#3F7F7F",height = 5, aspect = 3.5
```



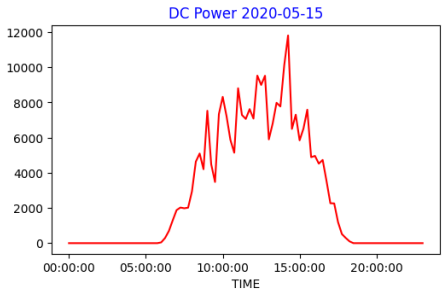

```
df_solar_plant1['DATE'].nunique()
```

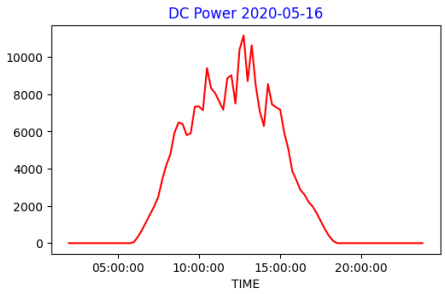
```
23
```

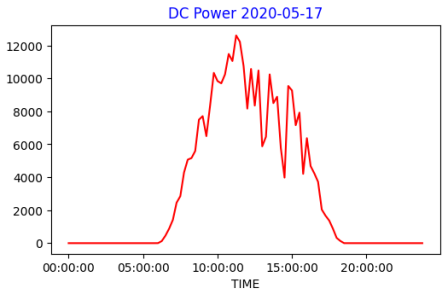
```
df_solar_plant2['DATE'].nunique()
```

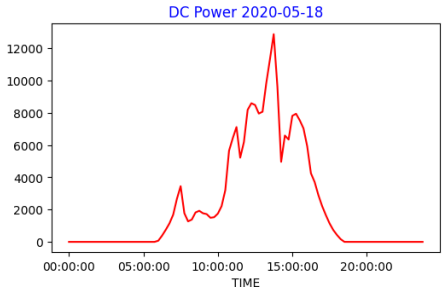
```
34
```

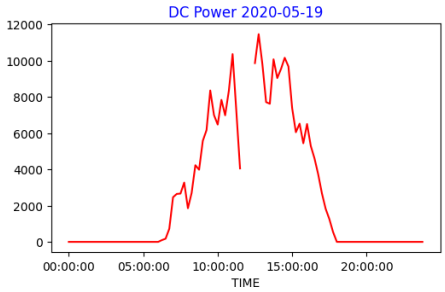
```
solar_dc1 = df_solar_plant1.pivot_table(values='DC_POWER', index='TIME', columns='DATE')

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_dc1, row=8, col=3)
```

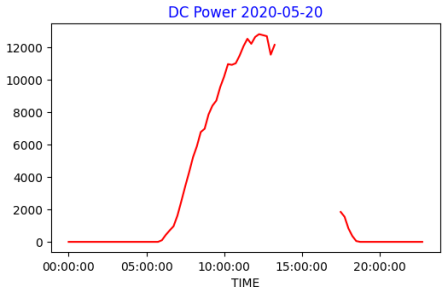

DC Power 2020-05-15
TIME

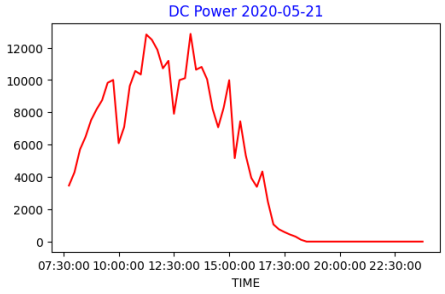

DC Power 2020-05-16
TIME

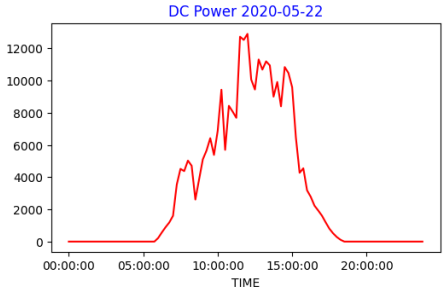

DC Power 2020-05-17
TIME

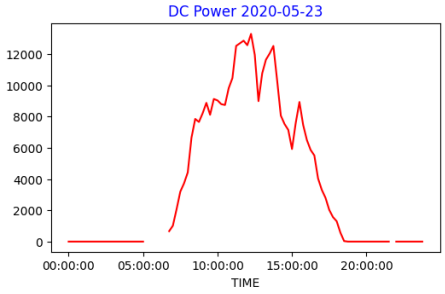

DC Power 2020-05-18
TIME

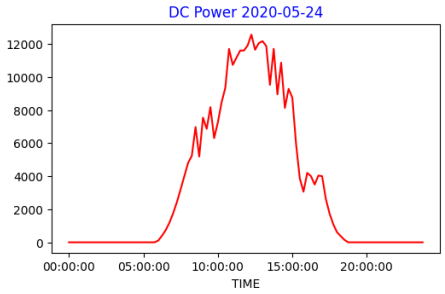

DC Power 2020-05-19
TIME

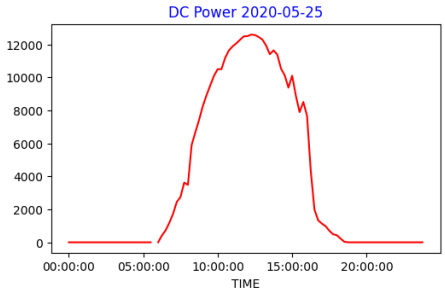

DC Power 2020-05-20
TIME

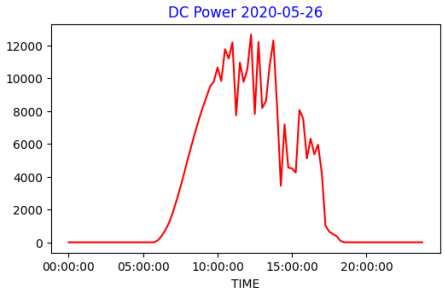

DC Power 2020-05-21
TIME

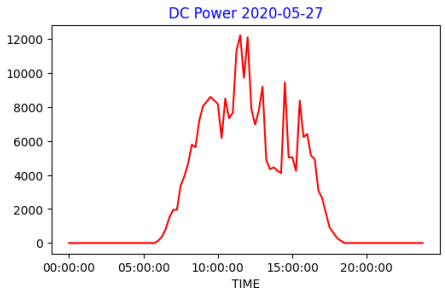

DC Power 2020-05-22
TIME

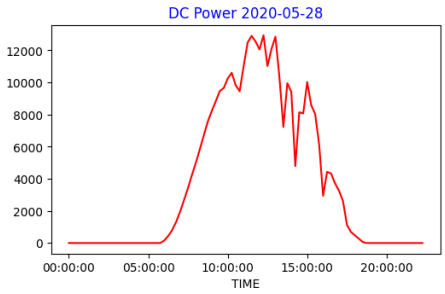

DC Power 2020-05-23
TIME

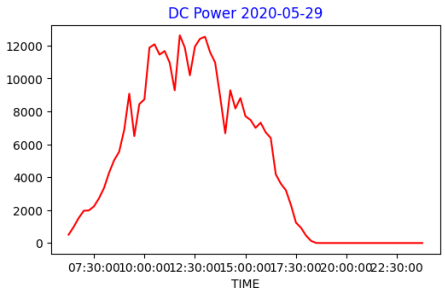

DC Power 2020-05-24
TIME

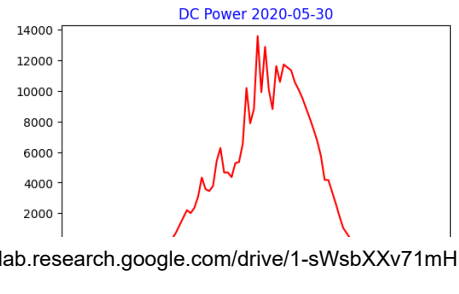

DC Power 2020-05-25
TIME

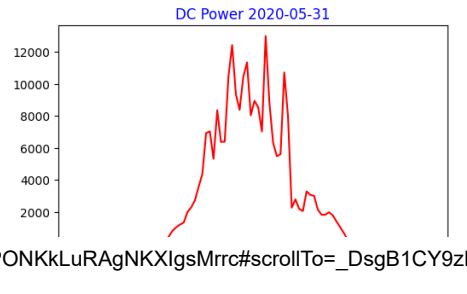

DC Power 2020-05-26
TIME

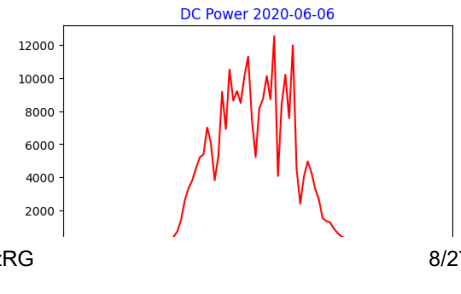

DC Power 2020-05-27
TIME


DC Power 2020-05-28
TIME

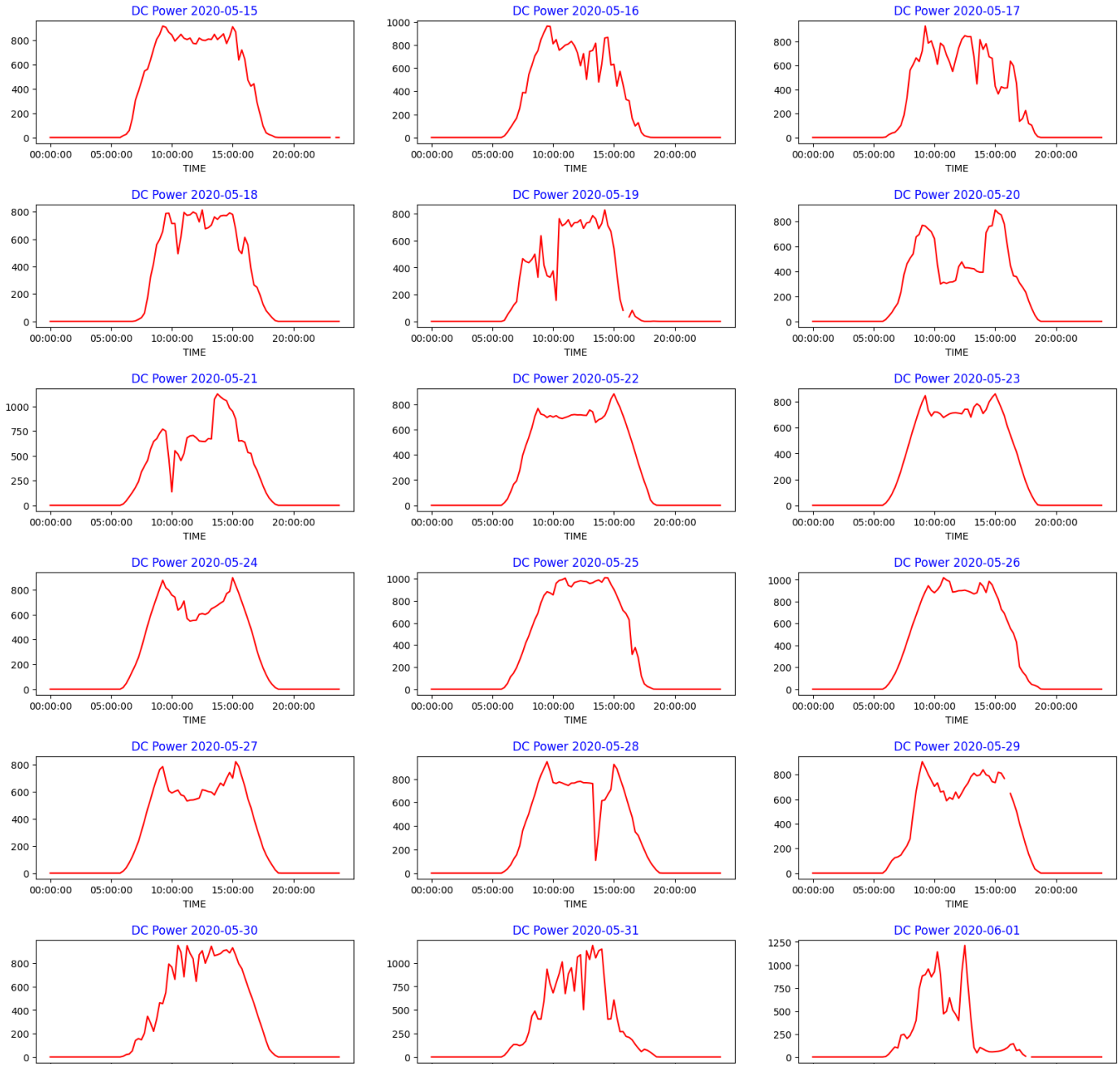

DC Power 2020-05-29
TIME


DC Power 2020-05-30

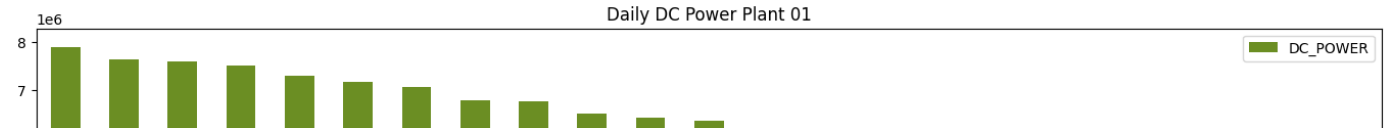

DC Power 2020-05-31


DC Power 2020-06-06

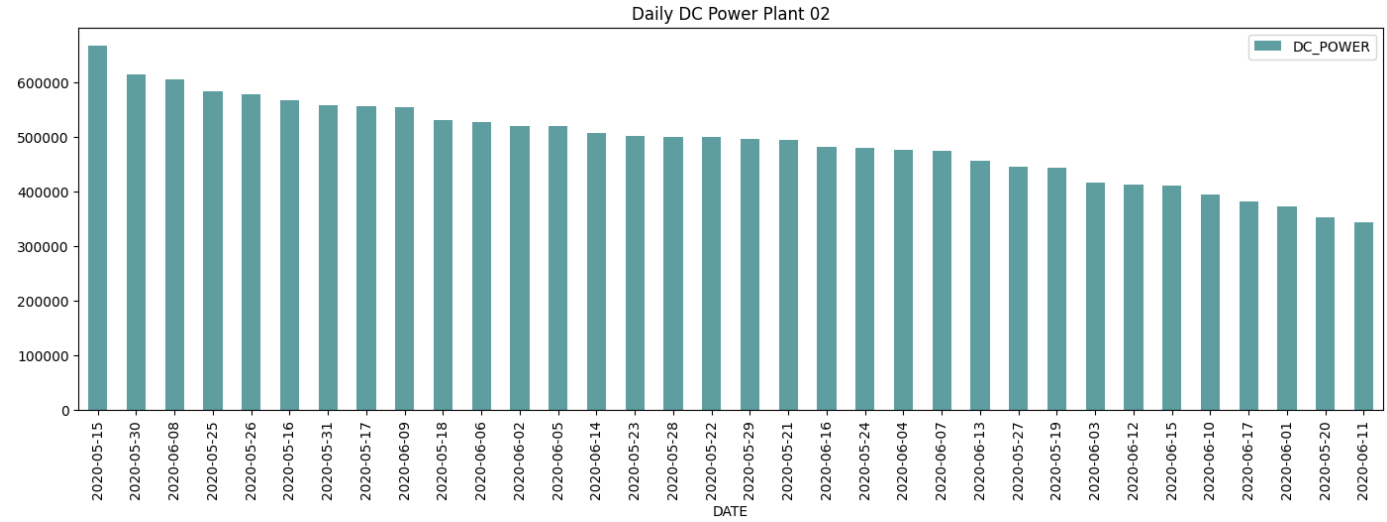
```
solar_dc2 = df_solar_plant2.pivot_table(values='DC_POWER', index='TIME', columns='DATE')

