Name: Ananya Godse SAP ID: 60009220161

Importing the necessary libraries

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

Importing the power generation data and weather sensor data for both plants

```
In [2]: plant1_generation = pd.read_csv(r"Solar Power Generation Data\Plant_1_Generation_Data.csv")
        print("PLANT 1 GENERATION DATA")
        display(plant1_generation)

        plant1_sensor = pd.read_csv(r"Solar Power Generation Data\Plant_1_Weather_Sensor_Data.csv")
        print("PLANT 1 WEATHER SENSOR DATA")
        display(plant1_sensor)

        plant2_generation = pd.read_csv(r"Solar Power Generation Data\Plant_2_Generation_Data.csv")
        print("PLANT 2 GENERATION DATA")
        display(plant2_generation)

        plant2_sensor = pd.read_csv(r"Solar Power Generation Data\Plant_2_Weather_Sensor_Data.csv")
        print("PLANT 2 WEATHER SENSOR DATA")
        display(plant1_sensor)
```

```
PLANT 1 GENERATION DATA
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0
1	15-05-2020 00:00	4135001	1IF53ai7Xc0U56Y	0.0	0.0	0.000	6183645.0
2	15-05-2020 00:00	4135001	3PZuoBAID5Wc2HD	0.0	0.0	0.000	6987759.0
3	15-05-2020 00:00	4135001	7JYdWkrLSPkdwr4	0.0	0.0	0.000	7602960.0
4	15-05-2020 00:00	4135001	McdE0feGgRqW7Ca	0.0	0.0	0.000	7158964.0
...	...	...	...	...	...	...	...
68773	17-06-2020 23:45	4135001	uHbuxQJl8lW7ozc	0.0	0.0	5967.000	7287002.0
68774	17-06-2020 23:45	4135001	wCURE6d3bPkepu2	0.0	0.0	5147.625	7028601.0
68775	17-06-2020 23:45	4135001	z9Y9gH1T5YWrNuG	0.0	0.0	5819.000	7251204.0
68776	17-06-2020 23:45	4135001	zBIq5rxdHJRwDNY	0.0	0.0	5817.000	6583369.0
68777	17-06-2020 23:45	4135001	zVJPv84UY57bAof	0.0	0.0	5910.000	7363272.0

68778 rows × 7 columns

```
PLANT 1 WEATHER SENSOR DATA
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	4135001	HmiyD2TTLFNqkNe	25.184316	22.857507	0.0
1	2020-05-15 00:15:00	4135001	HmiyD2TTLFNqkNe	25.084589	22.761668	0.0
2	2020-05-15 00:30:00	4135001	HmiyD2TTLFNqkNe	24.935753	22.592306	0.0
3	2020-05-15 00:45:00	4135001	HmiyD2TTLFNqkNe	24.846130	22.360852	0.0
4	2020-05-15 01:00:00	4135001	HmiyD2TTLFNqkNe	24.621525	22.165423	0.0
...	...	...	...	...	...	...
3177	2020-06-17 22:45:00	4135001	HmiyD2TTLFNqkNe	22.150570	21.480377	0.0
3178	2020-06-17 23:00:00	4135001	HmiyD2TTLFNqkNe	22.129816	21.389024	0.0
3179	2020-06-17 23:15:00	4135001	HmiyD2TTLFNqkNe	22.008275	20.709211	0.0
3180	2020-06-17 23:30:00	4135001	HmiyD2TTLFNqkNe	21.969495	20.734963	0.0
3181	2020-06-17 23:45:00	4135001	HmiyD2TTLFNqkNe	21.909288	20.427972	0.0

3182 rows × 6 columns

```
PLANT 2 GENERATION DATA
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	2020-05-15 00:00:00	4136001	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06
1	2020-05-15 00:00:00	4136001	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09
2	2020-05-15 00:00:00	4136001	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09
3	2020-05-15 00:00:00	4136001	Et9kgGMDl729KT4	0.0	0.0	269.933333	1.704250e+06
4	2020-05-15 00:00:00	4136001	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07
...	...	...	...	...	...	...	...
67693	2020-06-17 23:45:00	4136001	q49J1IKaHRwDQnt	0.0	0.0	4157.000000	5.207580e+05
67694	2020-06-17 23:45:00	4136001	rrq4fwE8jgrTyWY	0.0	0.0	3931.000000	1.211314e+08
67695	2020-06-17 23:45:00	4136001	vOuJvMaM2sgwLmb	0.0	0.0	4322.000000	2.427691e+06
67696	2020-06-17 23:45:00	4136001	xMbIugepa2P7lBB	0.0	0.0	4218.000000	1.068964e+08
67697	2020-06-17 23:45:00	4136001	xoJJ8DcxJEcupym	0.0	0.0	4316.000000	2.093357e+08

67698 rows × 7 columns

```
PLANT 2 WEATHER SENSOR DATA
```

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	4135001	HmiyD2TTLFNqkNe	25.184316	22.857507	0.0
1	2020-05-15 00:15:00	4135001	HmiyD2TTLFNqkNe	25.084589	22.761668	0.0
2	2020-05-15 00:30:00	4135001	HmiyD2TTLFNqkNe	24.935753	22.592306	0.0
3	2020-05-15 00:45:00	4135001	HmiyD2TTLFNqkNe	24.846130	22.360852	0.0
4	2020-05-15 01:00:00	4135001	HmiyD2TTLFNqkNe	24.621525	22.165423	0.0
...	...	...	...	...	...	...
3177	2020-06-17 22:45:00	4135001	HmiyD2TTLFNqkNe	22.150570	21.480377	0.0
3178	2020-06-17 23:00:00	4135001	HmiyD2TTLFNqkNe	22.129816	21.389024	0.0
3179	2020-06-17 23:15:00	4135001	HmiyD2TTLFNqkNe	22.008275	20.709211	0.0
3180	2020-06-17 23:30:00	4135001	HmiyD2TTLFNqkNe	21.969495	20.734963	0.0
3181	2020-06-17 23:45:00	4135001	HmiyD2TTLFNqkNe	21.909288	20.427972	0.0

3182 rows × 6 columns

In [3]:

```
plant1_generation.head()
```

Out[3]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0
1	15-05-2020 00:00	4135001	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0
2	15-05-2020 00:00	4135001	3PZuoBAID5Wc2HD	0.0	0.0	0.0	6987759.0
3	15-05-2020 00:00	4135001	7JYdWkrLSPkdwr4	0.0	0.0	0.0	7602960.0
4	15-05-2020 00:00	4135001	McdE0feGgRqW7Ca	0.0	0.0	0.0	7158964.0

In [4]:

```
plant1_sensor.head()
```

Out[4]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	4135001	HmiyD2TTLFNqkNe	25.184316	22.857507	0.0
1	2020-05-15 00:15:00	4135001	HmiyD2TTLFNqkNe	25.084589	22.761668	0.0
2	2020-05-15 00:30:00	4135001	HmiyD2TTLFNqkNe	24.935753	22.592306	0.0
3	2020-05-15 00:45:00	4135001	HmiyD2TTLFNqkNe	24.846130	22.360852	0.0
4	2020-05-15 01:00:00	4135001	HmiyD2TTLFNqkNe	24.621525	22.165423	0.0

In [5]:

```
plant2_generation.head()
```

Out[5]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	2020-05-15 00:00:00	4136001	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06
1	2020-05-15 00:00:00	4136001	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09
2	2020-05-15 00:00:00	4136001	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09
3	2020-05-15 00:00:00	4136001	Et9kgGMDl729KT4	0.0	0.0	269.933333	1.704250e+06
4	2020-05-15 00:00:00	4136001	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07

In [6]:
```
plant2_sensor.head()
```

Out[6]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	4136001	iq8k7ZNt4Mwm3w0	27.004764	25.060789	0.0
1	2020-05-15 00:15:00	4136001	iq8k7ZNt4Mwm3w0	26.880811	24.421869	0.0
2	2020-05-15 00:30:00	4136001	iq8k7ZNt4Mwm3w0	26.682055	24.427290	0.0
3	2020-05-15 00:45:00	4136001	iq8k7ZNt4Mwm3w0	26.500589	24.420678	0.0
4	2020-05-15 01:00:00	4136001	iq8k7ZNt4Mwm3w0	26.596148	25.088210	0.0

In [7]:
```
plant1_generation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68778 entries, 0 to 68777
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   DATE_TIME    68778 non-null  object 
 1   PLANT_ID     68778 non-null  int64  
 2   SOURCE_KEY   68778 non-null  object 
 3   DC_POWER     68778 non-null  float64
 4   AC_POWER     68778 non-null  float64
 5   DAILY_YIELD  68778 non-null  float64
 6   TOTAL_YIELD  68778 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 3.7+ MB
```

In [8]:
```
plant1_sensor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3182 entries, 0 to 3181
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   DATE_TIME            3182 non-null   object 
 1   PLANT_ID             3182 non-null   int64  
 2   SOURCE_KEY           3182 non-null   object 
 3   AMBIENT_TEMPERATURE  3182 non-null   float64
 4   MODULE_TEMPERATURE   3182 non-null   float64
 5   IRRADIATION          3182 non-null   float64
dtypes: float64(3), int64(1), object(2)
memory usage: 149.3+ KB
```

In [9]:
```
plant2_generation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67698 entries, 0 to 67697
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   DATE_TIME    67698 non-null  object 
 1   PLANT_ID     67698 non-null  int64  
 2   SOURCE_KEY   67698 non-null  object 
 3   DC_POWER     67698 non-null  float64
 4   AC_POWER     67698 non-null  float64
 5   DAILY_YIELD  67698 non-null  float64
 6   TOTAL_YIELD  67698 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 3.6+ MB
```

In [10]:

```
plant2_sensor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3259 entries, 0 to 3258
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   DATE_TIME            3259 non-null   object 
 1   PLANT_ID             3259 non-null   int64  
 2   SOURCE_KEY           3259 non-null   object 
 3   AMBIENT_TEMPERATURE  3259 non-null   float64
 4   MODULE_TEMPERATURE   3259 non-null   float64
 5   IRRADIATION          3259 non-null   float64
dtypes: float64(3), int64(1), object(2)
memory usage: 152.9+ KB
```

Observations:

1. DATE_TIME column data type needs to converted to Date time for all the datasets.
2. We know from the data description that the SOURCE_KEY column in the generation datasets is the Inverter ID and the Sensor Panel ID in the Weather Sensor Datasets. We'll rename the columns.

In [11]:

```
plant1_generation["PLANT_ID"].value_counts()
```

Out[11]:

```
PLANT_ID
4135001    68778
Name: count, dtype: int64
```

Observations:

As we know from data description and as proven above, all the records from the PLANT 1 GENERATION DATA belong to Plant 1. Since this doesn't provide any actionable insight, we'll drop the column.

In [12]:

```
plant1_generation["SOURCE_KEY"].value_counts()
```

Out[12]:

```
SOURCE_KEY
bvBOhCH3iADSZry    3155
1BY6WEcLGh8j5v7    3154
7JYdWkrLSPkdwr4    3133
VHMLBKoKgIrUVDU    3133
ZnxXDlPa8U1GXgE    3130
ih0vzX44oOqAx2f    3130
z9Y9gH1T5YWrNuG    3126
wCURE6d3bPkepu2    3126
uHbuxQJl8lW7ozc    3125
pkci93gMrogZuBj    3125
iCRJl6heRkivqQ3    3125
rGa61gmuvPhdLxV    3124
sjndEbLyjtCKgGv    3124
McdE0feGgRqW7Ca    3124
zVJPv84UY57bAof    3124
ZoEaEvLYb1n2sOq    3123
1IF53ai7Xc0U56Y    3119
adLQvlD726eNBSB    3119
zBIq5rxdHJRwDNY    3119
WRmjgnKYAwPKWDb    3118
3PZuoBAID5Wc2HD    3118
YxYtjZvoooNbGkE    3104
Name: count, dtype: int64
```

As we know from the data description, the SOURCE_KEY column in the PLANT 1 GENERATION DATA SET has the INVERTER ID

In [13]:
```
print(f"No. of Inverters in Plant 1: {len(plant1_generation['SOURCE_KEY'].value_counts())}")
```

```
No. of Inverters in Plant 1: 22
```

In [14]:
```
plant1_sensor["PLANT_ID"].value_counts()
```

Out[14]:
```
PLANT_ID
4135001    3182
Name: count, dtype: int64
```

All records in PLANT 1 WEATHER SENSOR DATA belong to Plant 1. Since this doesn't provide any actionable insight, we'll be dropping this column.

In [15]:
```
plant1_sensor["SOURCE_KEY"].value_counts()
```

Out[15]:
```
SOURCE_KEY
HmiyD2TTLFNqkNe    3182
Name: count, dtype: int64
```

As we know from the data description, the SOURCE_KEY column in the PLANT 1 WEATHER SENSOR DATA SET has the SENSOR PANEL ID and there is only one Sensor Panel in Plant 1. So since it doesn't provide any insight we can drop the column.

In [16]:
```
plant2_generation["PLANT_ID"].value_counts()
```

Out[16]:
```
PLANT_ID
4136001    67698
Name: count, dtype: int64
```

Observations:

As we know from data description and as proven above, all the records from the PLANT 2 GENERATION DATA belong to Plant 2. Since this doesn't provide any actionable insight, we'll be dropping the column.

In [17]:
```
plant2_generation["SOURCE_KEY"].value_counts()
```

Out[17]:
```
SOURCE_KEY
xoJJ8DcxJEcupym    3259
WcxssY2VbP4hApt    3259
9kRcWv60rDACzjR    3259
vOuJvMaM2sgwLmb    3259
rrq4fwE8jgrTyWY    3259
LYwnQax7tkwH5Cb    3259
LlT2YUhhzqhg5Sw    3259
q49J1IKaHRwDQnt    3259
oZZkBaNadn6DNKz    3259
PeE6FRyGXUgsRhN    3259
81aHJ1q11NBPMrL    3259
V94E5Ben1TlhnDV    3259
oZ35aAeoifZaQzV    3195
4UPUqMRk7TRMgml    3195
Qf4GUc1pJu5T6c6    3195
Mx2yZCDsyf6DPfv    3195
Et9kgGMDl729KT4    3195
Quc1TzYxW2pYoWX    3195
mqwcsP2rE7J0TFp    2355
NgDl19wMapZy17u    2355
IQ2d7wF4YD8zU1Q    2355
xMbIugepa2P7lBB    2355
Name: count, dtype: int64
```

As we know from the data description, the SOURCE_KEY column in the PLANT 2 GENERATION DATA SET has the INVERTER ID

In [18]:
```
print(f"No. of Inverters in Plant 2: {len(plant2_generation['SOURCE_KEY'].value_counts())}")
```

