Visualiser les spread sur le marché elec dù à l'implémentation du renouvelable

visualisation repartition prix elec des jours où pas production elec/ solaire

```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
Histo=pd.read_excel(r'HistoPrix.xlsx', sheet_name='SpotElec')
```

```
histo=Histo.iloc[1:,2:27]
histo.head()
```

	Unnamed: 2	Unnamed: 3	NaN	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11	...	Unnamed: 17	Unnamed: 18	Unnam
1	NaN	00:00:00	01:00:00	02:00:00	03:00:00	04:00:00	05:00:00	06:00:00	07:00:00	08:00:00	...	14:00:00	15:00:00	16:0(
2	Timestamp	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	...	Trade Close	Trade Close	Tr Cl
3	2018-01-01 00:00:00	14.43	10.26	7.48	4.52	4.24	8.36	22.58	41.92	47.95	...	46.1	43.86	4:
4	2018-01-02 00:00:00	12.14	5.48	5	0.99	-0.25	3.87	12.25	19.42	22.57	...	16.76	18.09	1(
5	2018-01-03 00:00:00	11.76	3.46	2.11	2.63	4.99	11.08	22.34	33.59	40.45	...	32.51	32.77	3:

5 rows × 25 columns

```
histo.shape
```

```
(102613, 25)
```

```
histo.columns=['Date', '01:00', '02:00', '03:00', '04:00', '05:00', '06:00', '07:00', '08:00', '09:00', '10:00', '11:00', '12:00',
histo.head()
```

	Date	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	...	15:00	16:00	17:00	18:
1	NaN	00:00:00	01:00:00	02:00:00	03:00:00	04:00:00	05:00:00	06:00:00	07:00:00	08:00:00	...	14:00:00	15:00:00	16:00:00	17:00:
2	Timestamp	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	Trade Close	...	Trade Close	Trade Close	Trade Close	Tra Clo
3	2018-01-01 00:00:00	14.43	10.26	7.48	4.52	4.24	8.36	22.58	41.92	47.95	...	46.1	43.86	43.64	47.
4	2018-01-02 00:00:00	12.14	5.48	5	0.99	-0.25	3.87	12.25	19.42	22.57	...	16.76	18.09	19.51	24.
5	2018-01-03 00:00:00	11.76	3.46	2.11	2.63	4.99	11.08	22.34	33.59	40.45	...	32.51	32.77	33.81	50.

5 rows × 25 columns

```
Commencez à coder ou à générer avec l'IA.
```

```
histo_melt=pd.melt(histo, id_vars='Date', var_name='Heure', value_name='Prix')
histo_melt.head()
```

	Date	Heure	Prix
0	NaN	01:00	00:00:00
1	Timestamp	01:00	Trade Close
2	2018-01-01 00:00:00	01:00	14.43
3	2018-01-02 00:00:00	01:00	12.14
4	2018-01-03 00:00:00	01:00	11.76

```
histo_melt['date']=pd.to_datetime(histo_melt['Date'], format='%Y-%m-%d %H:%M:%S',errors='coerce')

```

```
sorted_histo=histo_melt.sort_values(by=['Date', 'Heure'], ascending=[True, True])
sorted_histo.head(30)
```

	Date	Heure	Prix	date
2	2018-01-01 00:00:00	01:00	14.43	2018-01-01
102615	2018-01-01 00:00:00	02:00	10.26	2018-01-01
205228	2018-01-01 00:00:00	03:00	7.48	2018-01-01
307841	2018-01-01 00:00:00	04:00	4.52	2018-01-01
410454	2018-01-01 00:00:00	05:00	4.24	2018-01-01
513067	2018-01-01 00:00:00	06:00	8.36	2018-01-01
615680	2018-01-01 00:00:00	07:00	22.58	2018-01-01
718293	2018-01-01 00:00:00	08:00	41.92	2018-01-01
820906	2018-01-01 00:00:00	09:00	47.95	2018-01-01
923519	2018-01-01 00:00:00	10:00	49.9	2018-01-01
1026132	2018-01-01 00:00:00	11:00	50	2018-01-01
1128745	2018-01-01 00:00:00	12:00	49.7	2018-01-01
1231358	2018-01-01 00:00:00	13:00	48.16	2018-01-01
1333971	2018-01-01 00:00:00	14:00	47.24	2018-01-01
1436584	2018-01-01 00:00:00	15:00	46.1	2018-01-01
1539197	2018-01-01 00:00:00	16:00	43.86	2018-01-01
1641810	2018-01-01 00:00:00	17:00	43.64	2018-01-01
1744423	2018-01-01 00:00:00	18:00	47.37	2018-01-01
1847036	2018-01-01 00:00:00	19:00	46.08	2018-01-01
1949649	2018-01-01 00:00:00	20:00	41.08	2018-01-01
2052262	2018-01-01 00:00:00	21:00	34.94	2018-01-01
2154875	2018-01-01 00:00:00	22:00	25.46	2018-01-01
2257488	2018-01-01 00:00:00	23:00	23.25	2018-01-01
2360101	2018-01-01 00:00:00	24:00	15.49	2018-01-01
3	2018-01-02 00:00:00	01:00	12.14	2018-01-02
102616	2018-01-02 00:00:00	02:00	5.48	2018-01-02
205229	2018-01-02 00:00:00	03:00	5	2018-01-02
307842	2018-01-02 00:00:00	04:00	0.99	2018-01-02
410455	2018-01-02 00:00:00	05:00	-0.25	2018-01-02
513068	2018-01-02 00:00:00	06:00	3.87	2018-01-02

```
sorted_histo['Heure']=sorted_histo['Heure'].replace("24:00", "00:00")
```

```
print(type(sorted_histo['Heure'].iloc[2]))
print(type(sorted_histo['date'].iloc[2]))
```

```
<class 'str'>
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