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_dc2, row=12, col=3)
```



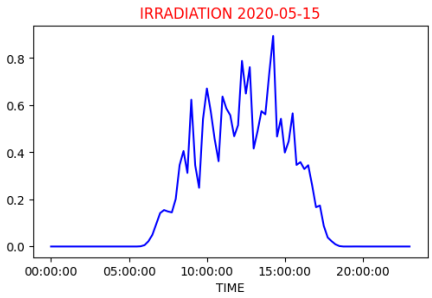

```
daily_dc_plant1 = df_solar_plant1.groupby('DATE')['DC_POWER'].agg('sum')

ax = daily_dc_plant1.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='OliveDrab')
plt.title('Daily DC Power Plant 01')
plt.show()
```

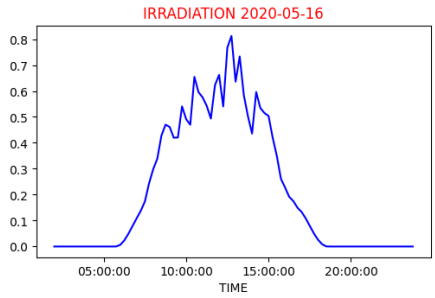


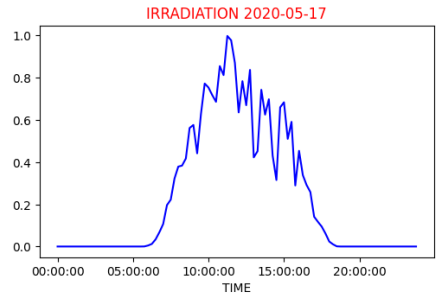

```
daily_dc_plant2 = df_solar_plant2.groupby('DATE')['DC_POWER'].agg('sum')

ax = daily_dc_plant2.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='cadetblue')
plt.title('Daily DC Power Plant 02')
plt.show()
```

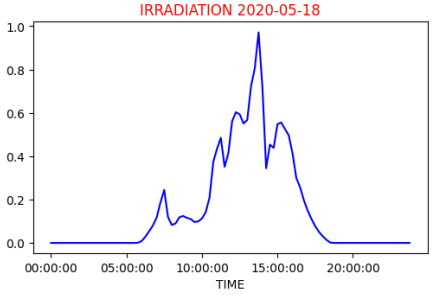


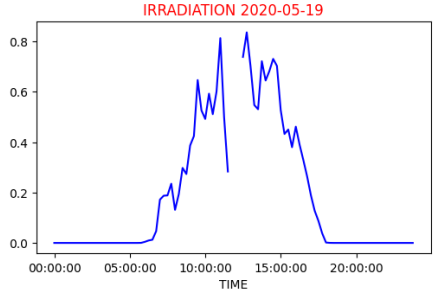

```
solar_irradiation_plant1 = df_solar_plant1.pivot_table(values='IRRADIATION', index='TIME', columns='DATE')

def Daywise_plot(data= None, row = None, col = None, title='IRRADIATION'):
    cols = data.columns
    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='red')