```
No. of Inverters in Plant 2: 22
```

In [19]:
```
plant2_sensor["PLANT_ID"].value_counts()
```

Out[19]:

```
PLANT_ID
4136001    3259
Name: count, dtype: int64
```

All records in PLANT 2 WEATHER SENSOR DATA belong to Plant 2. Since this doesn't provide any actionable insight, we'll be dropping this column.

In [20]:

```
plant2_sensor["SOURCE_KEY"].value_counts()
```

Out[20]:

```
SOURCE_KEY
iq8k7ZNt4Mwm3w0    3259
Name: count, dtype: int64
```

As we know from the data description, the SOURCE_KEY column in the PLANT 2 WEATHER SENSOR DATA SET has the SENSOR PANEL ID and there is only one Sensor Panel in Plant 2. Since it doesn't provide any insight we can drop the column.

Renaming & Dropping Columns:

In [21]:

```
plant1_generation.rename(columns={"SOURCE_KEY":"INVERTER_ID"}, inplace=True)
plant1_generation
```

Out[21]:

	DATE_TIME	PLANT_ID	INVERTER_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0
1	15-05-2020 00:00	4135001	1IF53ai7Xc0U56Y	0.0	0.0	0.000	6183645.0
2	15-05-2020 00:00	4135001	3PZuoBAID5Wc2HD	0.0	0.0	0.000	6987759.0
3	15-05-2020 00:00	4135001	7JYdWkrLSPkdwr4	0.0	0.0	0.000	7602960.0
4	15-05-2020 00:00	4135001	McdE0feGgRqW7Ca	0.0	0.0	0.000	7158964.0
...	...	...	...	...	...	...	...
68773	17-06-2020 23:45	4135001	uHbuxQJl8lW7ozc	0.0	0.0	5967.000	7287002.0
68774	17-06-2020 23:45	4135001	wCURE6d3bPkepu2	0.0	0.0	5147.625	7028601.0
68775	17-06-2020 23:45	4135001	z9Y9gH1T5YWrNuG	0.0	0.0	5819.000	7251204.0
68776	17-06-2020 23:45	4135001	zBIq5rxdHJRwDNY	0.0	0.0	5817.000	6583369.0
68777	17-06-2020 23:45	4135001	zVJPv84UY57bAof	0.0	0.0	5910.000	7363272.0

68778 rows × 7 columns

In [22]:

```
plant1_generation.drop("PLANT_ID", axis=1, inplace=True)
plant1_generation
```

Out[22]:

	DATE_TIME	INVERTER_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0
1	15-05-2020 00:00	1IF53ai7Xc0U56Y	0.0	0.0	0.000	6183645.0
2	15-05-2020 00:00	3PZuoBAID5Wc2HD	0.0	0.0	0.000	6987759.0
3	15-05-2020 00:00	7JYdWkrLSPkdwr4	0.0	0.0	0.000	7602960.0
4	15-05-2020 00:00	McdE0feGgRqW7Ca	0.0	0.0	0.000	7158964.0
...	...	...	...	...	...	...
68773	17-06-2020 23:45	uHbuxQJl8lW7ozc	0.0	0.0	5967.000	7287002.0
68774	17-06-2020 23:45	wCURE6d3bPkepu2	0.0	0.0	5147.625	7028601.0
68775	17-06-2020 23:45	z9Y9gH1T5YWrNuG	0.0	0.0	5819.000	7251204.0
68776	17-06-2020 23:45	zBIq5rxdHJRwDNY	0.0	0.0	5817.000	6583369.0
68777	17-06-2020 23:45	zVJPv84UY57bAof	0.0	0.0	5910.000	7363272.0

68778 rows × 6 columns

In [23]:
```
plant1_sensor.drop(["SOURCE_KEY", "PLANT_ID"], axis=1, inplace=True)
plant1_sensor
```

Out[23]:

	DATE_TIME	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	25.184316	22.857507	0.0
1	2020-05-15 00:15:00	25.084589	22.761668	0.0
2	2020-05-15 00:30:00	24.935753	22.592306	0.0
3	2020-05-15 00:45:00	24.846130	22.360852	0.0
4	2020-05-15 01:00:00	24.621525	22.165423	0.0
...	...	...	...	...
3177	2020-06-17 22:45:00	22.150570	21.480377	0.0
3178	2020-06-17 23:00:00	22.129816	21.389024	0.0
3179	2020-06-17 23:15:00	22.008275	20.709211	0.0
3180	2020-06-17 23:30:00	21.969495	20.734963	0.0
3181	2020-06-17 23:45:00	21.909288	20.427972	0.0

3182 rows × 4 columns

In [24]:
```
plant2_generation.rename(columns={"SOURCE_KEY":"INVERTER_ID"}, inplace=True)
plant2_generation
```

Out[24]:

	DATE_TIME	PLANT_ID	INVERTER_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	2020-05-15 00:00:00	4136001	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06
1	2020-05-15 00:00:00	4136001	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09
2	2020-05-15 00:00:00	4136001	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09
3	2020-05-15 00:00:00	4136001	Et9kgGMDl729KT4	0.0	0.0	269.933333	1.704250e+06
4	2020-05-15 00:00:00	4136001	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07
...	...	...	...	...	...	...	...
67693	2020-06-17 23:45:00	4136001	q49J1IKaHRwDQnt	0.0	0.0	4157.000000	5.207580e+05
67694	2020-06-17 23:45:00	4136001	rrq4fwE8jgrTyWY	0.0	0.0	3931.000000	1.211314e+08
67695	2020-06-17 23:45:00	4136001	vOuJvMaM2sgwLmb	0.0	0.0	4322.000000	2.427691e+06
67696	2020-06-17 23:45:00	4136001	xMbIugepa2P7lBB	0.0	0.0	4218.000000	1.068964e+08
67697	2020-06-17 23:45:00	4136001	xoJJ8DcxJEcupym	0.0	0.0	4316.000000	2.093357e+08

67698 rows × 7 columns

In [25]:
```
plant2_generation.drop("PLANT_ID", axis=1, inplace=True)
plant2_generation
```

Out[25]:

	DATE_TIME	INVERTER_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06
1	2020-05-15 00:00:00	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09
2	2020-05-15 00:00:00	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09
3	2020-05-15 00:00:00	Et9kgGMDl729KT4	0.0	0.0	269.933333	1.704250e+06
4	2020-05-15 00:00:00	IQ2d7wF4YD8zU1Q	0.0	0.0	3177.000000	1.994153e+07
...	...	...	...	...	...	...
67693	2020-06-17 23:45:00	q49J1IKaHRwDQnt	0.0	0.0	4157.000000	5.207580e+05
67694	2020-06-17 23:45:00	rrq4fwE8jgrTyWY	0.0	0.0	3931.000000	1.211314e+08
67695	2020-06-17 23:45:00	vOuJvMaM2sgwLmb	0.0	0.0	4322.000000	2.427691e+06
67696	2020-06-17 23:45:00	xMbIugepa2P7lBB	0.0	0.0	4218.000000	1.068964e+08
67697	2020-06-17 23:45:00	xoJJ8DcxJEcupym	0.0	0.0	4316.000000	2.093357e+08

67698 rows × 6 columns

In [26]:

```
plant2_sensor.drop(["SOURCE_KEY", "PLANT_ID"], axis=1, inplace=True)
plant2_sensor
```

Out[26]:

	DATE_TIME	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	27.004764	25.060789	0.0
1	2020-05-15 00:15:00	26.880811	24.421869	0.0
2	2020-05-15 00:30:00	26.682055	24.427290	0.0
3	2020-05-15 00:45:00	26.500589	24.420678	0.0
4	2020-05-15 01:00:00	26.596148	25.088210	0.0
...	...	...	...	...
3254	2020-06-17 22:45:00	23.511703	22.856201	0.0
3255	2020-06-17 23:00:00	23.482282	22.744190	0.0
3256	2020-06-17 23:15:00	23.354743	22.492245	0.0
3257	2020-06-17 23:30:00	23.291048	22.373909	0.0
3258	2020-06-17 23:45:00	23.202871	22.535908	0.0

3259 rows × 4 columns

Handling Missing & Duplicate Values

In [27]:

```
plant1_generation.isnull().sum()
```

Out[27]:

```
DATE_TIME      0
INVERTER_ID    0
DC_POWER       0
AC_POWER       0
DAILY_YIELD    0
TOTAL_YIELD    0
dtype: int64
```

In [28]:

```
plant1_sensor.isnull().sum()
```

Out[28]:

```
DATE_TIME              0
AMBIENT_TEMPERATURE    0
MODULE_TEMPERATURE     0
IRRADIATION            0
dtype: int64
```

In [29]:

```
plant2_generation.isnull().sum()
```

Out[29]:

```
DATE_TIME      0
INVERTER_ID    0
DC_POWER       0
AC_POWER       0
DAILY_YIELD    0
TOTAL_YIELD    0
dtype: int64
```

In [30]:

```
plant2_sensor.isnull().sum()
```

Out[30]:

```
DATE_TIME              0
AMBIENT_TEMPERATURE    0
MODULE_TEMPERATURE     0
IRRADIATION            0
dtype: int64
```

In [31]:

```
plant1_generation.duplicated().sum()
```

Out[31]:

```
0
```

In [32]:

```
plant1_sensor.duplicated().sum()
```

Out[32]:

```
0
```

In [33]:

```
plant2_generation.duplicated().sum()
```

Out[33]:

```
0
```

In [34]:

```
plant2_sensor.duplicated().sum()
```

Out[34]:

```
0
```

There are no missing values or duplicated values in any of the datasets.

Changing the data type of DATE_TIME to datetime

In [35]:

```
plant1_generation["DATE_TIME"] = pd.to_datetime(plant1_generation["DATE_TIME"], format='%d-%m-%Y %H:
```

In [36]:

```
plant1_generation.dtypes
```

Out[36]:

```
DATE_TIME      datetime64[ns]
INVERTER_ID            object
DC_POWER              float64
AC_POWER              float64
DAILY_YIELD           float64
TOTAL_YIELD           float64
dtype: object
```

In [37]:

```
plant1_sensor["DATE_TIME"] = pd.to_datetime(plant1_generation["DATE_TIME"], format="%Y-%m-%d %H:%M:
```

In [38]:

```
plant1_sensor.dtypes
```

Out[38]:

```
DATE_TIME              datetime64[ns]
AMBIENT_TEMPERATURE           float64
MODULE_TEMPERATURE            float64
IRRADIATION                   float64
dtype: object
```

In [39]:

```
plant2_generation["DATE_TIME"] = pd.to_datetime(plant1_generation["DATE_TIME"], format="%Y-%m-%d %H:
```

In [40]:

```
plant2_generation.dtypes
```

Out[40]:

```
DATE_TIME      datetime64[ns]
INVERTER_ID            object
DC_POWER              float64
AC_POWER              float64
DAILY_YIELD           float64
TOTAL_YIELD           float64
dtype: object
```

In [41]:

```
plant2_sensor["DATE_TIME"] = pd.to_datetime(plant1_generation["DATE_TIME"], format="%Y-%m-%d %H:%M:
```

In [42]:

```
plant2_sensor.dtypes
```

Out[42]:

```
DATE_TIME              datetime64[ns]
AMBIENT_TEMPERATURE           float64
MODULE_TEMPERATURE            float64
IRRADIATION                   float64
dtype: object
```

Summary Statistics

PLANT 1

In [43]:

```
plant1_generation.describe()
```

Out[43]:

	DATE_TIME	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
count	68778	68778.000000	68778.000000	68778.000000	6.877800e+04
mean	2020-06-01 08:02:49.458256896	3147.426211	307.802752	3295.968737	6.978712e+06
min	2020-05-15 00:00:00	0.000000	0.000000	0.000000	6.183645e+06
25%	2020-05-24 00:45:00	0.000000	0.000000	0.000000	6.512003e+06
50%	2020-06-01 14:30:00	429.000000	41.493750	2658.714286	7.146685e+06
75%	2020-06-09 20:00:00	6366.964286	623.618750	6274.000000	7.268706e+06
max	2020-06-17 23:45:00	14471.125000	1410.950000	9163.000000	7.846821e+06
std	NaN	4036.457169	394.396439	3145.178309	4.162720e+05

Observations:

1. The data was collected from 15 May 2020 to 17 June 2020. According to the India Meteorological Department, monsoon covered the whole country by 26 June 2020 and hit Kerala on June 1. So if the plants are in south-west India then the values from 1st June onwards may be affected by rain.
2. The difference between the avg. DC power and the avg. AC power is a lot. Something seems wrong because only around 10% of the DC power is being converted into AC.
3. There's a pretty big jump in the Q2 to Q3 and from Q3 to Q4 values in DC_POWER & AC_POWER.

In [44]:

```
plant1_sensor.describe()
```

Out[44]:

	DATE_TIME	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
count	3182	3182.000000	3182.000000	3182.000000
mean	2020-05-15 19:36:36.543054592	25.531606	31.091015	0.228313
min	2020-05-15 00:00:00	20.398505	18.140415	0.000000
25%	2020-05-15 09:15:00	22.705182	21.090553	0.000000
50%	2020-05-15 18:15:00	24.613814	24.618060	0.024653
75%	2020-05-16 06:45:00	27.920532	41.307840	0.449588
max	2020-05-16 15:45:00	35.252486	65.545714	1.221652
std	NaN	3.354856	12.261222	0.300836

Observations:

There is a pretty big difference between the AMBIENT_TEMPERATURE & MODULE_TEMPERATURE values at Q3 & Q4.

PLANT 2

In [45]:

```
plant2_generation.describe()
```

Out[45]:

	DATE_TIME	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
count	67698	67698.000000	67698.000000	67698.000000	6.769800e+04
mean	2020-06-01 01:45:59.159798016	246.701961	241.277825	3294.890295	6.589448e+08
min	2020-05-15 00:00:00	0.000000	0.000000	0.000000	0.000000e+00
25%	2020-05-23 21:15:00	0.000000	0.000000	272.750000	1.996494e+07
50%	2020-06-01 08:15:00	0.000000	0.000000	2911.000000	2.826276e+08
75%	2020-06-09 10:45:00	446.591667	438.215000	5534.000000	1.348495e+09
max	2020-06-17 11:30:00	1420.933333	1385.420000	9873.000000	2.247916e+09
std	NaN	370.569597	362.112118	2919.448386	7.296678e+08

Observations:

1. The data collection dates of both plants are the same.
2. Unlike Plant 1, the DC_POWER & AC_POWER of Plant 2 is in line.
3. Consequently, there isn't much difference in the Q3 & Q4 values of DC_POWER & AC_POWER.

In [46]:

```
plant2_sensor.describe()
```

Out[46]:

	DATE_TIME	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
count	3259	3259.000000	3259.000000	3259.000000
mean	2020-05-15 20:05:55.968088064	28.069400	32.772408	0.232737
min	2020-05-15 00:00:00	20.942385	20.265123	0.000000
25%	2020-05-15 09:15:00	24.602135	23.716881	0.000000
50%	2020-05-15 18:30:00	26.981263	27.534606	0.019040
75%	2020-05-16 07:30:00	31.056757	40.480653	0.438717
max	2020-05-16 16:45:00	39.181638	66.635953	1.098766
std	NaN	4.061556	11.344034	0.312693

Observation:

1. There isn't much difference in the avg, Q1, Q2 & Q3 values of AMBIENT_TEMPERATURE & MODULE_TEMPERATURE.
2. The max value of MODULE_TEMPERATURE is much higher than the max value of AMBIENT_TEMPERATURE.

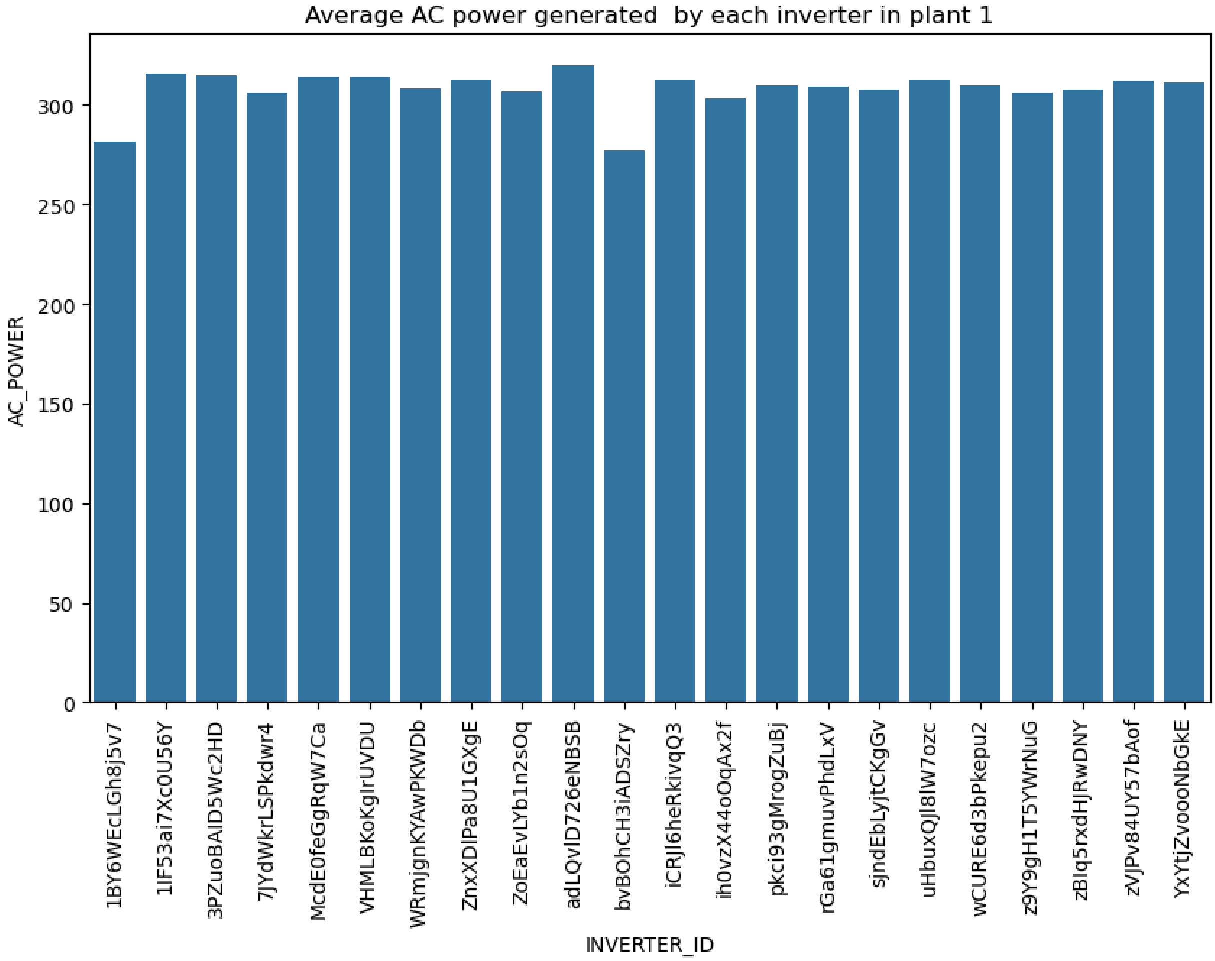
Comparison between Plant 1 & Plant 2:

1. The average DC Power produced by Plant 1 is 13x the average DC power produced by Plant 2.
2. But the average AC Power produced by both is almost the same. There is something definitely wrong with Plant 1's DC Power data.
3. The daily yield of both the plants is similar.
4. But the average TOTAL_YIELD OF Plant 2 is 7x of Plant 1.
5. Plant 1 & Plant 2 get the same amount of irradiation.
6. The average module & ambient temperatures of both plants is also similar.

Analyzing the Inverters in both plants

In [47]:

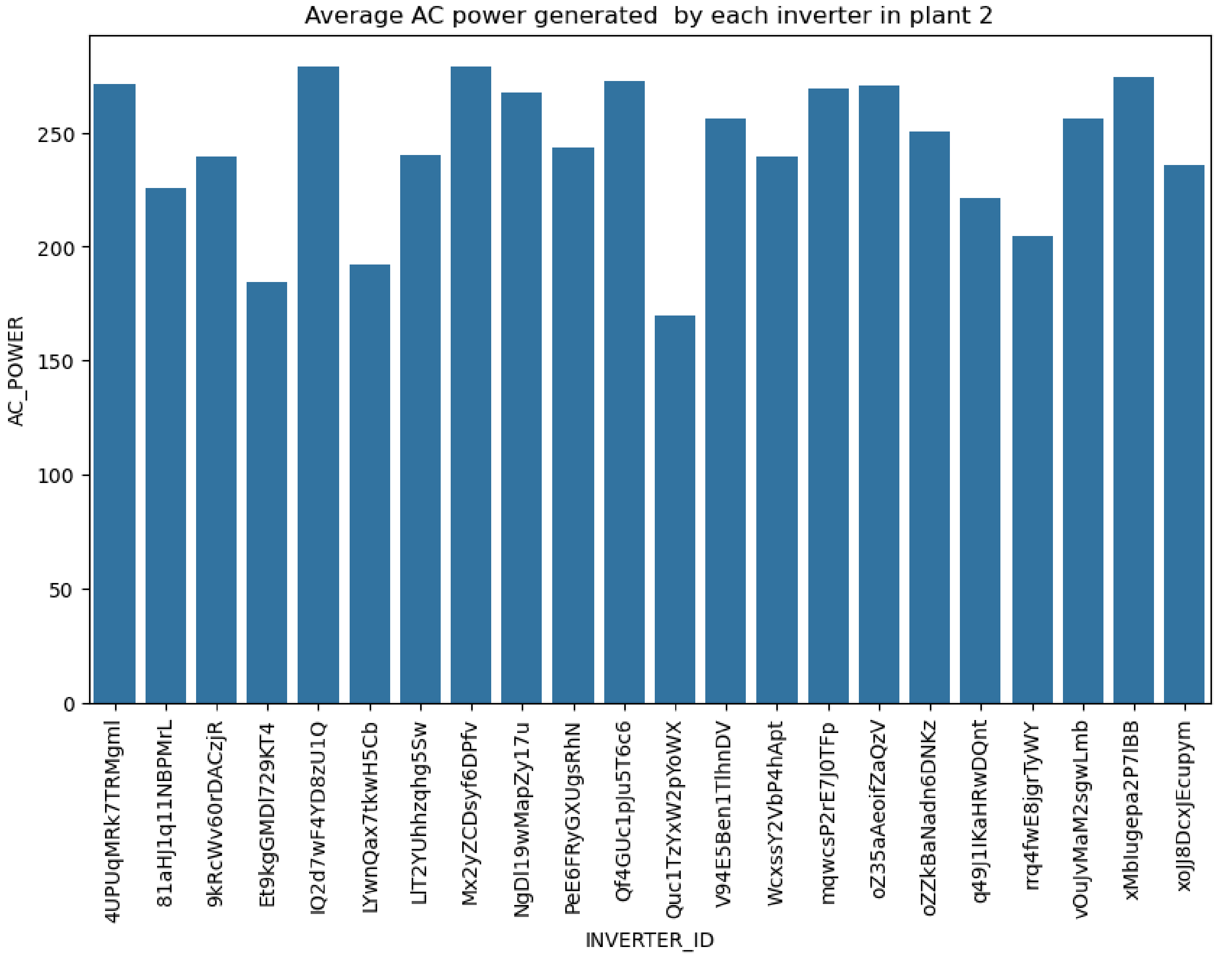
```
plt.figure(figsize=(10, 6))
sns.barplot(data=plant1_generation, x="INVERTER_ID", y="AC_POWER", errorbar=None).set(title="Average
plt.xticks(rotation=90)
plt.show()
```



Observation:

All inverters in plant 1 produce the same amount of AC POWER except for two that produce less.

```
In [48]:  plt.figure(figsize=(10,6))
          sns.barplot(data=plant2_generation, x="INVERTER_ID", y="AC_POWER", errorbar=None).set(title="Average
          plt.xticks(rotation=90)
          plt.show()
```

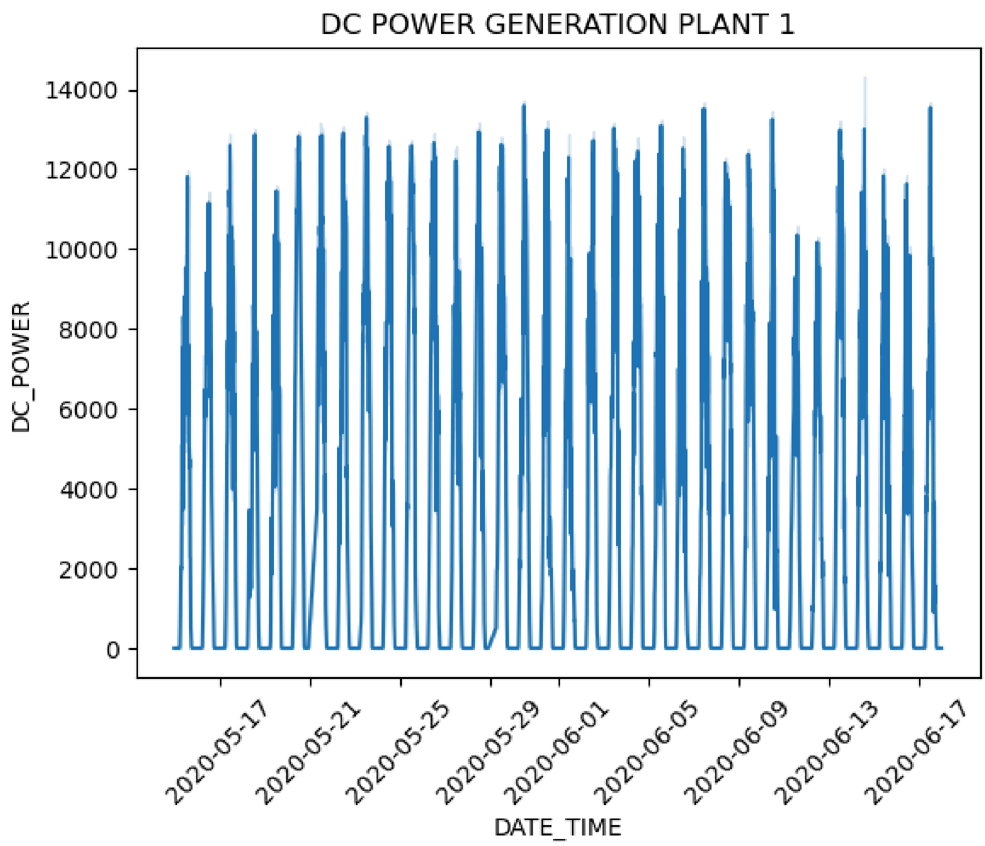


Observation:

The AC POWER production of inverters in plant 2 is all over the place, with 4 inverters performing very poorly.

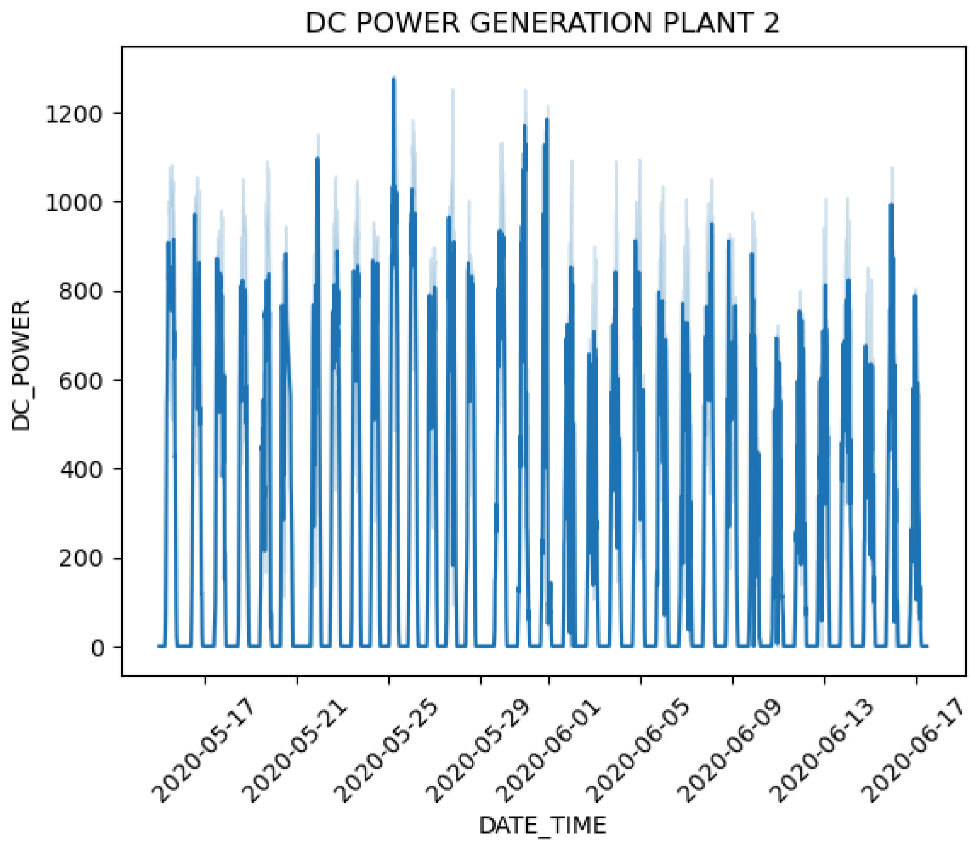
DC POWER Generation in the plants

In [49]:
```
sns.lineplot(data=plant1_generation, x='DATE_TIME', y='DC_POWER').set(title="DC POWER GENERATION PLAI
plt.xticks(rotation=45)
plt.show()
```



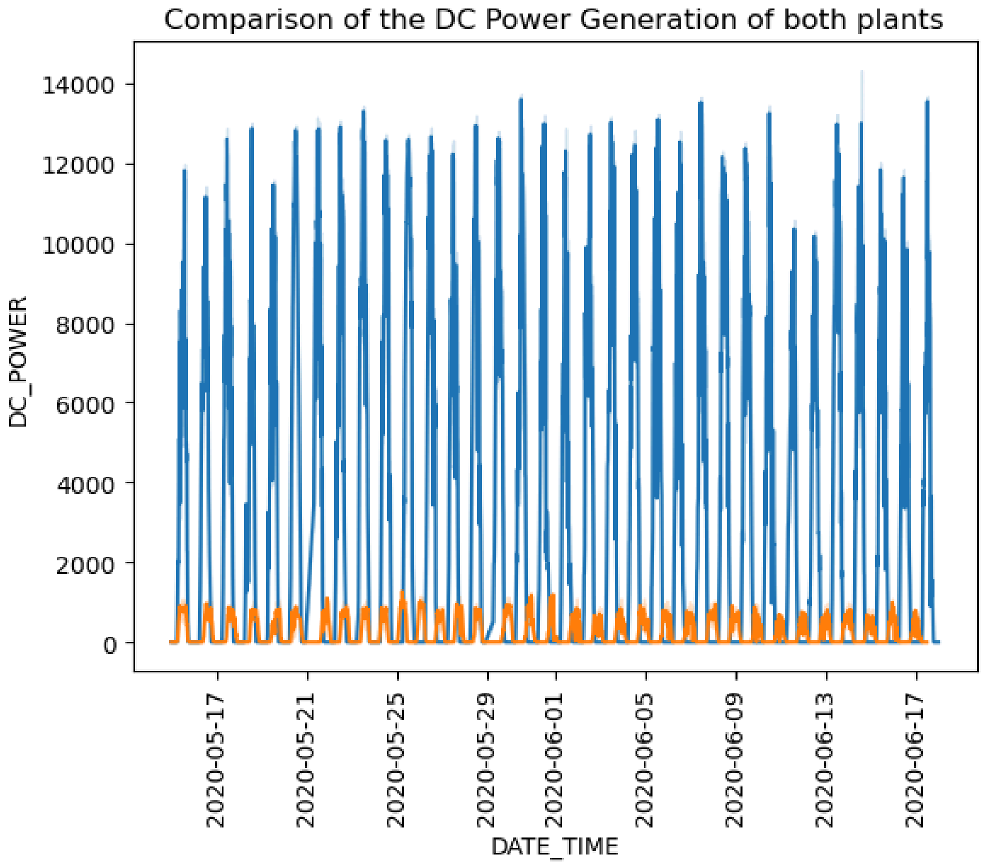
In [50]:

```
sns.lineplot(data=plant2_generation, x='DATE_TIME', y='DC_POWER').set(title="DC POWER GENERATION PLA
plt.xticks(rotation=45)
plt.show()
```



In [51]:

```
sns.lineplot(data=plant1_generation, x='DATE_TIME', y='DC_POWER')
sns.lineplot(data=plant2_generation, x='DATE_TIME', y='DC_POWER')
plt.title("Comparison of the DC Power Generation of both plants")
plt.xticks(rotation=90)
plt.show()
```



Observations:

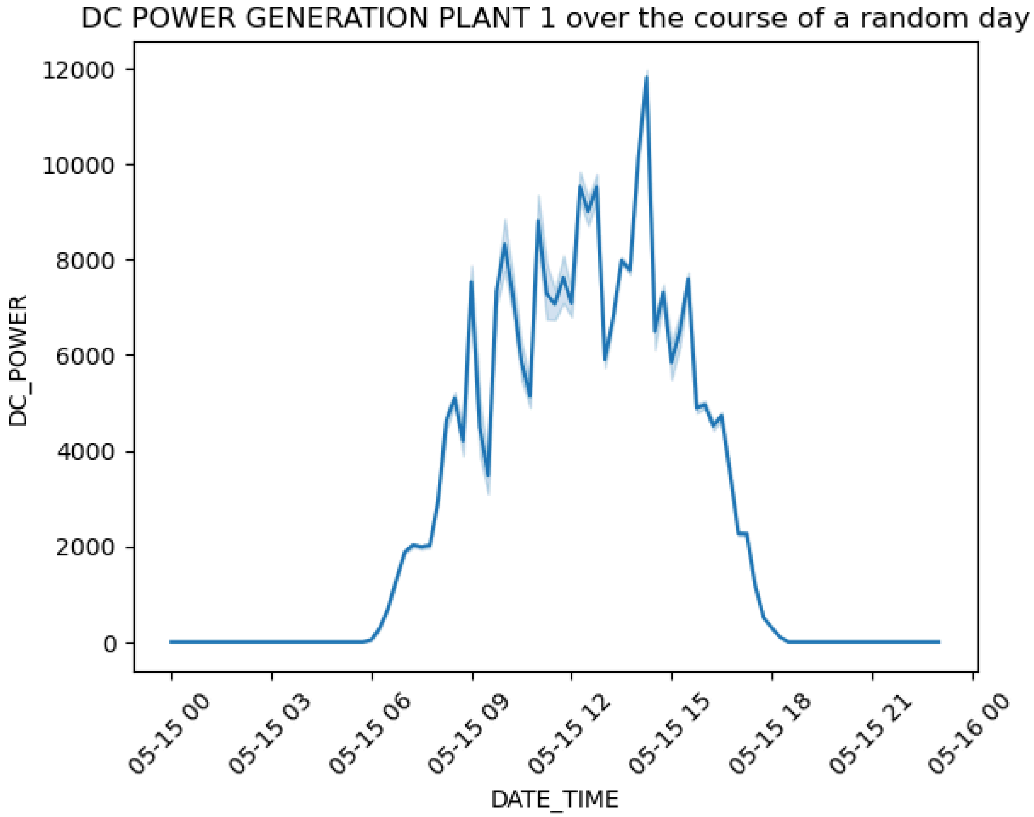
The DC power produced by Plant 1 (blue) is way higher than plant 2 (orange)

In [52]:

```
selected_day = '2020-05-15'
df_single_day = plant1_generation[plant1_generation['DATE_TIME'].dt.date == pd.to_datetime(selected_
```

In [53]:

```
sns.lineplot(data=df_single_day, x='DATE_TIME', y='DC_POWER').set(title="DC POWER GENERATION PLANT 1
plt.xticks(rotation=45)
plt.show()
```

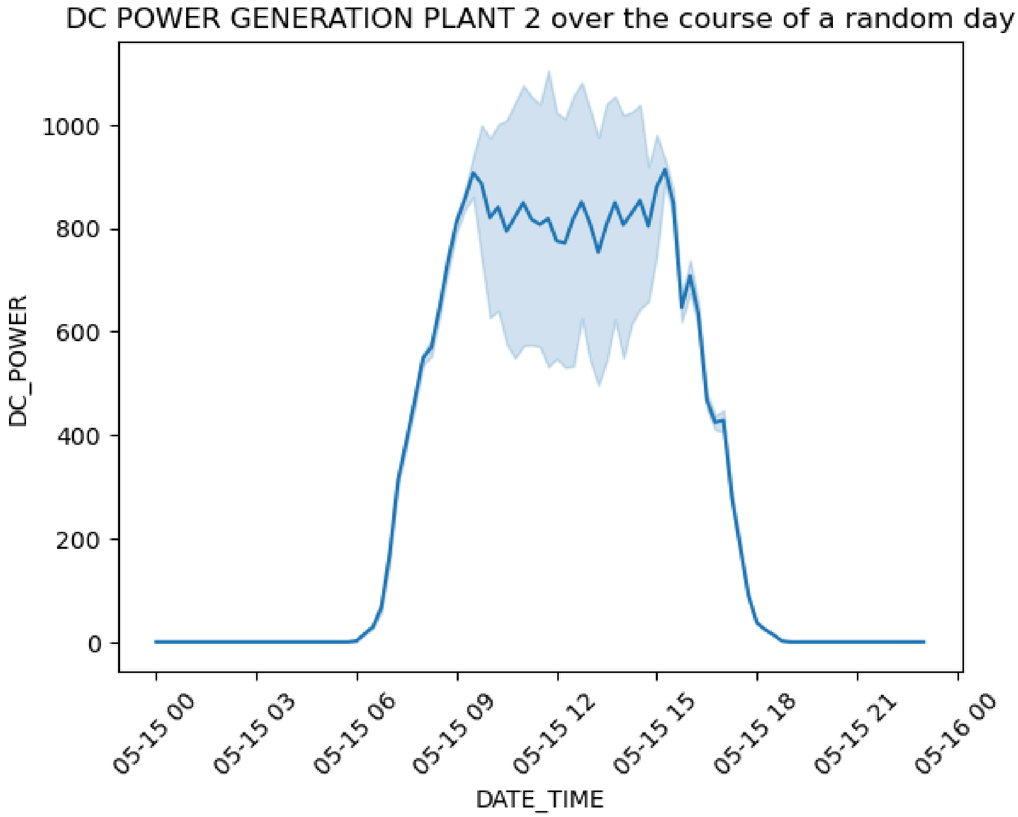


Observation:

DC power is generated only between 6 am to 6 pm which makes sense since those are the daylight hours. Most power was produced between the hours of 10 am to 3 pm.

In [54]:
```
df_single_day2 = plant2_generation[plant2_generation['DATE_TIME'].dt.date == pd.to_datetime(selected
```

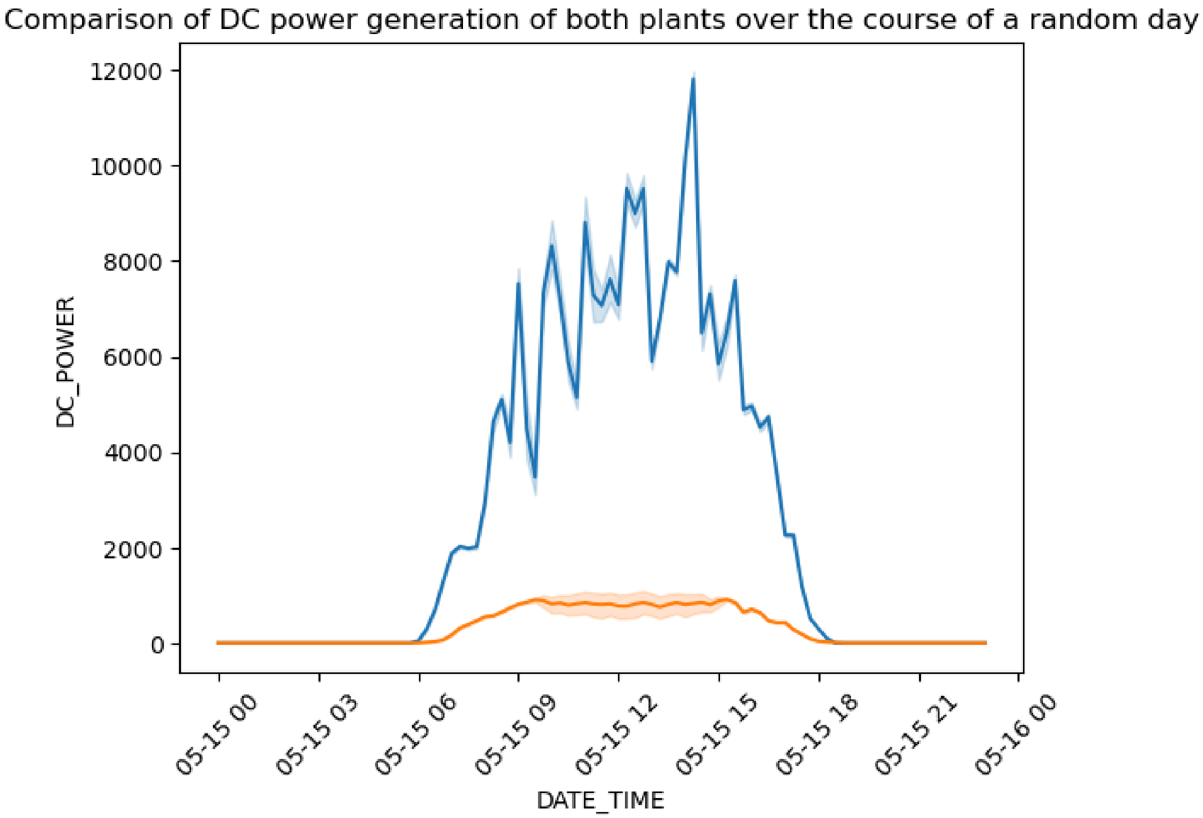
In [55]:
```
sns.lineplot(data=df_single_day2, x='DATE_TIME', y='DC_POWER').set(title="DC POWER GENERATION PLANT
plt.xticks(rotation=45)
plt.show()
```



Observations:

Power was only produced during 6 am to 6 pm which makes sense since those are the daylight hours. Power production remained mostly constant throughout the day.

In [56]:
```
sns.lineplot(data=df_single_day, x='DATE_TIME', y='DC_POWER')
sns.lineplot(data=df_single_day2, x='DATE_TIME', y='DC_POWER')
plt.title("Comparison of DC power generation of both plants over the course of a random day")
plt.xticks(rotation=45)
plt.show()
```



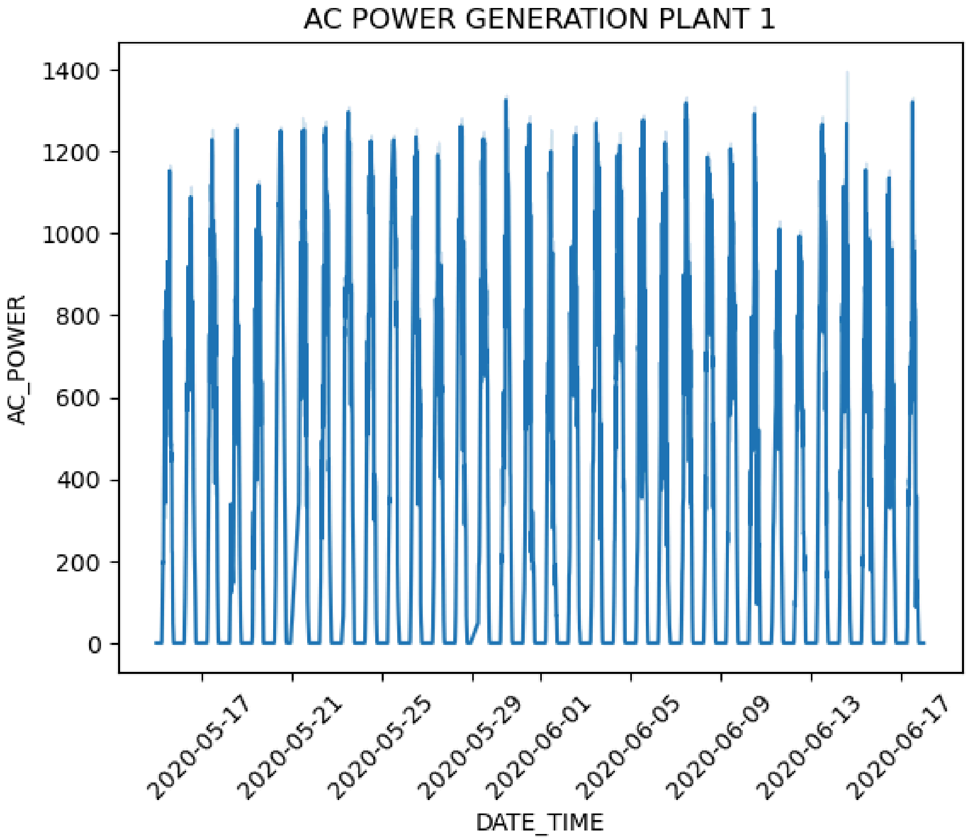
Observations:

Plant 1 produces more DC power than plant 2

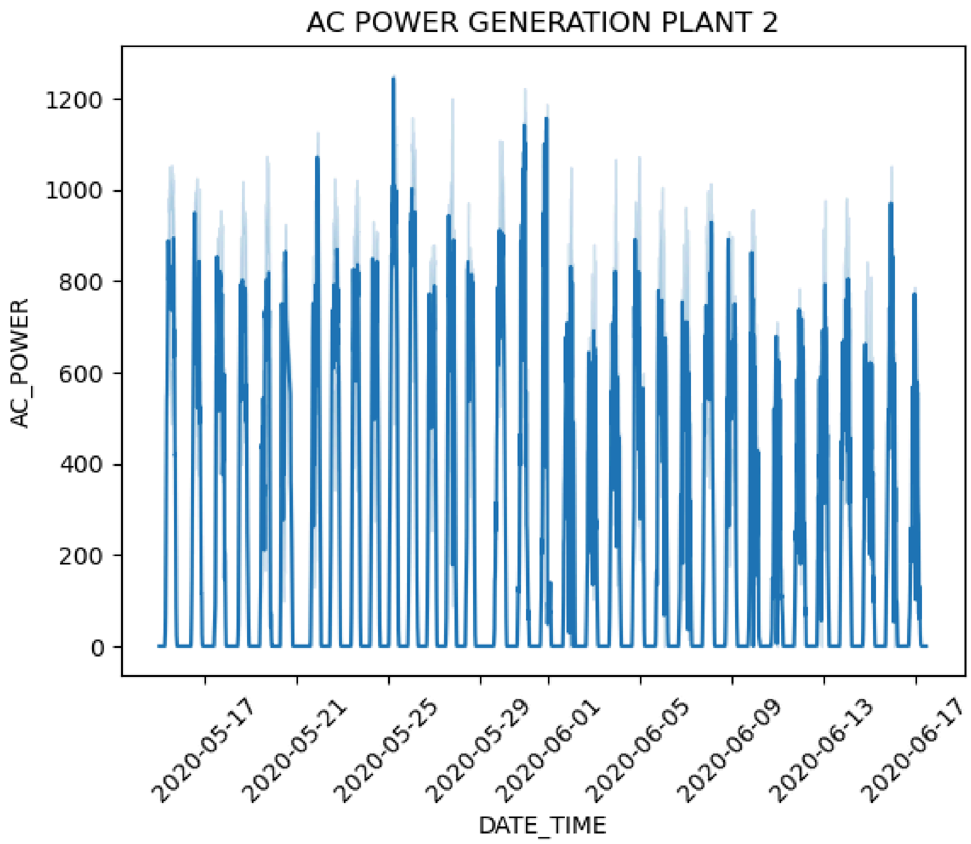
AC Power generation in the plants

In [57]:

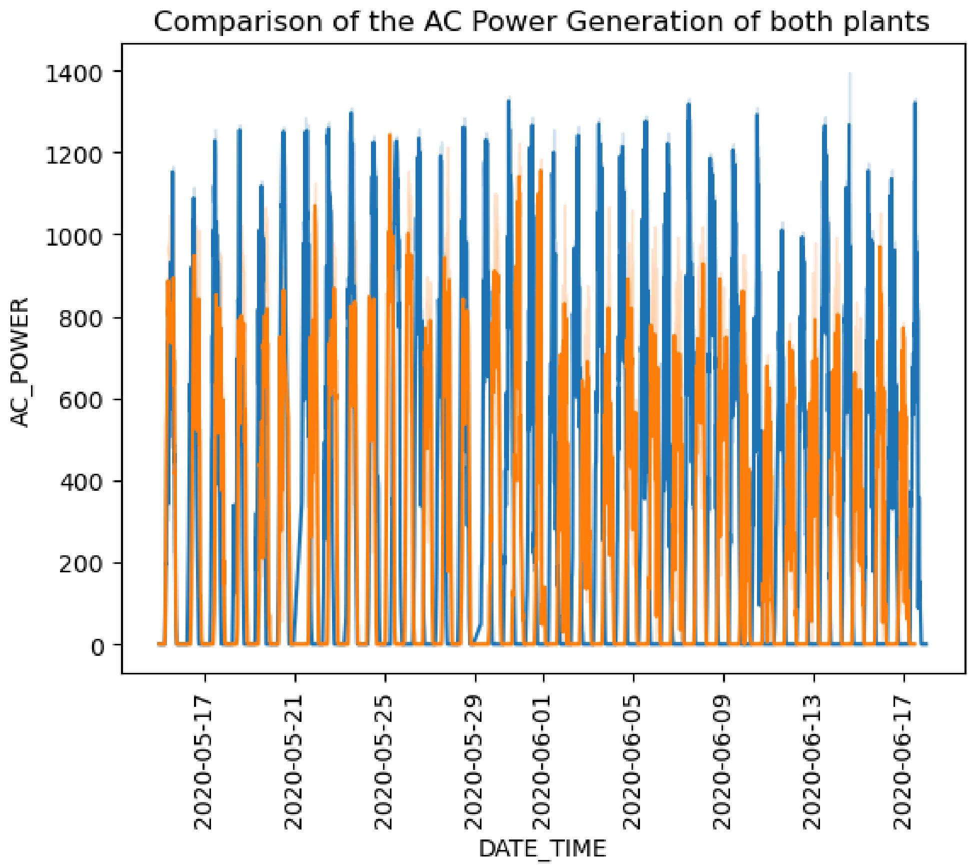
```
sns.lineplot(data=plant1_generation, x='DATE_TIME', y='AC_POWER').set(title="AC POWER GENERATION PLA
plt.xticks(rotation=45)
plt.show()
```



```
In [58]: sns.lineplot(data=plant2_generation, x='DATE_TIME', y='AC_POWER').set(title="AC POWER GENERATION PLA
         plt.xticks(rotation=45)
         plt.show()
```



```
In [59]: sns.lineplot(data=plant1_generation, x='DATE_TIME', y='AC_POWER')
         sns.lineplot(data=plant2_generation, x='DATE_TIME', y='AC_POWER')
         plt.title("Comparison of the AC Power Generation of both plants")
         plt.xticks(rotation=90)
         plt.show()
```



Observations:

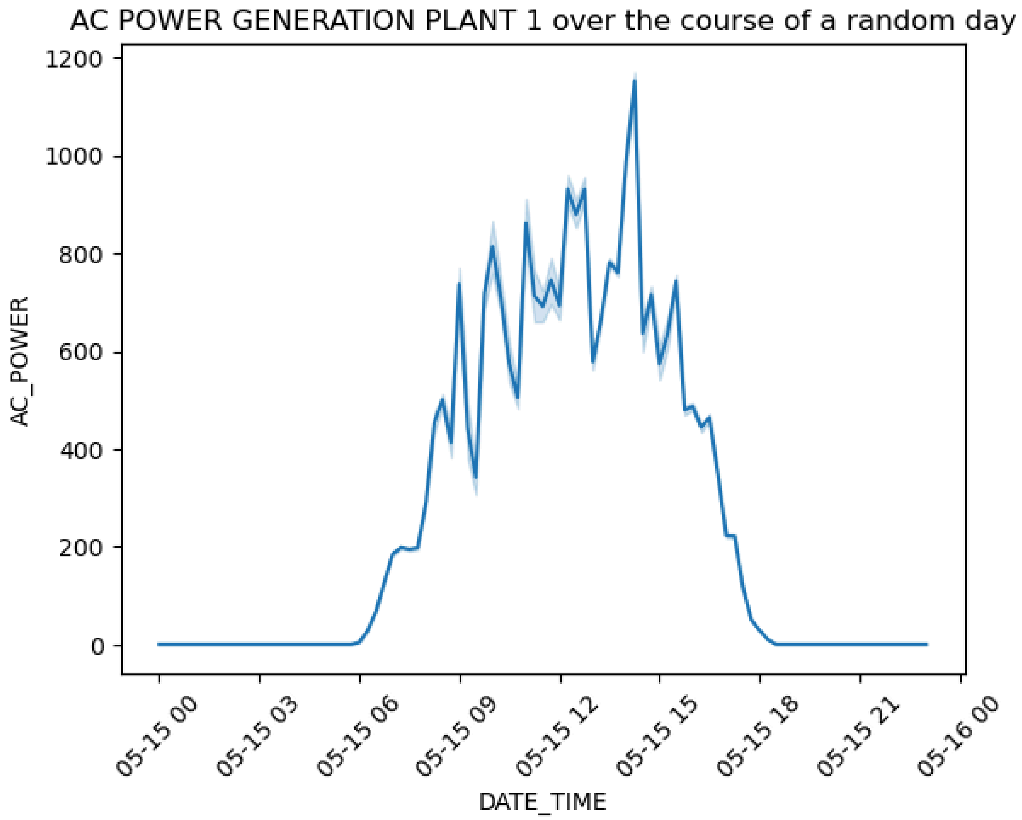
The AC power produced by Plant 1 (blue) is way higher than plant 2 (orange)

In [60]:

```
df_single_day3 = plant1_generation[plant1_generation['DATE_TIME'].dt.date == pd.to_datetime(selected
```

In [61]:

```
sns.lineplot(data=df_single_day3, x='DATE_TIME', y='AC_POWER').set(title="AC POWER GENERATION PLANT
plt.xticks(rotation=45)
plt.show()
```



Observation:

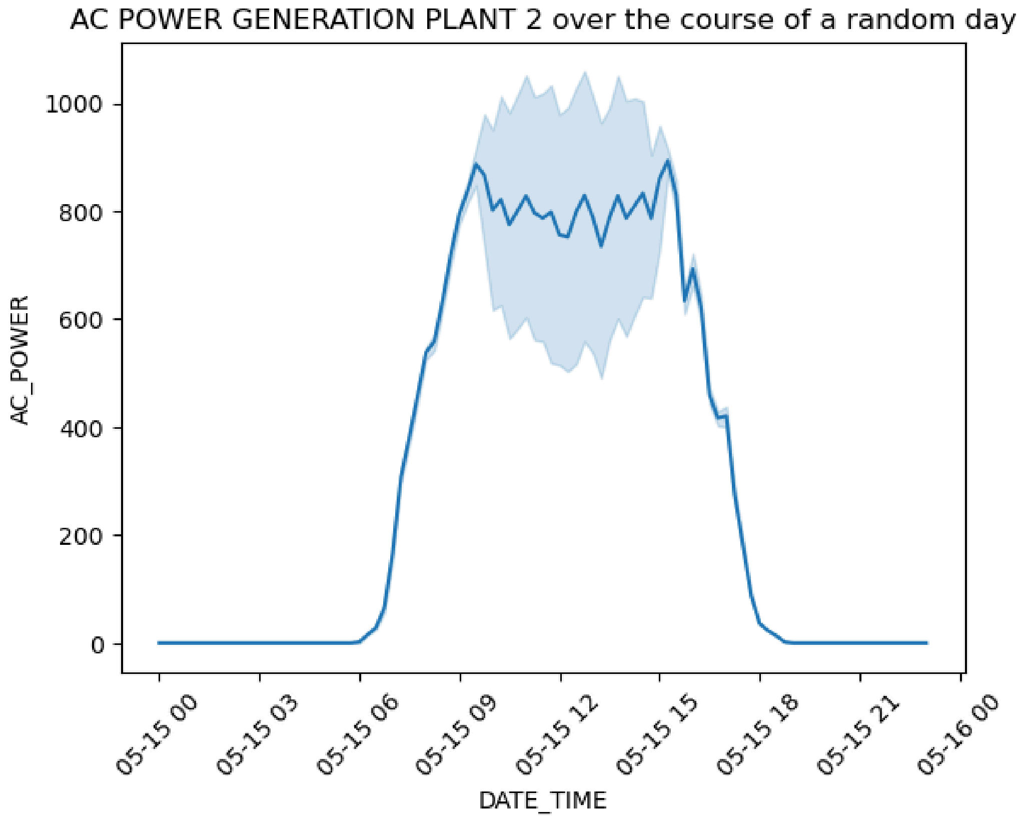
DC power is generated only between 6 am to 6 pm which makes sense since those are the daylight hours. Most power was produced between the hours of 10 am to 3 pm.

In [62]:

```
df_single_day4 = plant2_generation[plant2_generation['DATE_TIME'].dt.date == pd.to_datetime(selected
```

In [63]:

```
sns.lineplot(data=df_single_day4, x='DATE_TIME', y='AC_POWER').set(title="AC POWER GENERATION PLANT
plt.xticks(rotation=45)
plt.show()
```

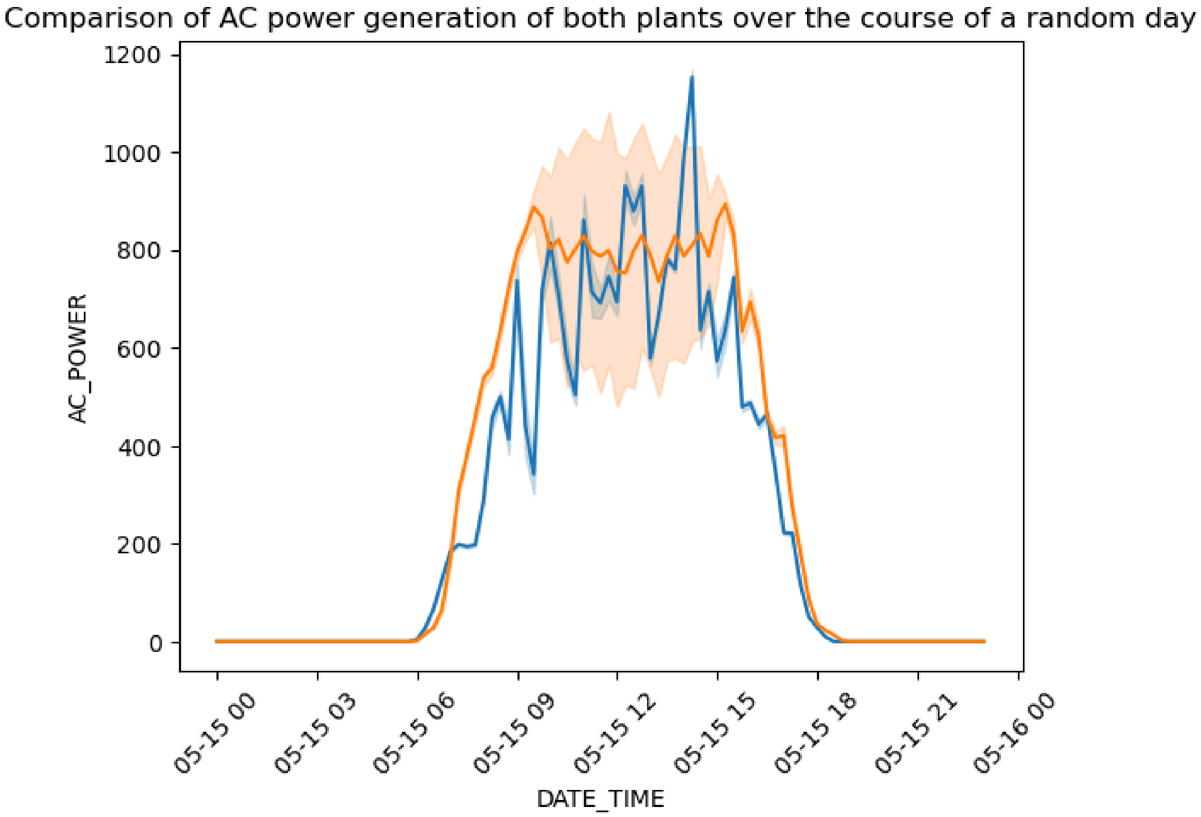


Observations:

Power was only produced during 6 am to 6 pm which makes sense since those are the daylight hours. Power production remained mostly constant throughout the day.

In [64]:

```python
sns.lineplot(data=df_single_day3, x='DATE_TIME', y='AC_POWER')
sns.lineplot(data=df_single_day4, x='DATE_TIME', y='AC_POWER')
plt.title("Comparison of AC power generation of both plants over the course of a random day")
plt.xticks(rotation=45)
plt.show()
```



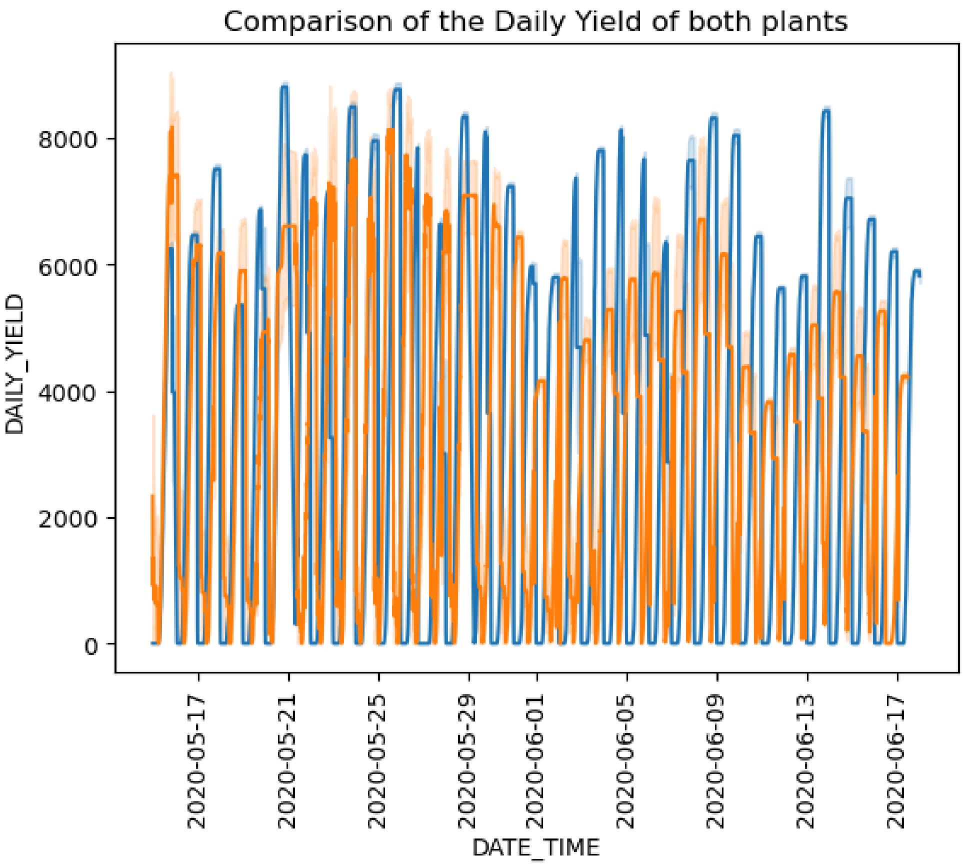
Observations:

AC Power produced by plant 1 throughout a day fluctatuates a lot but plant 2 remains fairly constant. But overall, there isn't a massive difference in scale the way there is with the DC power.

Comparison of Daily Yield of Both Plants:

In [65]:

```
sns.lineplot(data=plant1_generation, x='DATE_TIME', y='DAILY_YIELD')
sns.lineplot(data=plant2_generation, x='DATE_TIME', y='DAILY_YIELD')
plt.title("Comparison of the Daily Yield of both plants")
plt.xticks(rotation=90)
plt.show()
```



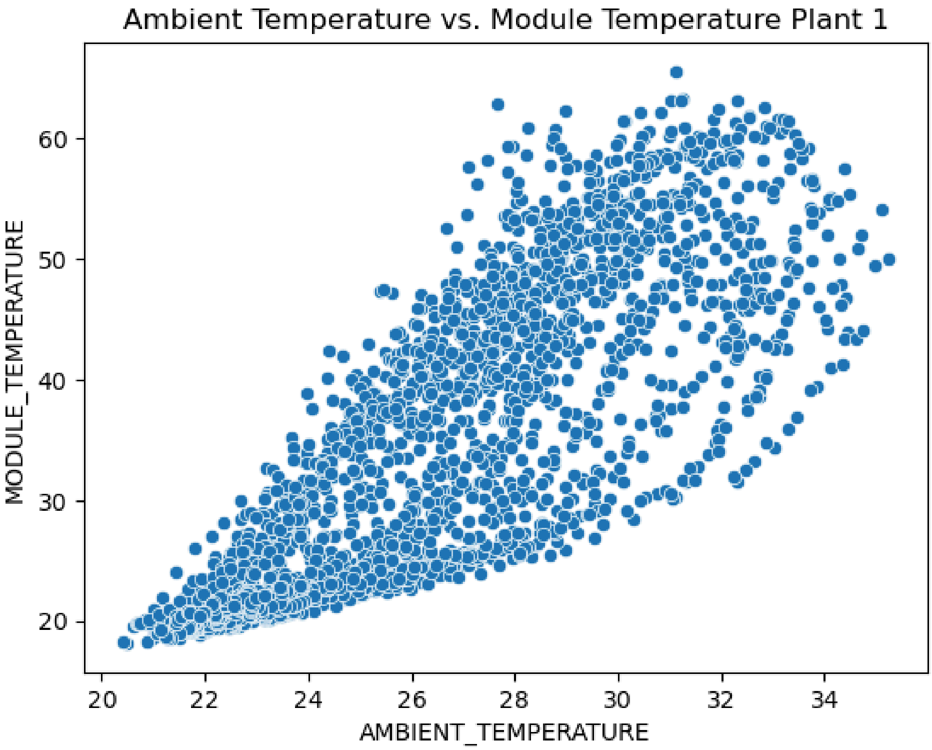
Observations:

1. On average, the daily yield of plant 1 (blue) is much greater than plant 2 (orange).
2. The daily yield of plant 2 dropped after June 1. We can only assume because of monsoon.

Relationship between Ambient Temperature & Module Temperature:

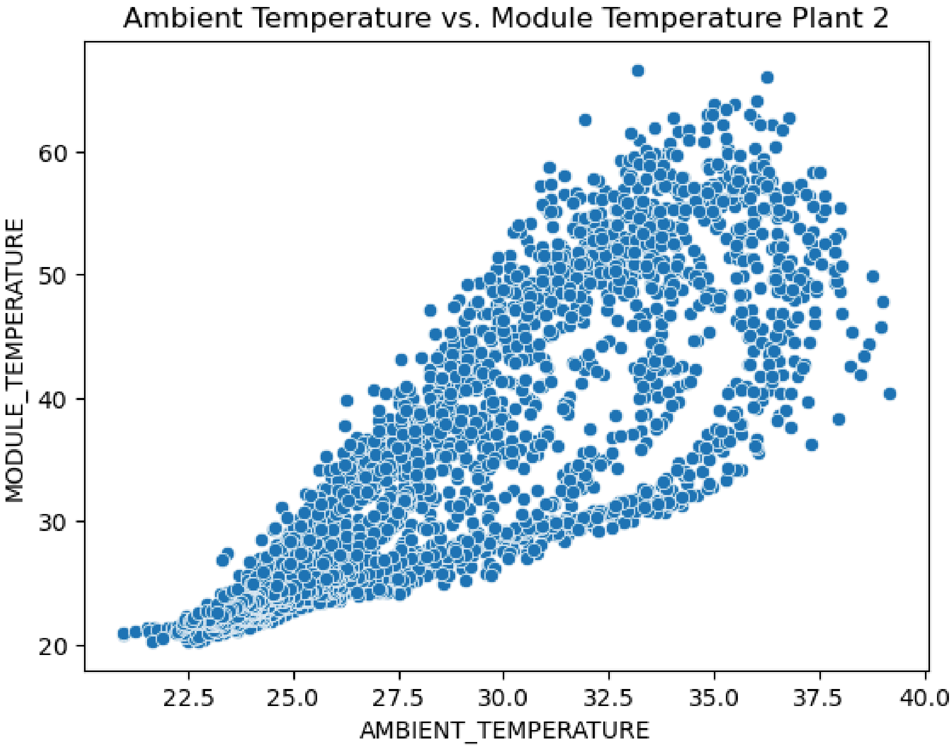
In [66]:

```
sns.scatterplot(data=plant1_sensor, x="AMBIENT_TEMPERATURE", y="MODULE_TEMPERATURE").set(title="Ambi
plt.show()
```



In [67]:

```
sns.scatterplot(data=plant2_sensor, x="AMBIENT_TEMPERATURE", y="MODULE_TEMPERATURE").set(title="Ambi
plt.show()
```



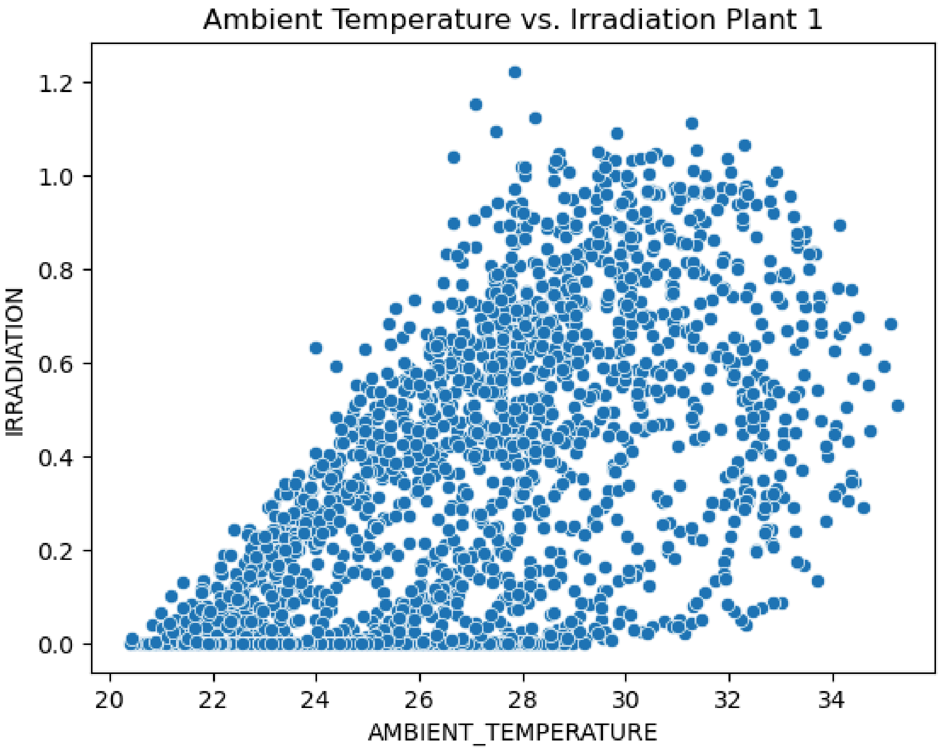
Observation:

There is a Positive Correlation between Module Temperature and Ambient Temperature. The Module Temperature increases as the Ambient Temperature increases.

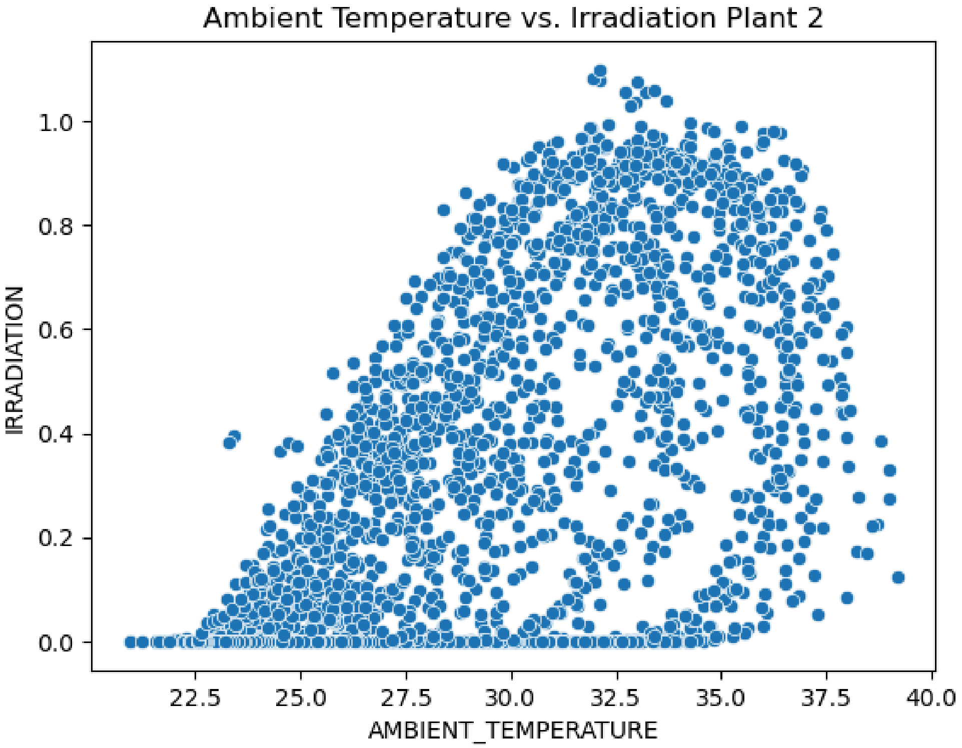
Relationship between Ambient Temperature & Irradiation:

In [68]:

```
sns.scatterplot(data=plant1_sensor, x="AMBIENT_TEMPERATURE", y="IRRADIATION").set(title="Ambient Tem
plt.show()
```

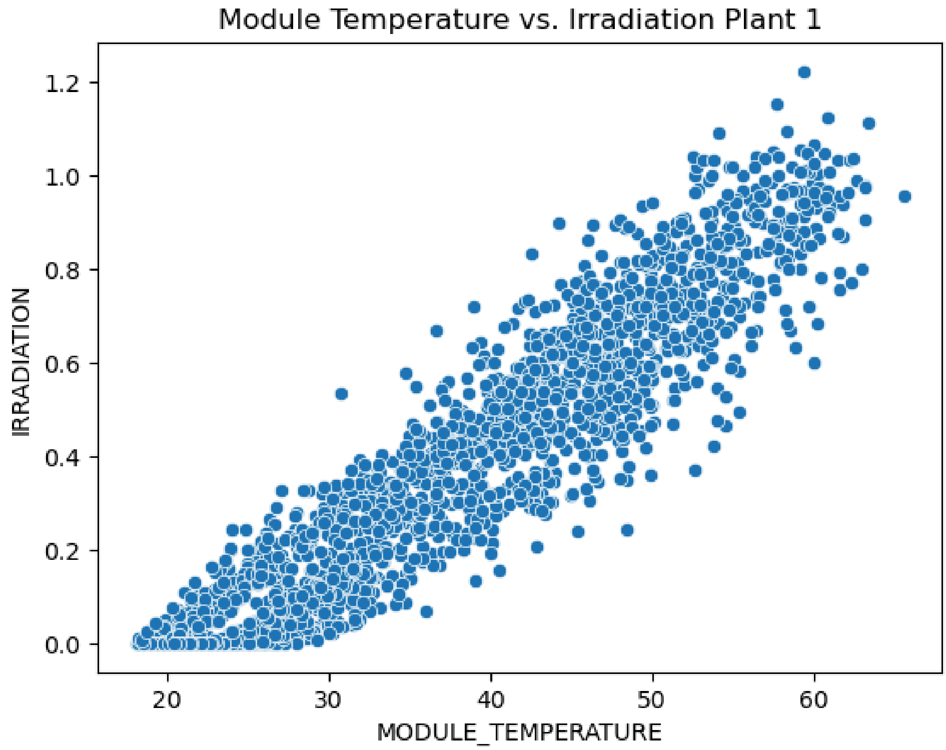


```
In [69]: sns.scatterplot(data=plant2_sensor, x="AMBIENT_TEMPERATURE", y="IRRADIATION").set(title="Ambient Tem
         plt.show()
```

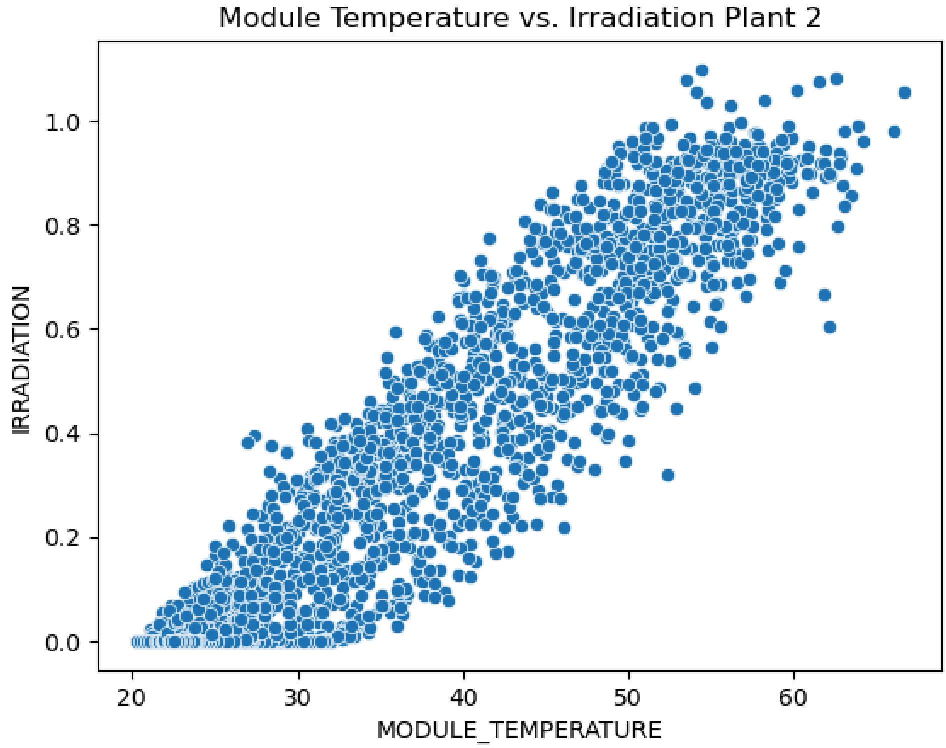


Relationship between Module Temperature & Irradiation:

```
In [70]: sns.scatterplot(data=plant1_sensor, x="MODULE_TEMPERATURE", y="IRRADIATION").set(title="Module Tempe
         plt.show()
```



In [71]:
```
sns.scatterplot(data=plant2_sensor, x="MODULE_TEMPERATURE", y="IRRADIATION").set(title="Module Tempe
plt.show()
```



Observations:

There is a Positive Correlation between Irradiation and Module Temperature. Irradiation increases as Module Temperature increases.

Merging the Power Generation + Weather Sensor Data

In [81]:
```
plant1 = pd.merge(plant1_generation, plant1_sensor, on="DATE_TIME", how="left")
plant1.reset_index(drop=True)
```

Out[81]:

	DATE_TIME	INVERTER_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE
0	2020-05-15 00:00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0	25.184316
1	2020-05-15 00:00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0	25.084589
2	2020-05-15 00:00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0	24.935753
3	2020-05-15 00:00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0	24.846130
4	2020-05-15 00:00:00	1BY6WEcLGh8j5v7	0.0	0.0	0.000	6259559.0	24.621525
...	...	...	...	...	...	...	...
134125	2020-06-17 23:45:00	uHbuxQJl8lW7ozc	0.0	0.0	5967.000	7287002.0	NaN
134126	2020-06-17 23:45:00	wCURE6d3bPkepu2	0.0	0.0	5147.625	7028601.0	NaN
134127	2020-06-17 23:45:00	z9Y9gH1T5YWrNuG	0.0	0.0	5819.000	7251204.0	NaN
134128	2020-06-17 23:45:00	zBIq5rxdHJRwDNY	0.0	0.0	5817.000	6583369.0	NaN
134129	2020-06-17 23:45:00	zVJPv84UY57bAof	0.0	0.0	5910.000	7363272.0	NaN

134130 rows × 9 columns

In [82]:

```python
plant2 = pd.merge(plant2_generation, plant2_sensor, on="DATE_TIME", how="left")
plant2.reset_index(drop=True)
```

Out[82]:

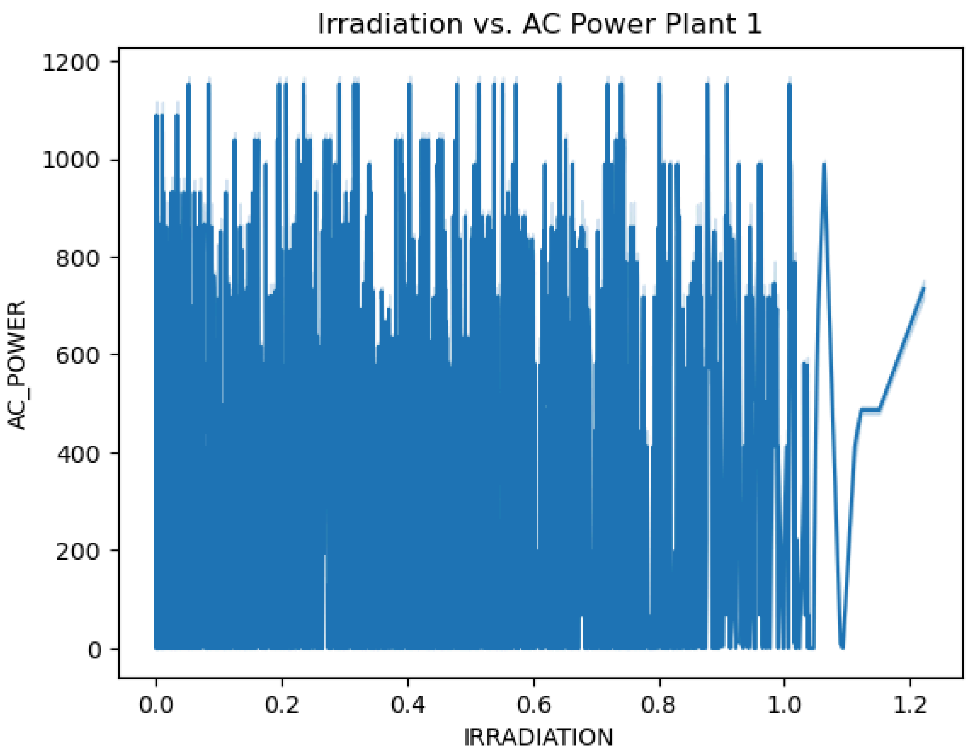
	DATE_TIME	INVERTER_ID	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	AMBIENT_TEMPERATURE
0	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.0	2429011.0	27.004764
1	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.0	2429011.0	26.880811
2	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.0	2429011.0	26.682055
3	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.0	2429011.0	26.500589
4	2020-05-15 00:00:00	4UPUqMRk7TRMgml	0.0	0.0	9425.0	2429011.0	26.596148
...	...	...	...	...	...	...	...
134651	2020-06-17 11:30:00	q49J1IKaHRwDQnt	0.0	0.0	4157.0	520758.0	NaN
134652	2020-06-17 11:30:00	rrq4fwE8jgrTyWY	0.0	0.0	3931.0	121131356.0	NaN
134653	2020-06-17 11:30:00	vOuJvMaM2sgwLmb	0.0	0.0	4322.0	2427691.0	NaN
134654	2020-06-17 11:30:00	xMbIugepa2P7lBB	0.0	0.0	4218.0	106896394.0	NaN
134655	2020-06-17 11:30:00	xoJJ8DcxJEcupym	0.0	0.0	4316.0	209335741.0	NaN

134656 rows × 9 columns

Exploring the Relationship between AC Power and Irradiation

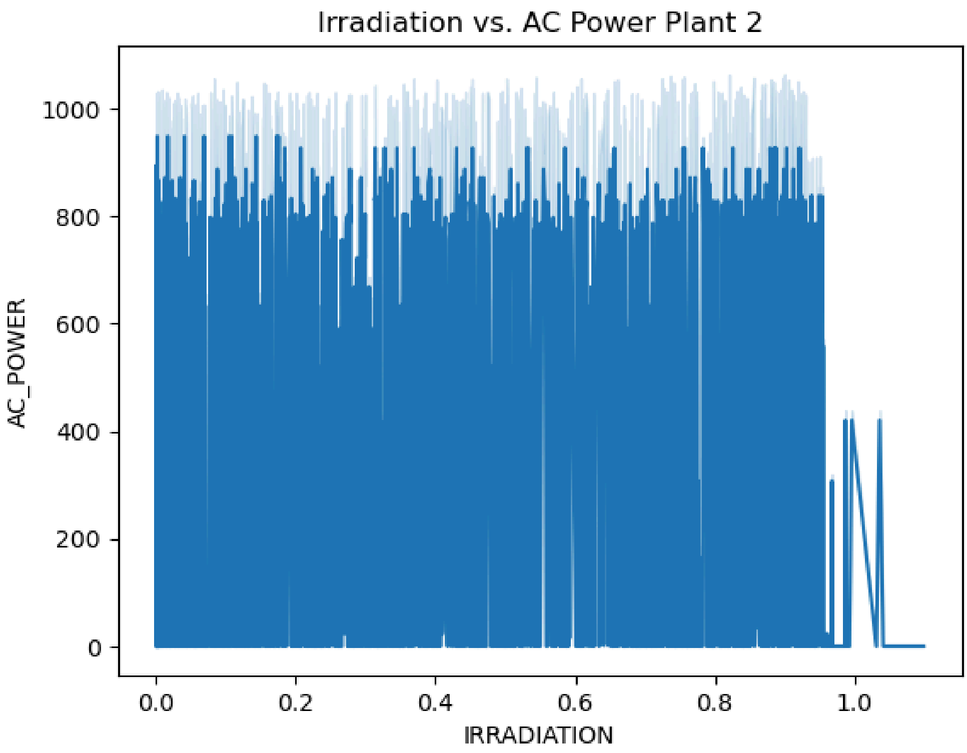
In [83]:

```
sns.lineplot(data=plant1, x="IRRADIATION", y="AC_POWER").set(title="Irradiation vs. AC Power Plant 1
plt.show()
```



In [84]:

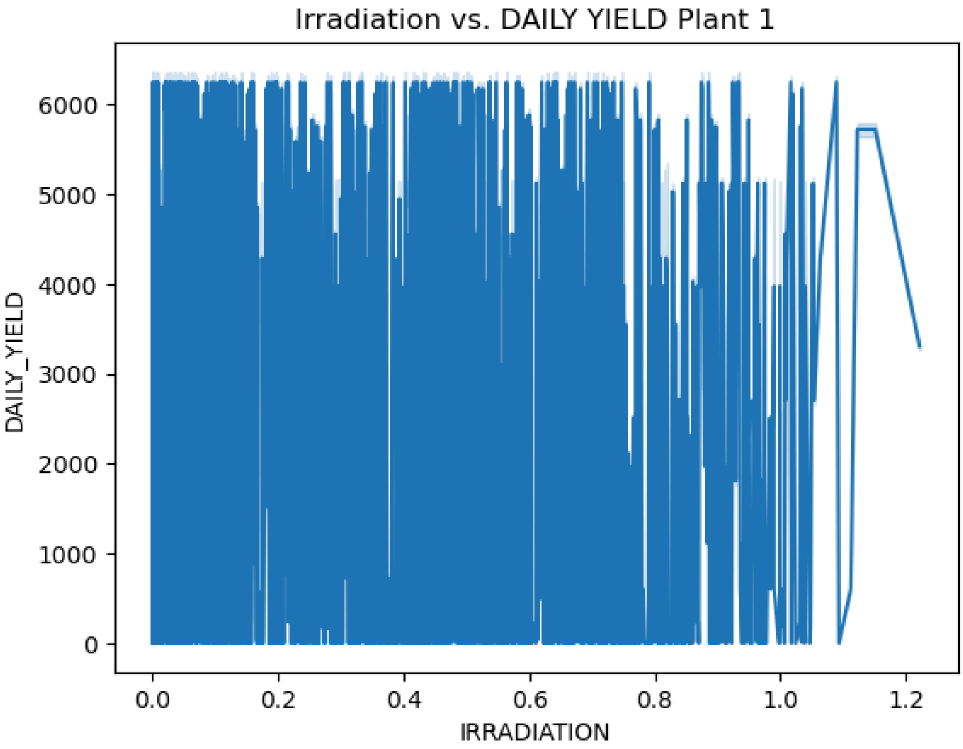
```
sns.lineplot(data=plant2, x="IRRADIATION", y="AC_POWER").set(title="Irradiation vs. AC Power Plant 2
plt.show()
```



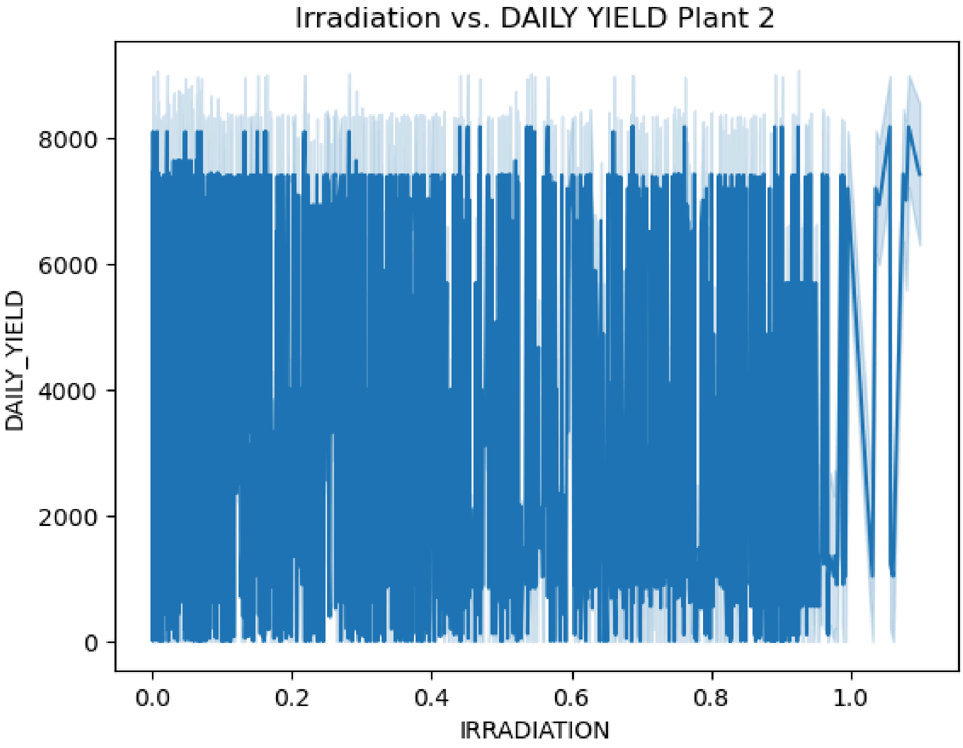
Exploring the Relationship between Daily Yield and Irradiation

In [85]:

```
sns.lineplot(data=plant1, x="IRRADIATION", y="DAILY_YIELD").set(title="Irradiation vs. DAILY YIELD P
plt.show()
```

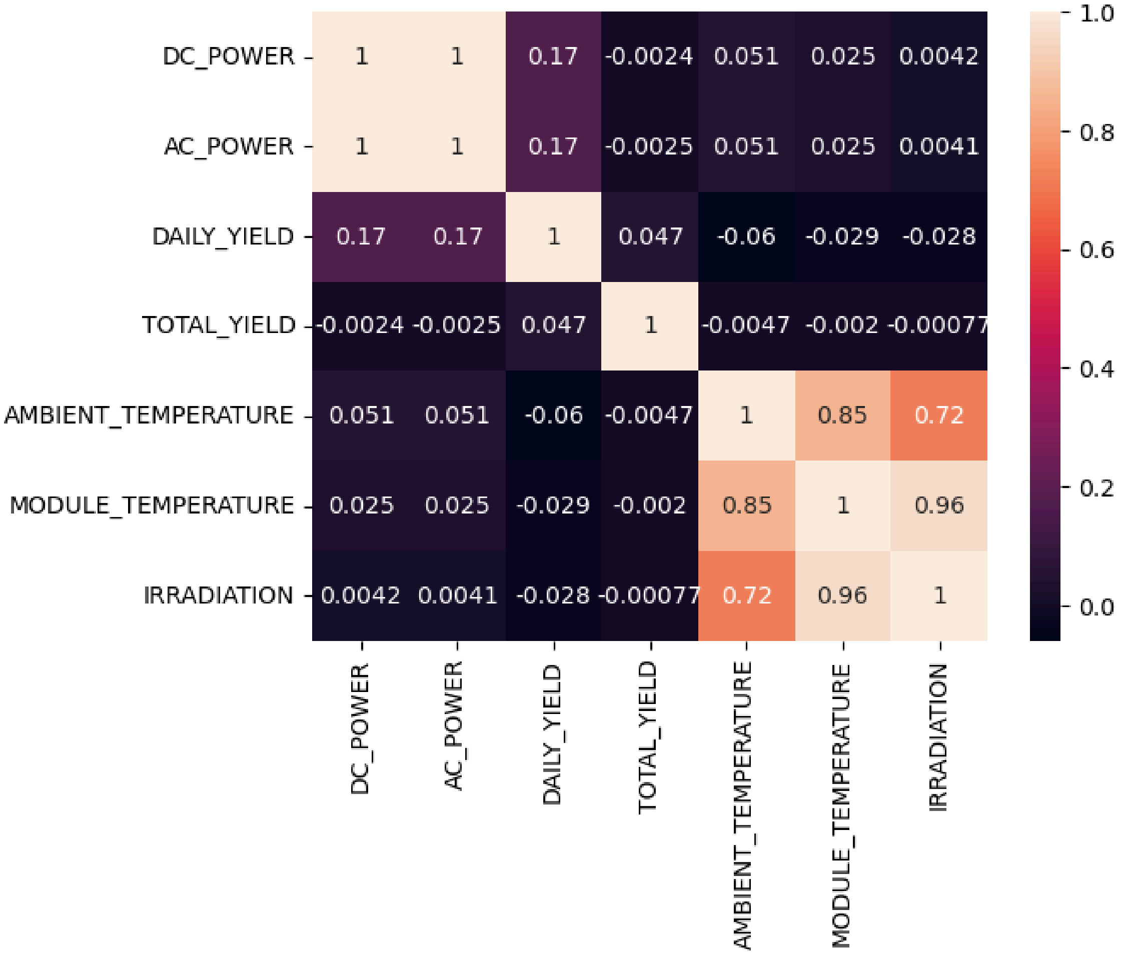


```
In [86]: sns.lineplot(data=plant2, x="IRRADIATION", y="DAILY_YIELD").set(title="Irradiation vs. DAILY YIELD P
         plt.show()
```

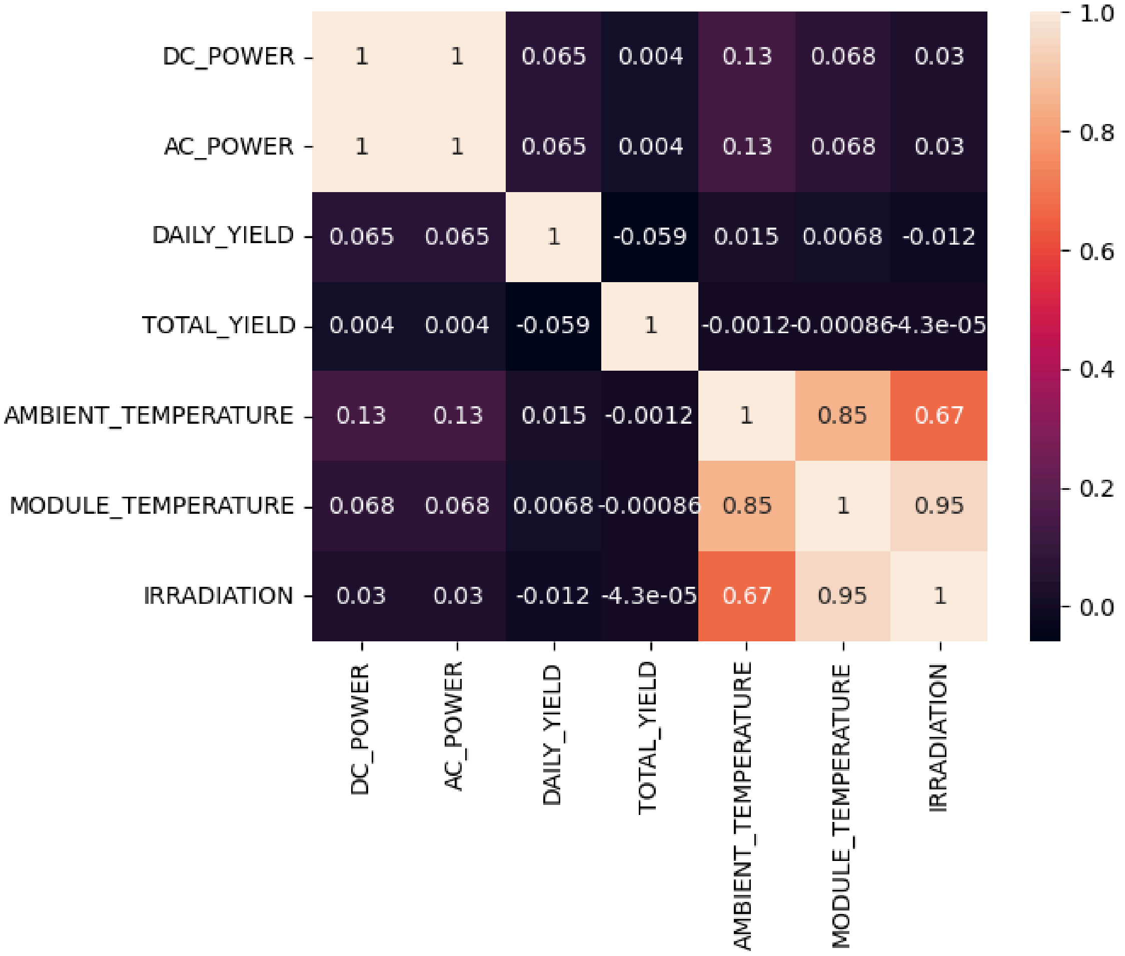


Exploring Correlation between features:

```
In [87]: sns.heatmap(plant1.corr(numeric_only=True), annot=True, fmt='.2g')
         plt.show()
```



```
In [88]: sns.heatmap(plant2.corr(numeric_only=True), annot=True, fmt='.2g')
         plt.show()
```



Observations:

1. Highest Positive Correlation is between Irradiation & Module Temperature.
2. A very high positive correlation between Module Temperature & Ambient Temperature
3. A high positive correlation between Irradiation & Ambient Temperature
4. Positive Correlations between all features except the ones that involve Total Yield.

Exporting Merged Dataframes as csv

In [89]:

```
plant1.to_csv('plant1_merged.csv')
plant2.to_csv('plant2_merged.csv')
```