Daywise_plot(data=solar_irradiation_plant1, row=8, col=3)
```

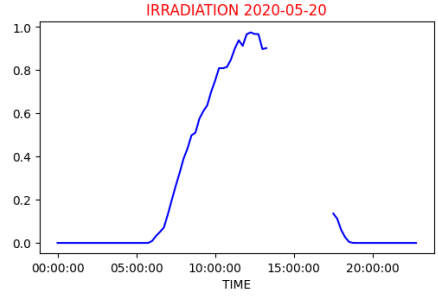

IRRADIATION 2020-05-15
TIME

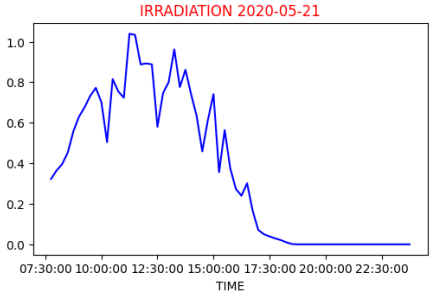

IRRADIATION 2020-05-16
TIME

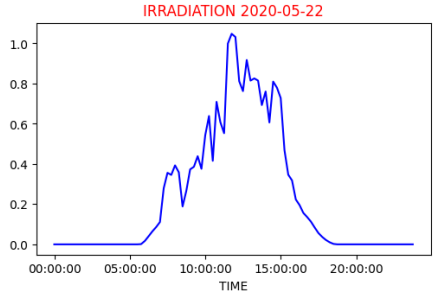

IRRADIATION 2020-05-17
TIME

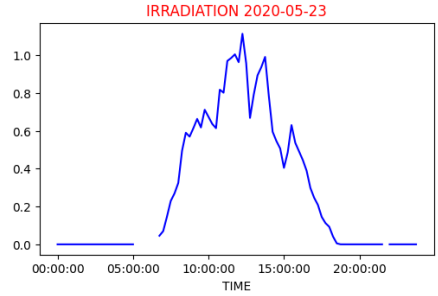

IRRADIATION 2020-05-18
TIME

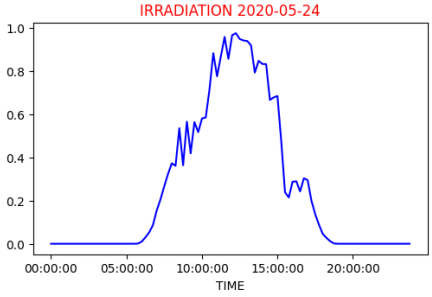

IRRADIATION 2020-05-19
TIME

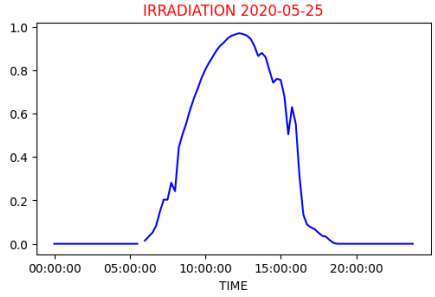

IRRADIATION 2020-05-20
TIME

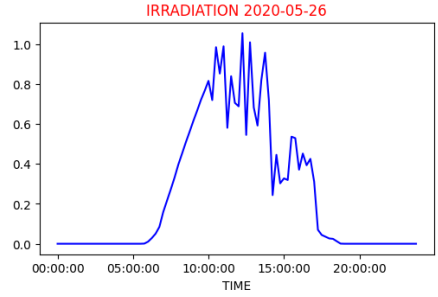

IRRADIATION 2020-05-21
TIME

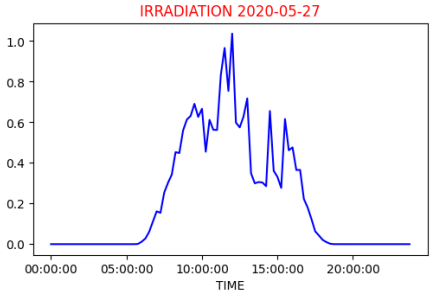

IRRADIATION 2020-05-22
TIME

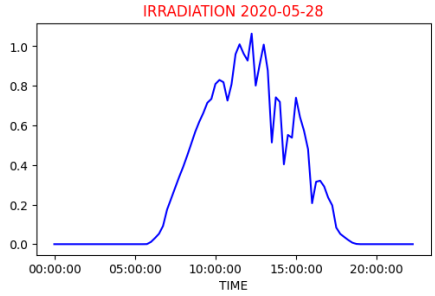

IRRADIATION 2020-05-23
TIME

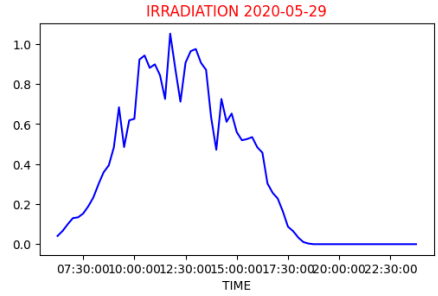

IRRADIATION 2020-05-24
TIME

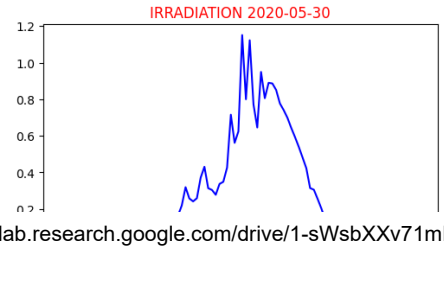

IRRADIATION 2020-05-25
TIME

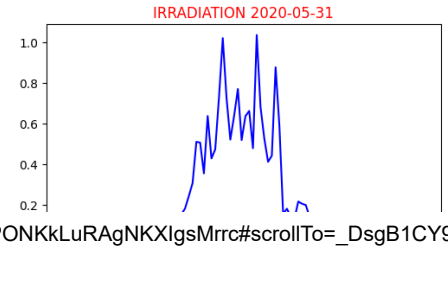

IRRADIATION 2020-05-26
TIME

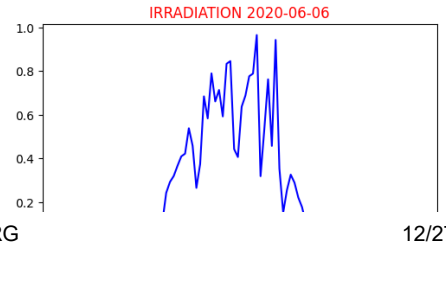

IRRADIATION 2020-05-27
TIME


IRRADIATION 2020-05-28
TIME

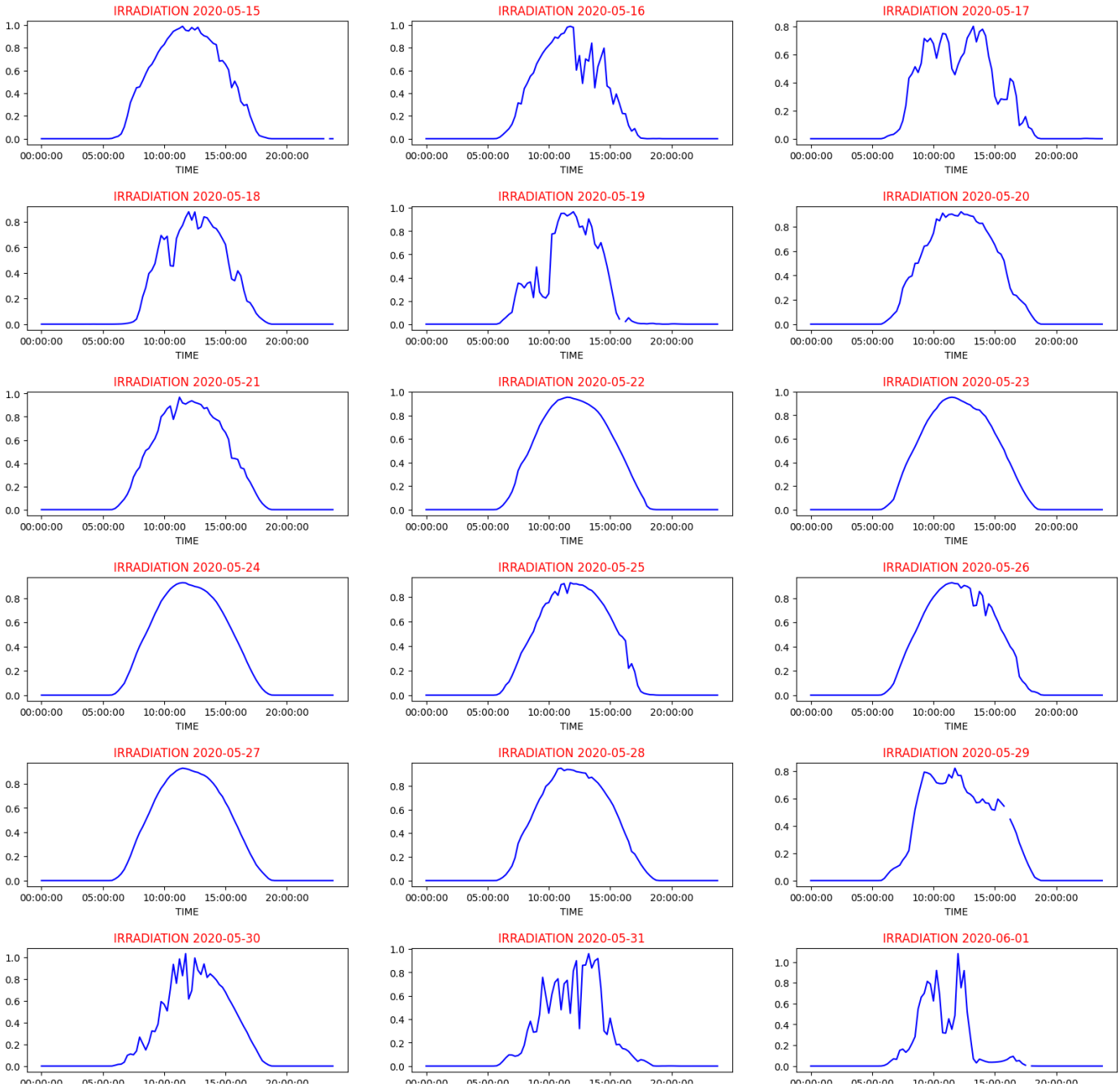

IRRADIATION 2020-05-29
TIME


IRRADIATION 2020-05-30

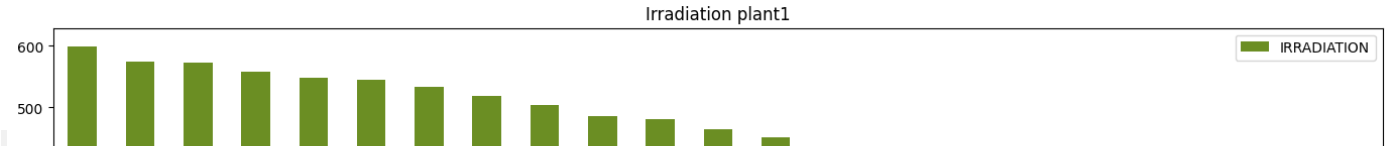

IRRADIATION 2020-05-31


IRRADIATION 2020-06-06

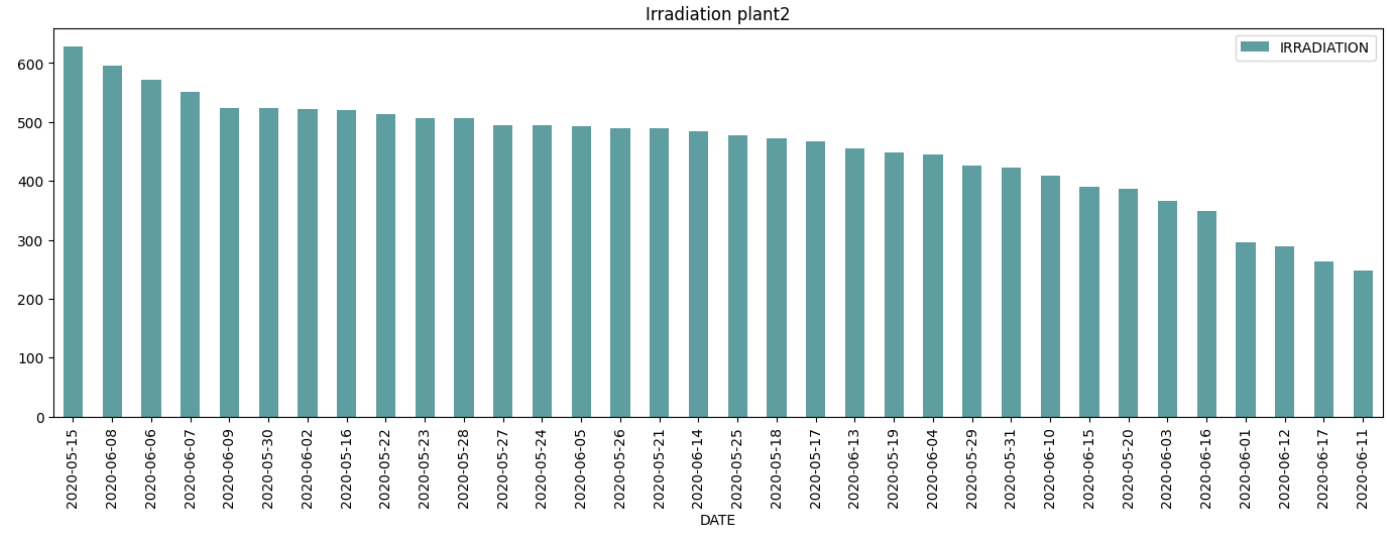
```
solar_irradiation_plant2 = df_solar_plant2.pivot_table(values='IRRADIATION', index='TIME', columns='DATE')

def Daywise_plot(data= None, row = None, col = None, title='IRRADIATION'):
    cols = data.columns
    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='red')

Daywise_plot(data=solar_irradiation_plant2, row=12, col=3)
```



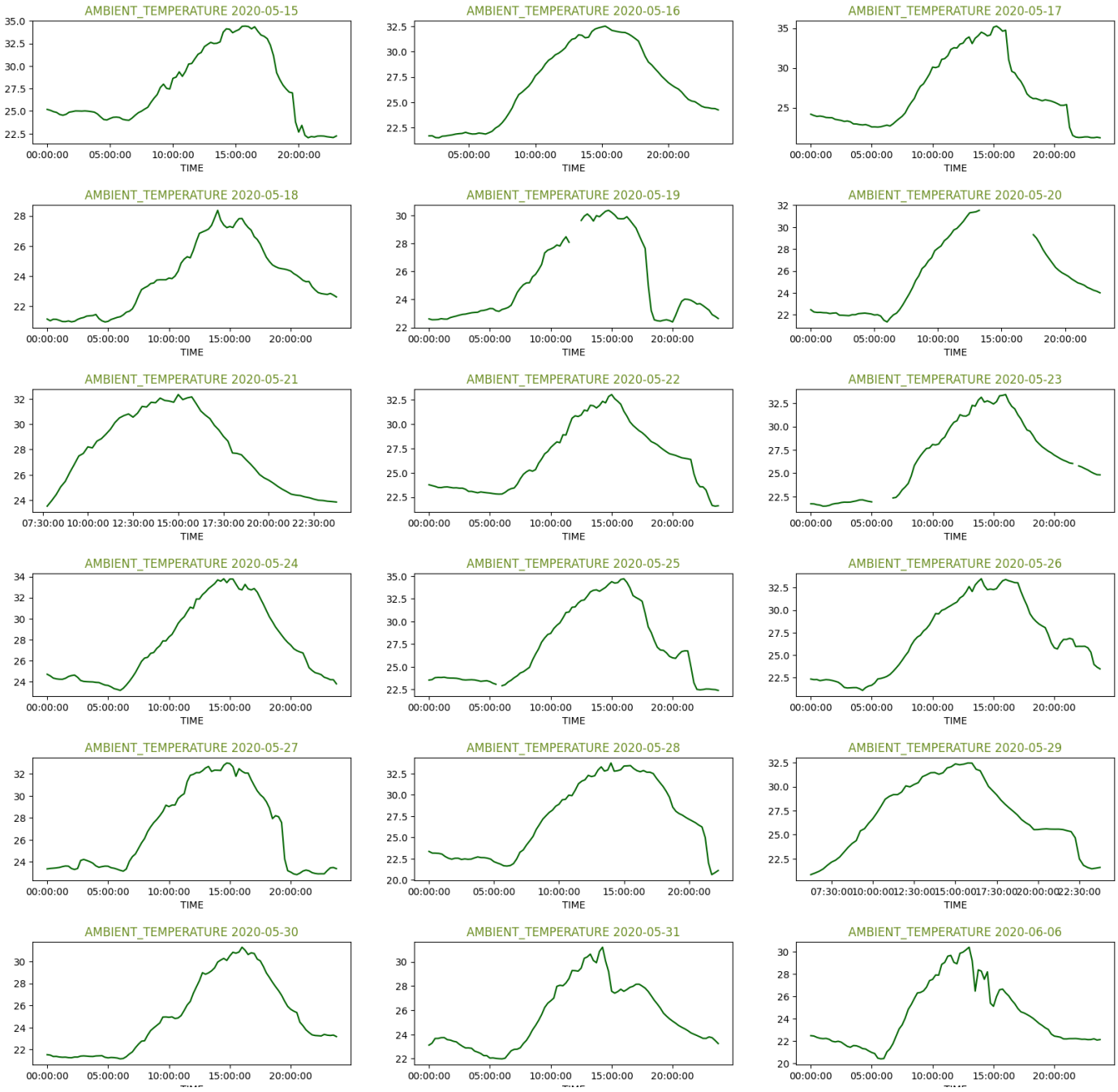

```
daily_irradiation_plant1 = df_solar_plant1.groupby('DATE')['IRRADIATION'].agg('sum')

daily_irradiation_plant1.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='OliveDrab')
plt.title('Irradiation plant1')
plt.show()
```



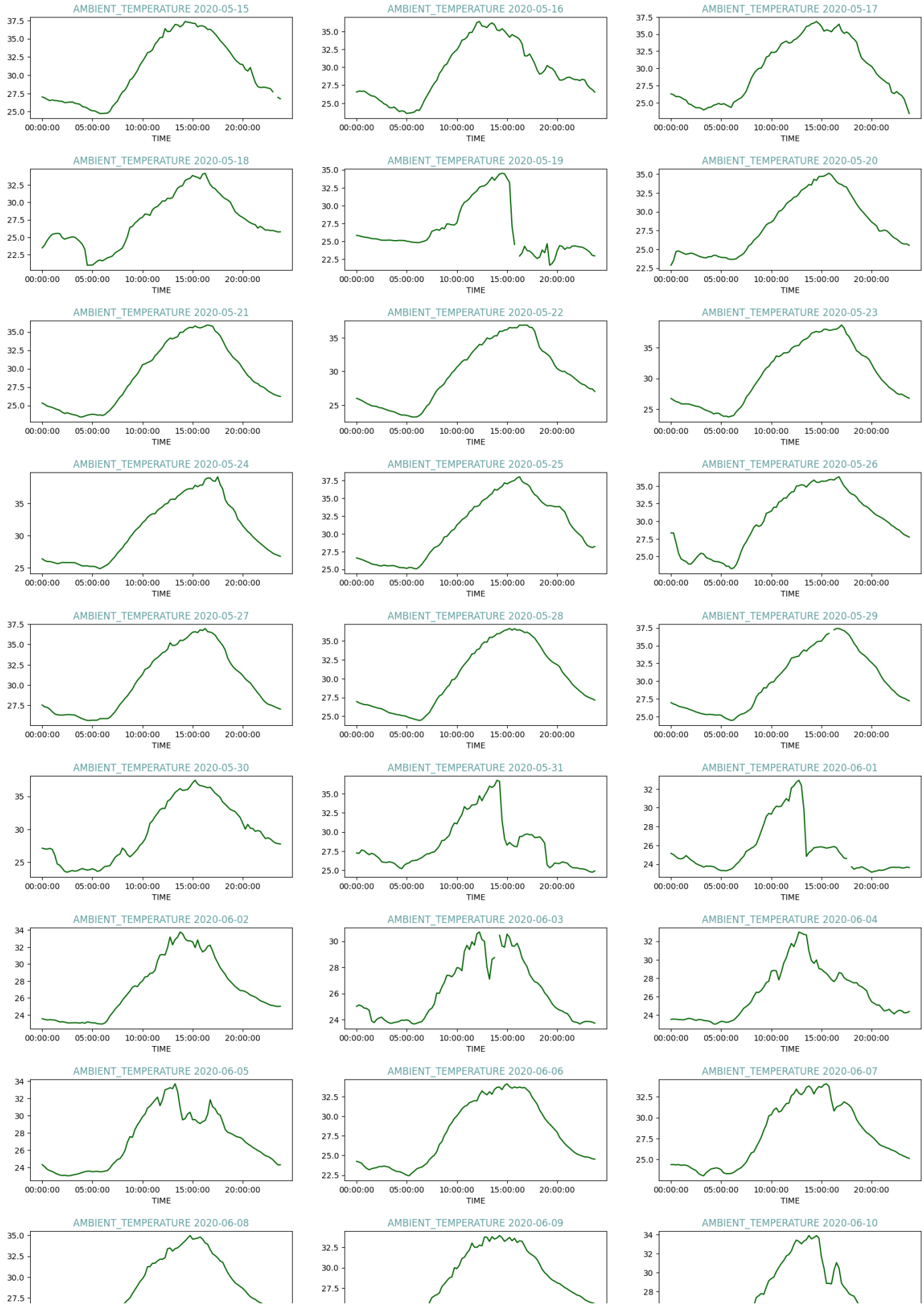

```
daily_irradiation_plant2 = df_solar_plant2.groupby('DATE')['IRRADIATION'].agg('sum')

daily_irradiation_plant2.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='cadetblue')
plt.title('Irradiation plant2')
plt.show()
```



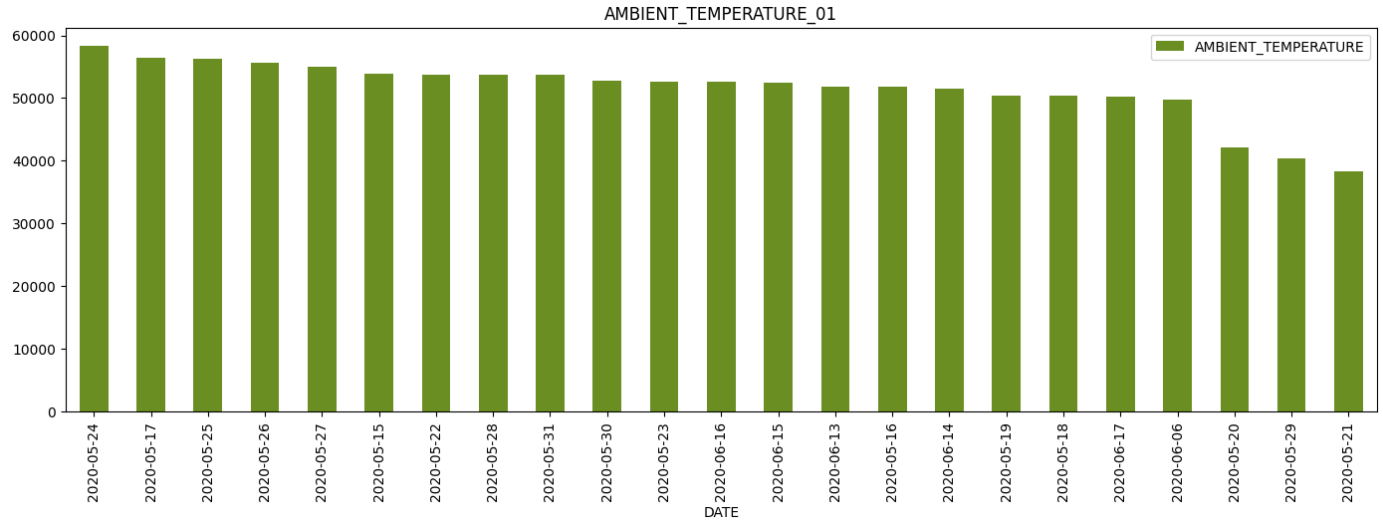

```
solar_ambiant_temp_1 = df_solar_plant1.pivot_table(values='AMBIENT_TEMPERATURE', index='TIME', columns='DATE')

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('{} {}'.format(title, cols[i-1]),color='OliveDrab')

Daywise_plot(data=solar_ambiant_temp_1, row=12, col=3)
```



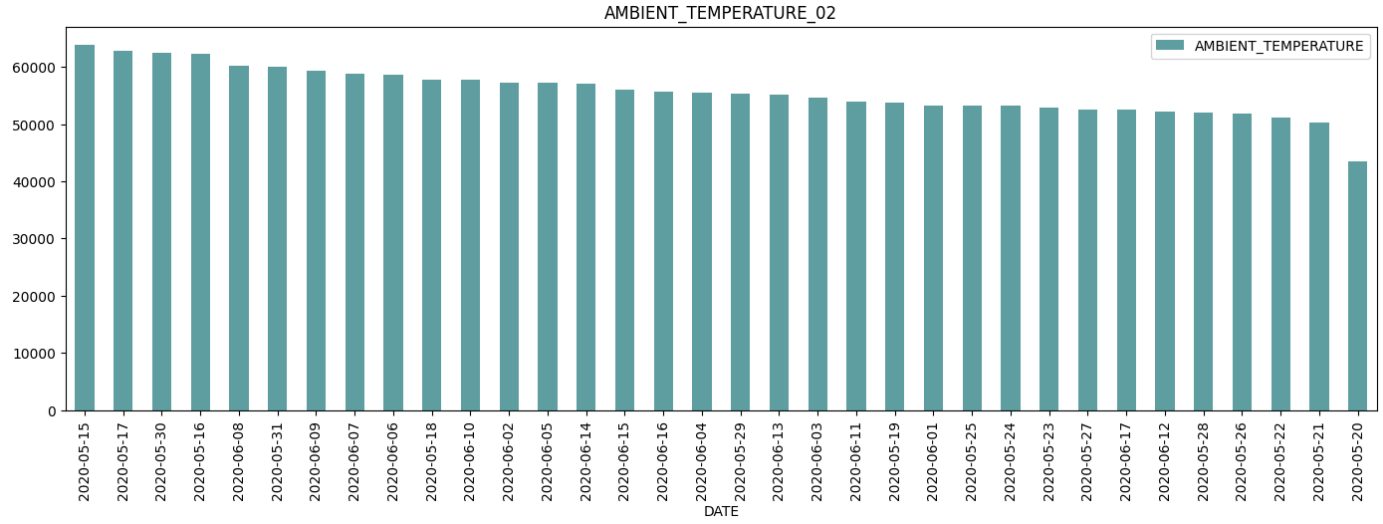

```
solar_ambiant_temp_2 = df_solar_plant2.pivot_table(values='AMBIENT_TEMPERATURE', index='TIME', columns='DATE')

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('{} {}'.format(title, cols[i-1]),color='cadetblue')

Daywise_plot(data=solar_ambiant_temp_2, row=12, col=3)
```


AMBIENT_TEMPERATURE 2020-05-15
AMBIENT_TEMPERATURE 2020-05-16
AMBIENT_TEMPERATURE 2020-05-17
AMBIENT_TEMPERATURE 2020-05-18
AMBIENT_TEMPERATURE 2020-05-19
AMBIENT_TEMPERATURE 2020-05-20
AMBIENT_TEMPERATURE 2020-05-21
AMBIENT_TEMPERATURE 2020-05-22
AMBIENT_TEMPERATURE 2020-05-23
AMBIENT_TEMPERATURE 2020-05-24
AMBIENT_TEMPERATURE 2020-05-25
AMBIENT_TEMPERATURE 2020-05-26
AMBIENT_TEMPERATURE 2020-05-27
AMBIENT_TEMPERATURE 2020-05-28
AMBIENT_TEMPERATURE 2020-05-29
AMBIENT_TEMPERATURE 2020-05-30
AMBIENT_TEMPERATURE 2020-05-31
AMBIENT_TEMPERATURE 2020-06-01
AMBIENT_TEMPERATURE 2020-06-02
AMBIENT_TEMPERATURE 2020-06-03
AMBIENT_TEMPERATURE 2020-06-04
AMBIENT_TEMPERATURE 2020-06-05
AMBIENT_TEMPERATURE 2020-06-06
AMBIENT_TEMPERATURE 2020-06-07
AMBIENT_TEMPERATURE 2020-06-08
AMBIENT_TEMPERATURE 2020-06-09
AMBIENT_TEMPERATURE 2020-06-10
TIME

```
daily_ambient_temp_1 = df_solar_plant1.groupby('DATE')['AMBIENT_TEMPERATURE'].agg('sum')

daily_ambient_temp_1.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='OliveDrab')
plt.title('AMBIENT_TEMPERATURE_01')
plt.show()
```

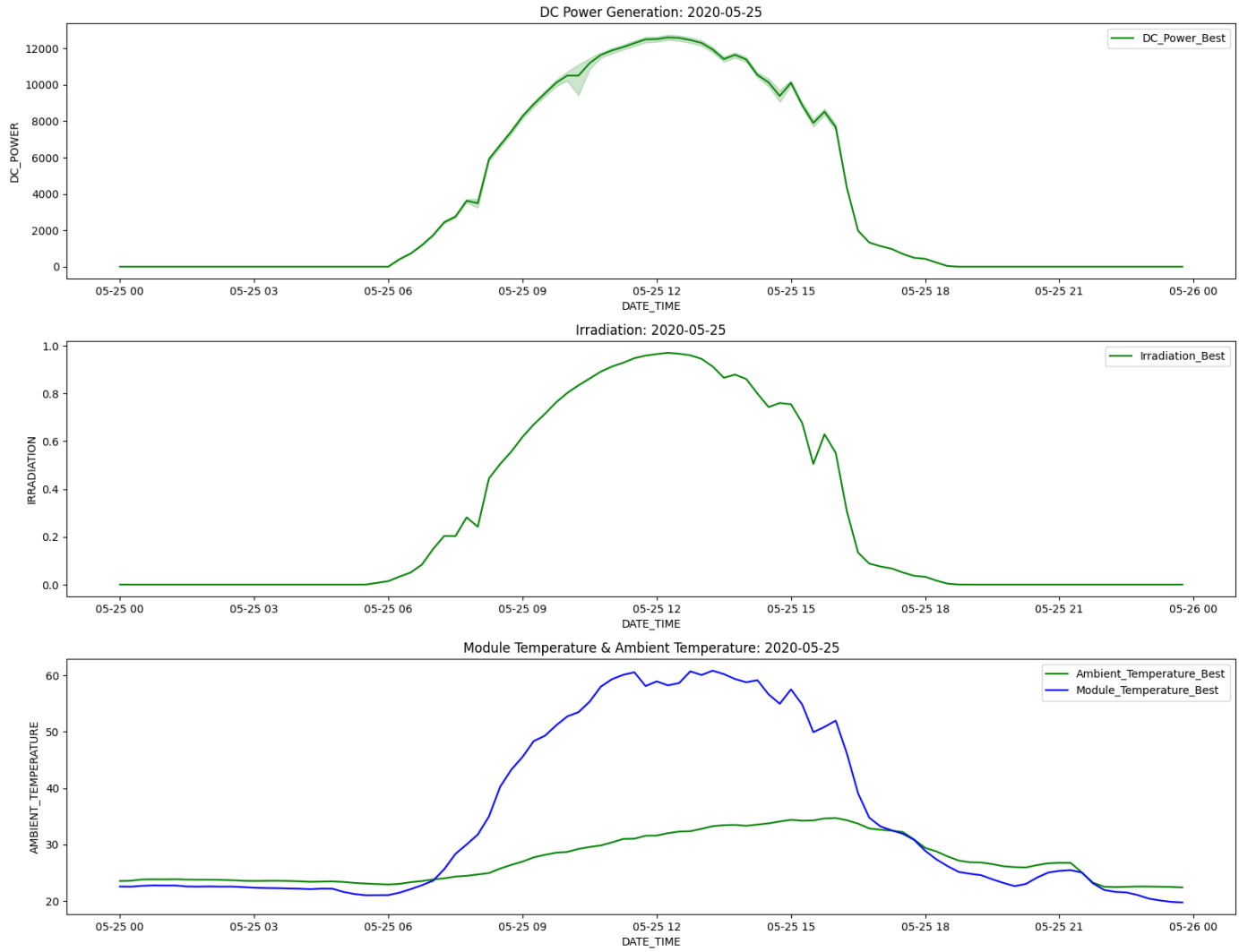



```
daily_ambient_temp_2 = df_solar_plant2.groupby('DATE')['AMBIENT_TEMPERATURE'].agg('sum')

daily_ambient_temp_2.sort_values(ascending=False).plot.bar(figsize=(17,5), legend=True,color='cadetblue')
plt.title('AMBIENT_TEMPERATURE_02')
plt.show()
```




```
plt.figure(figsize=(16,16))

date = "2020-05-25"

plt.subplot(411)
sns.lineplot(x="DATE_TIME", y="DC_POWER", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="DC_Power_Best", color='green')
plt.title("DC Power Generation: {}".format(date))

plt.subplot(412)
sns.lineplot(x="DATE_TIME", y="IRRADIATION", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="Irradiation_Best", color='g
plt.title("Irradiation: {}".format(date))

plt.subplot(413)
sns.lineplot(x="DATE_TIME", y="AMBIENT_TEMPERATURE", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="Ambient_Temperature
sns.lineplot(x="DATE_TIME", y="MODULE_TEMPERATURE", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="Module_Temperature_E
```

```
plt.title("Module Temperature & Ambient Temperature: {}".format(date))

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



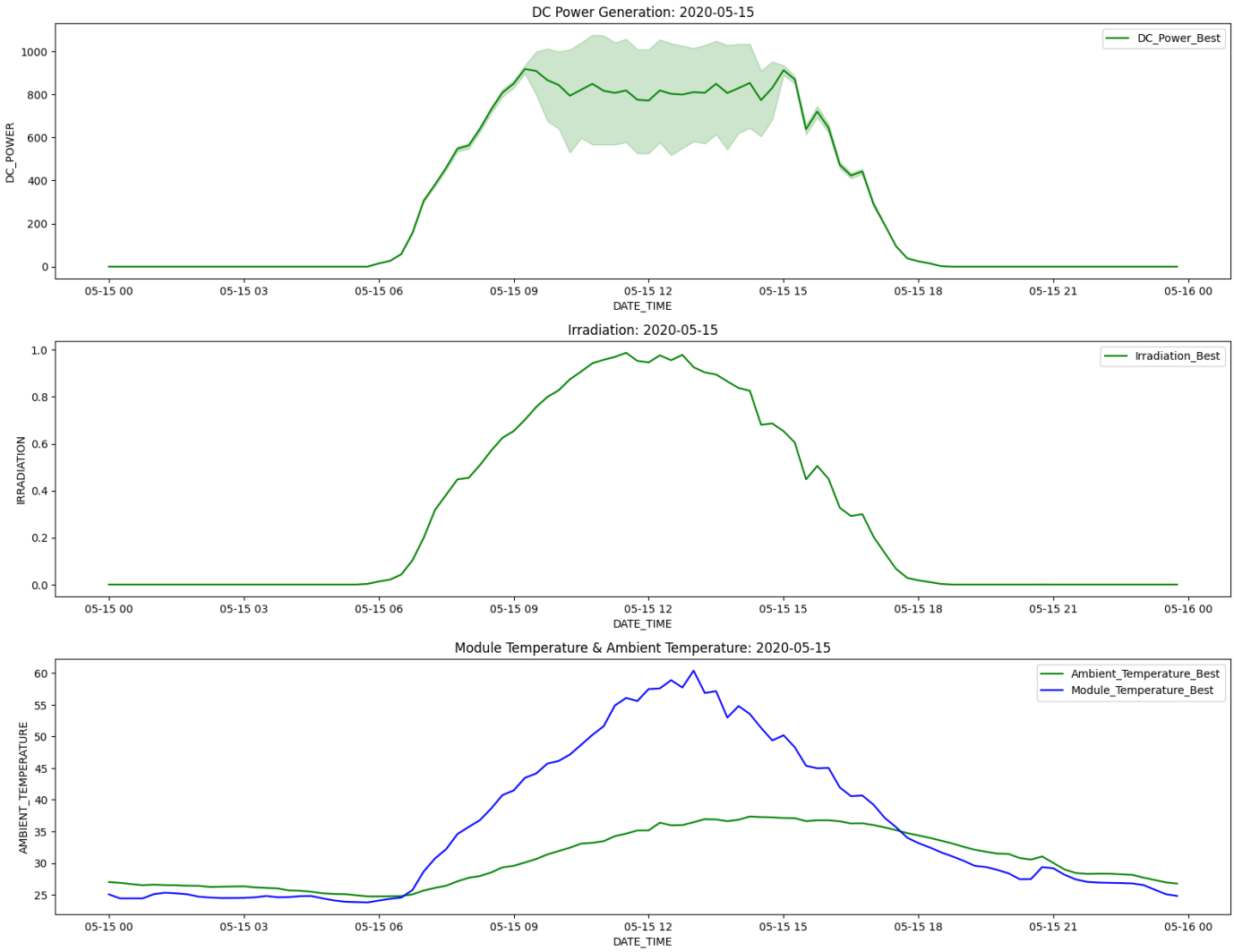

```
plt.subplot(411)
sns.lineplot(data=df_solar_plant1[df_solar_plant1["DATE_STRING"].isin(date)], x="DATE_TIME", y="DC_POWER", label="DC_Power_Best",color='greer
plt.title("DC Power Generation: {}".format(date[0]))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-54-c0251a98cc4b> in <cell line: 2>()
      1 plt.subplot(411)
----> 2 sns.lineplot(data=df_solar_plant1[df_solar_plant1["DATE_STRING"].isin(date)], x="DATE_TIME", y="DC_POWER",
label="DC_Power_Best",color='green')
```

```
date = "2020-05-18"
plt.figure(figsize=(16,16))

plt.subplot(411)
sns.lineplot(x="DATE_TIME", y="DC_POWER", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="DC_Power_Worst", color='red')
plt.title("DC Power Generation: {}".format(date))

plt.subplot(412)
sns.lineplot(x="DATE_TIME", y="IRRADIATION", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="Irradiation_Worst", color='
plt.title("Irradiation: {}".format(date))

plt.subplot(413)
sns.lineplot(x="DATE_TIME", y="AMBIENT_TEMPERATURE", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="Ambient_Temperature
sns.lineplot(x="DATE_TIME", y="MODULE_TEMPERATURE", data=df_solar_plant1[df_solar_plant1["DATE_STRING"] == date], label="Module_Temperature_W
plt.title("Module Temperature & Ambient Temperature: {}".format(date))

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

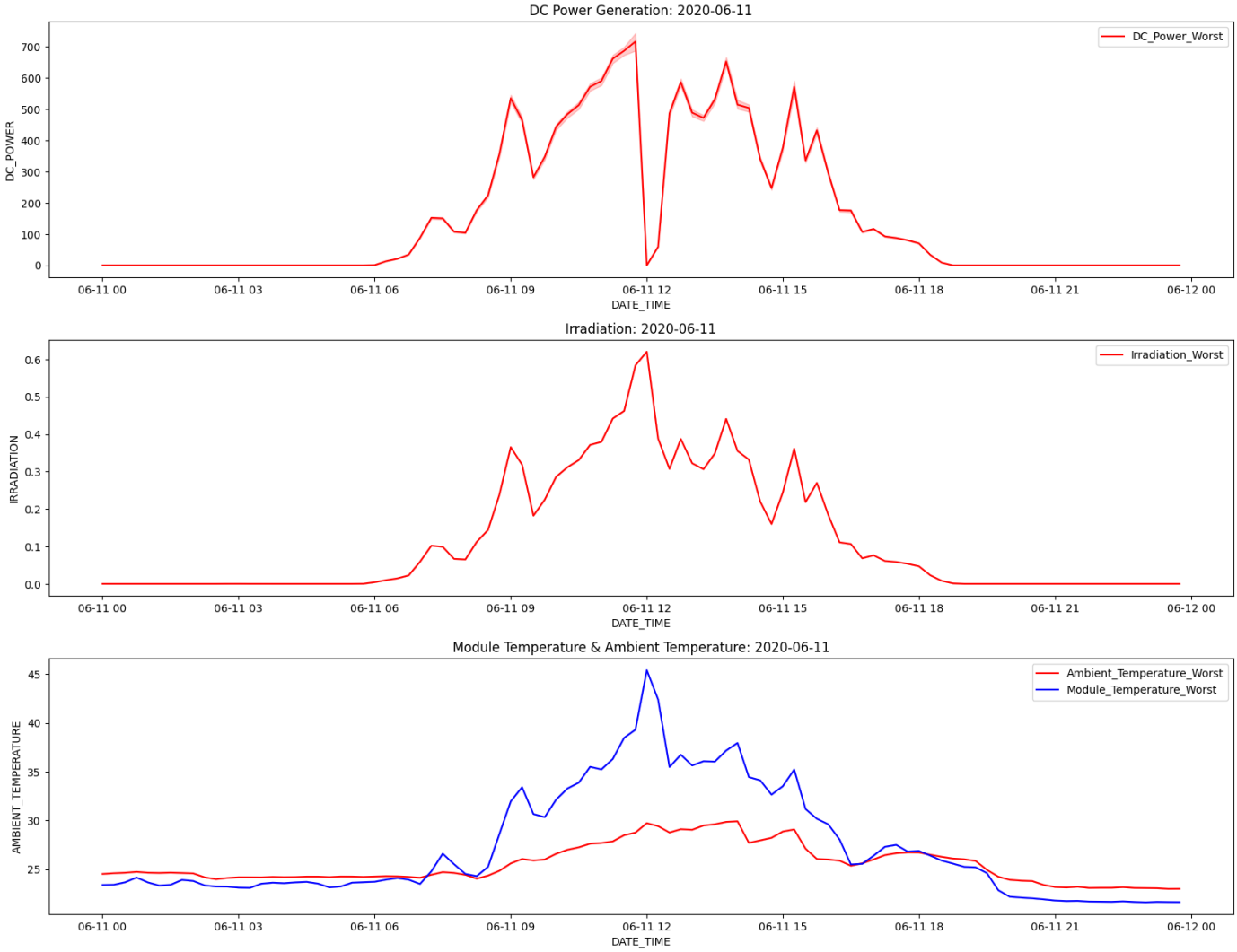

```
plt.figure(figsize=(16,16))

date = "2020-05-15"
```

```
plt.subplot(411)
sns.lineplot(x="DATE_TIME", y="DC_POWER", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="DC_Power_Best", color='green')
plt.title("DC Power Generation: {}".format(date))

plt.subplot(412)
sns.lineplot(x="DATE_TIME", y="IRRADIATION", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="Irradiation_Best", color='g
plt.title("Irradiation: {}".format(date))

plt.subplot(413)
sns.lineplot(x="DATE_TIME", y="AMBIENT_TEMPERATURE", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="Ambient_Temperature
sns.lineplot(x="DATE_TIME", y="MODULE_TEMPERATURE", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="Module_Temperature_B
plt.title("Module Temperature & Ambient Temperature: {}".format(date))

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

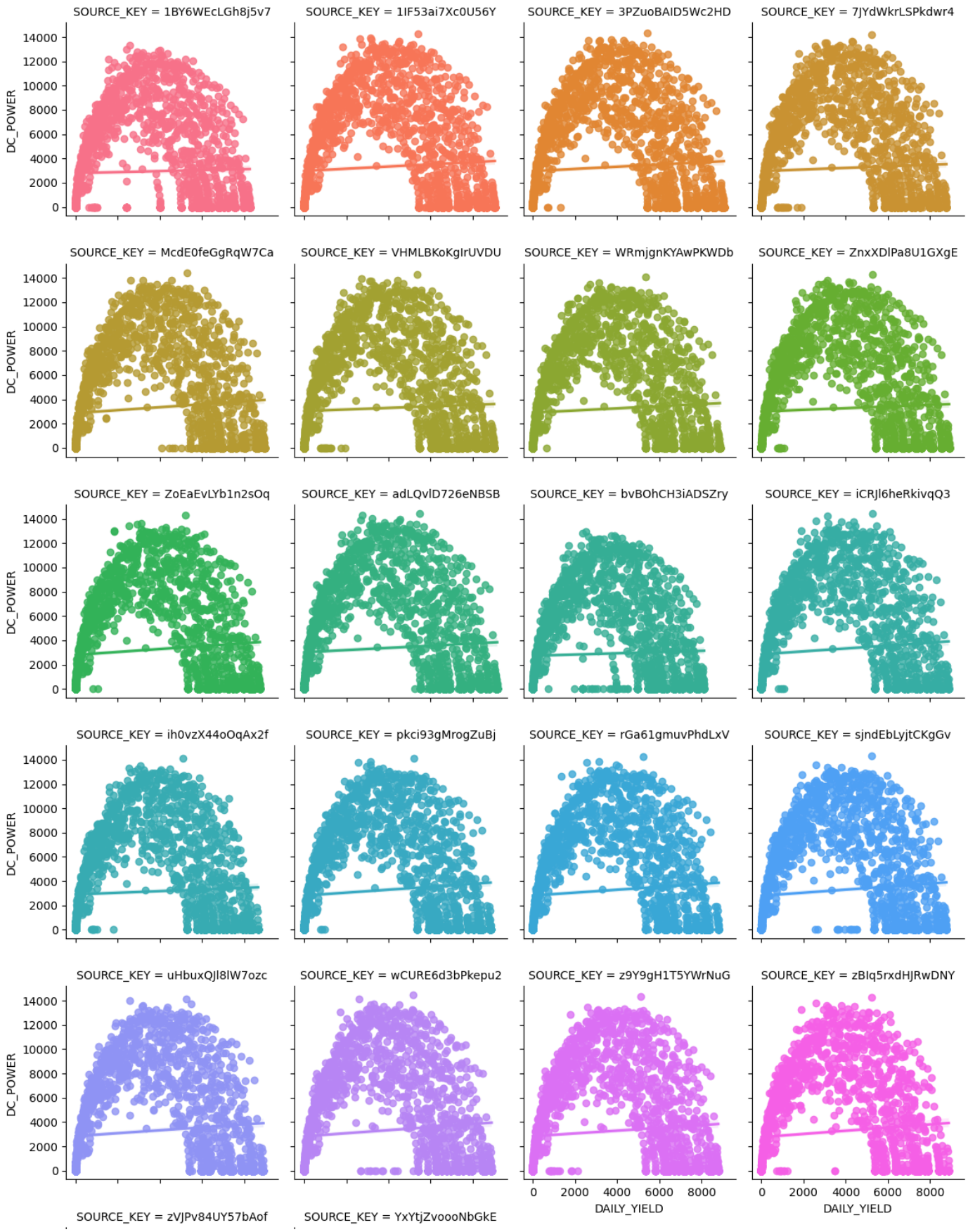



```
date = "2020-06-11"
plt.figure(figsize=(16,16))

plt.subplot(411)
sns.lineplot(x="DATE_TIME", y="DC_POWER", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="DC_Power_Worst", color='red')
plt.title("DC Power Generation: {}".format(date))

plt.subplot(412)
sns.lineplot(x="DATE_TIME", y="IRRADIATION", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="Irradiation_Worst", color='
plt.title("Irradiation: {}".format(date))
```

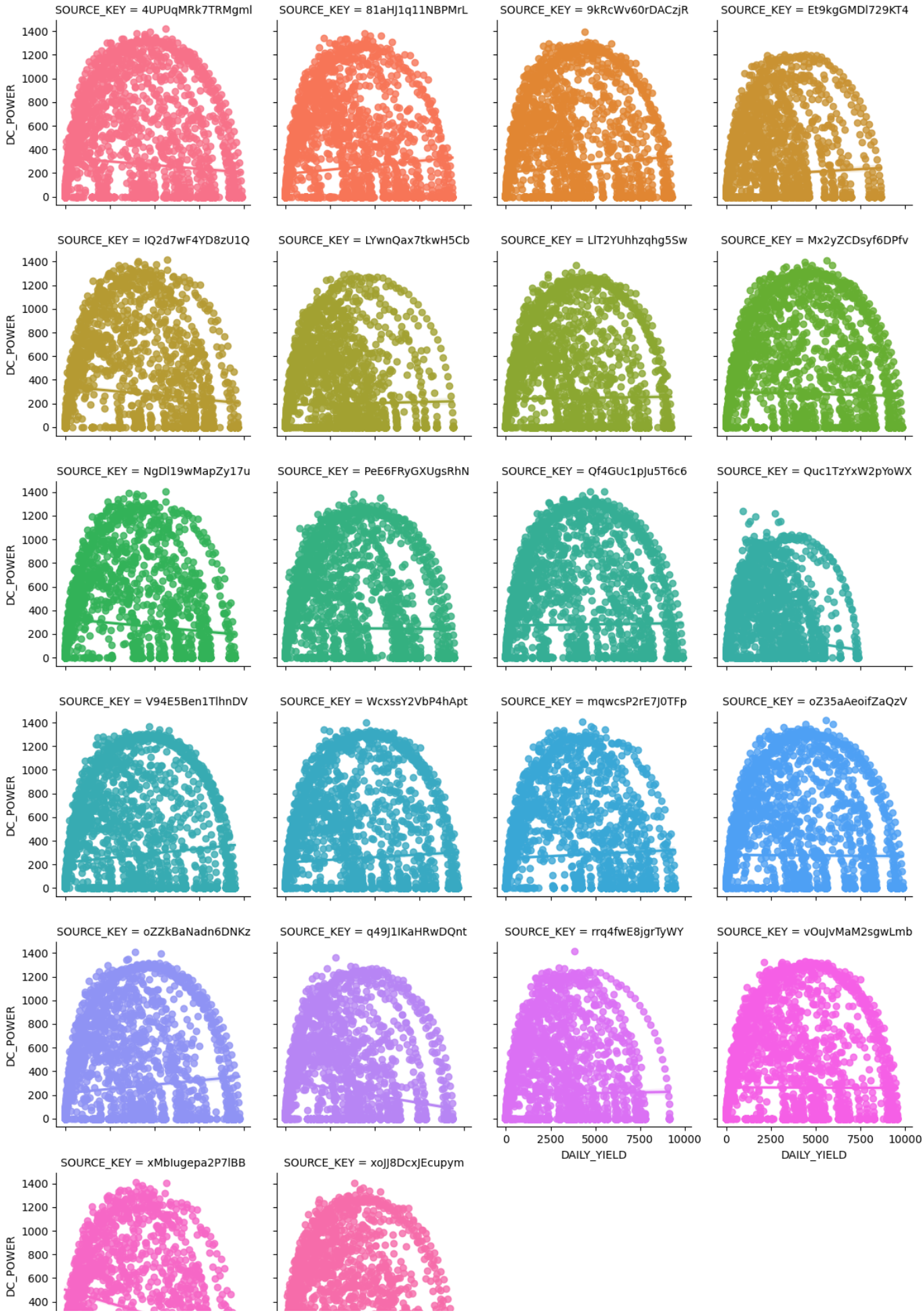
```
plt.subplot(413)
sns.lineplot(x="DATE_TIME", y="AMBIENT_TEMPERATURE", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="Ambient_Temperature
sns.lineplot(x="DATE_TIME", y="MODULE_TEMPERATURE", data=df_solar_plant2[df_solar_plant2["DATE_STRING"] == date], label="Module_Temperature_W
plt.title("Module Temperature & Ambient Temperature: {}".format(date))

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




```
sns.lmplot(y="DC_POWER",x="DAILY_YIELD",hue="SOURCE_KEY",col="SOURCE_KEY",height=3,col_wrap=4,data=df_solar_plant1,fit_reg=True);
```




```
sns.lmplot(y="DC_POWER",x="DAILY_YIELD",hue="SOURCE_KEY",col="SOURCE_KEY",height=3,col_wrap=4,data=df_solar_plant2,fit_reg=True);
```


SOURCE_KEY = 4UPUqMRk7TRMgml
SOURCE_KEY = 81aHJ1q11NBPMrL
SOURCE_KEY = 9kRcWv60rDACzjR
SOURCE_KEY = Et9kgGMDl729KT4
SOURCE_KEY = IQ2d7wF4YD8zU1Q
SOURCE_KEY = LYwnQax7tkwH5Cb
SOURCE_KEY = LlT2YUhhzqhg5Sw
SOURCE_KEY = Mx2yZCDsyf6DPfv
SOURCE_KEY = NgDl19wMapZy17u
SOURCE_KEY = PeE6FRyGXUgsRhN
SOURCE_KEY = Qf4GUc1pJu5T6c6
SOURCE_KEY = Quc1TzYxW2pYoWX
SOURCE_KEY = V94E5Ben1TlhnDV
SOURCE_KEY = WcxssY2VbP4hApt
SOURCE_KEY = mqwcsP2rE7J0TFp
SOURCE_KEY = oZ35aAeoifZaQzV
SOURCE_KEY = oZZkBaNadn6DNKz
SOURCE_KEY = q49J1IKaHRwDQnt
SOURCE_KEY = rrq4fwE8jgrTyWY
SOURCE_KEY = vOuJvMaM2sgwLmb
SOURCE_KEY = xMblugepa2P7lBB
SOURCE_KEY = xoJJ8DcxJEcupym
DC_POWER
DAILY_YIELD
0
200
400
600
800
1000
1200
1400
2500
5000
7500
10000

```
solar_dc_power_1 = df_solar_plant1[df_solar_plant1['DC_POWER'] > 0]['DC_POWER'].values
solar_ac_power_1 = df_solar_plant1[df_solar_plant1['AC_POWER'] > 0]['AC_POWER'].values

solar_plant_eff = (np.max(solar_ac_power_1)/np.max(solar_dc_power_1 ))*100
print(f"Power ratio AC/DC (Efficiency) of Solar Power Plant 001:  {solar_plant_eff:0.3f} %")
```

```
Power ratio AC/DC (Efficiency) of Solar Power Plant 001:  9.750 %
```

```
np.max(solar_dc_power_1)
```

```
14471.125
```

```
np.max(solar_ac_power_1)
```

```
1410.95
```

```
solar_dc_power_2 = df_solar_plant2[df_solar_plant2['DC_POWER'] > 0]['DC_POWER'].values
solar_ac_power_2 = df_solar_plant2[df_solar_plant2['AC_POWER'] > 0]['AC_POWER'].values

solar_plant_eff = (np.max(solar_ac_power_2)/np.max(solar_dc_power_2 ))*100
print(f"Power ratio AC/DC (Efficiency) of Solar Power Plant 002:  {solar_plant_eff:0.3f} %")
```

```
Power ratio AC/DC (Efficiency) of Solar Power Plant 002:  97.501 %
```

```
AC_list_1=[]
for i in df_solar_plant1['AC_POWER']:
    if i>0:
        AC_list_1.append(i)
AC_list_1
#AC_list.sort()
#AC_list.reverse()
len(AC_list_1)
```

```
24620
```

```
AC_list_2=[]
for i in df_solar_plant2['AC_POWER']:
    if i>0:
        AC_list_2.append(i)
AC_list_2
#AC_list.sort()
#AC_list.reverse()
len(AC_list_2)
```

```
32036
```

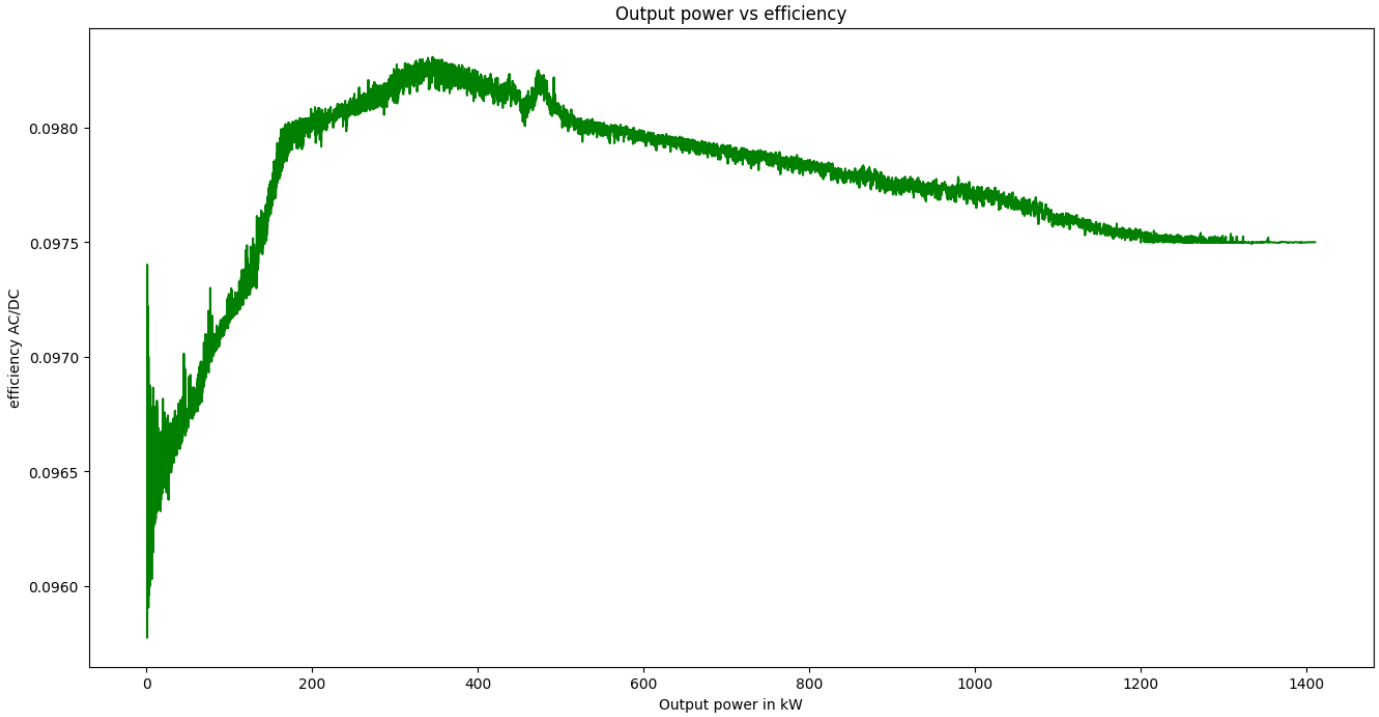
```
#Here we take all nonzero DC values and plot them on histogram
DC_list_1=[]
for i in df_solar_plant1['DC_POWER']:
    if i>0:
        DC_list_1.append(i)
DC_list_1
DC_list_1.sort()
DC_list_1.reverse()
len(DC_list_1)
```

```
24620
```

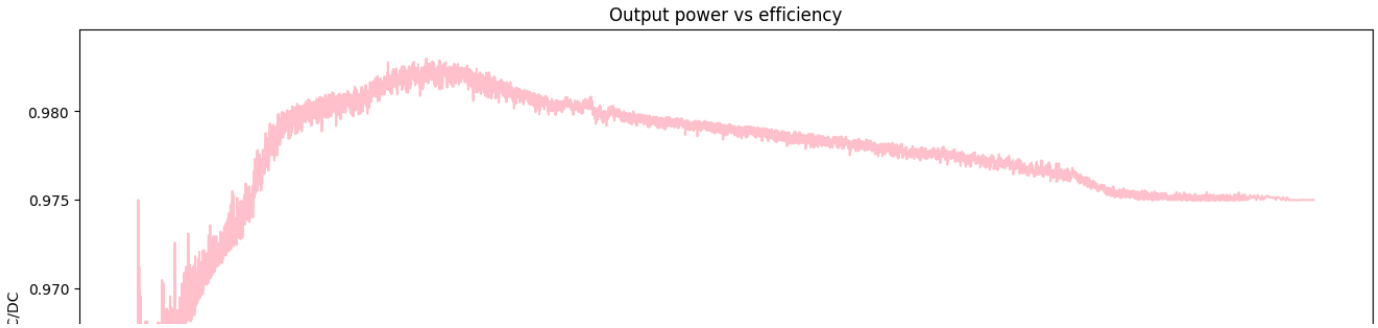
```
#Here we take all nonzero DC values and plot them on histogram
DC_list_2=[]
for i in df_solar_plant2['DC_POWER']:
    if i>0:
        DC_list_2.append(i)
DC_list_2
DC_list_2.sort()
DC_list_2.reverse()
len(DC_list_2)
```

```
32036
```

```
plt.figure(figsize=(16,8))
AC_list_1.sort()
DC_list_1.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_1,DC_list_1)]

plt.plot(AC_list_1,eff,color='green')
plt.xlabel('Output power in kW')
plt.ylabel('efficiency AC/DC')
plt.title('Output power vs efficiency');
```




```
plt.figure(figsize=(16,8))
AC_list_2.sort()
DC_list_2.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_2,DC_list_2)]

plt.plot(AC_list_2,eff,color='pink')
plt.xlabel('Output power in kW')
plt.ylabel('efficiency AC/DC')
plt.title('Output power vs efficiency');
```



Plant 001 data set Train and analytics

```
df2 = df_solar_plant1.copy()
X = df2[['DAILY_YIELD','TOTAL_YIELD','AMBIENT_TEMPERATURE','MODULE_TEMPERATURE','IRRADIATION','DC_POWER']]
y = df2['AC_POWER']
```

```
X.head()
```

	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION	DC_POWER
0	0.0	6259559.0	25.184316	22.857507	0.0	0.0
1	0.0	6183645.0	25.184316	22.857507	0.0	0.0
2	0.0	6987759.0	25.184316	22.857507	0.0	0.0
3	0.0	7602960.0	25.184316	22.857507	0.0	0.0
4	0.0	7158964.0	25.184316	22.857507	0.0	0.0

```
y.head()
```

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: AC_POWER, dtype: float64
```

```
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)
```

```
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:4.4f}%')
```

```
LR Model score = 99.9994%
```

```
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 %
```

```
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)
```