```
sorted_histo
```

	Date	Heure	Prix	date	start_date	Year
2	2018-01-01 00:00:00	01:00:00	14.43	2018-01-01	2018-01-01T01:00:00+01:00	2018.0
102615	2018-01-01 00:00:00	02:00:00	10.26	2018-01-01	2018-01-01T02:00:00+01:00	2018.0
205228	2018-01-01 00:00:00	03:00:00	7.48	2018-01-01	2018-01-01T03:00:00+01:00	2018.0
307841	2018-01-01 00:00:00	04:00:00	4.52	2018-01-01	2018-01-01T04:00:00+01:00	2018.0
410454	2018-01-01 00:00:00	05:00:00	4.24	2018-01-01	2018-01-01T05:00:00+01:00	2018.0
...	...	...	...	...	...	...
2462707	NaN	00:00:00	NaN	NaT	NaTT00:00:00+01:00	NaN
2462708	NaN	00:00:00	NaN	NaT	NaTT00:00:00+01:00	NaN
2462709	NaN	00:00:00	NaN	NaT	NaTT00:00:00+01:00	NaN
2462710	NaN	00:00:00	NaN	NaT	NaTT00:00:00+01:00	NaN
2462711	NaN	00:00:00	NaN	NaT	NaTT00:00:00+01:00	NaN

2462712 rows × 6 columns

```
sorted_histo['Heure']=pd.to_datetime(sorted_histo['Heure'], format='%H:%M', errors='coerce').dt.time
```

```
sorted_histo.head()
```

	Date	Heure	Prix	date
2	2018-01-01 00:00:00	01:00:00	14.43	2018-01-01
102615	2018-01-01 00:00:00	02:00:00	10.26	2018-01-01
205228	2018-01-01 00:00:00	03:00:00	7.48	2018-01-01
307841	2018-01-01 00:00:00	04:00:00	4.52	2018-01-01
410454	2018-01-01 00:00:00	05:00:00	4.24	2018-01-01

```
sorted_histo['date']=sorted_histo['date'].dt.date
sorted_histo['date']=pd.to_datetime(sorted_histo['date'], format='%Y-%m-%d', errors='coerce')
```

```
sorted_histo['start_date']=sorted_histo['date'].astype(str)+'T'+sorted_histo['Heure'].astype(str)+'+01:00'
```

```
sorted_histo['Year'] = sorted_histo['date'].dt.year

```

```
px.scatter(sorted_histo, x="start_date", y="Prix",title="Prix Spot de l'électricité en France en euro", width=1200, height=800)

```

```
Commencez à coder ou à générer avec l'IA.
```

Données de production

```
import requests
import base64

client_id='24ac945e-c636-4a70-a1aa-a54c09a7c142'
client_secret='731c955d-a37e-489e-abc5-eefeb55d23dc'

auth_url="https://digital.iservices.rte-france.com/token/oauth/"

credentials = f"{client_id}:{client_secret}"
encoded_credentials = base64.b64encode(credentials.encode()).decode()

headers = {
    'Authorization': f'Basic {encoded_credentials}',
    'Content-Type': 'application/x-www-form-urlencoded'
}

token=requests.post(auth_url, headers=headers).json()['access_token']
print(token)
```

```
0565dIQYL89vnMxHTAj02iAjdvlWidhGGRHYLTlvt01omM4LadHgYp
```

```
headers = {
    'Authorization': f'Bearer {token}'
}
url = 'https://digital.iservices.rte-france.com/open_api/generation_forecast/v2/forecasts'
params = {
    'production_type': 'SOLAR',
    'start_date': '2015-06-01T00:00:00Z',
    'end_date': '2015-06-19T00:00:00Z',
    'type':'D-1'
}
response = requests.get(url, headers=headers, params=params)
forecast=response.json()
forecast_values=forecast['forecasts'][0]
fore=pd.DataFrame(forecast_values['values'])

```

```
fore.head(10)
```

	start_date	end_date	updated_date	value
0	2015-06-18T00:00:00+02:00	2015-06-18T01:00:00+02:00	2015-06-17T20:15:00+02:00	0.00
1	2015-06-18T01:00:00+02:00	2015-06-18T02:00:00+02:00	2015-06-17T20:14:00+02:00	0.00
2	2015-06-18T02:00:00+02:00	2015-06-18T03:00:00+02:00	2015-06-17T20:13:00+02:00	0.00
3	2015-06-18T03:00:00+02:00	2015-06-18T04:00:00+02:00	2015-06-17T20:09:00+02:00	0.00
4	2015-06-18T04:00:00+02:00	2015-06-18T05:00:00+02:00	2015-06-17T20:09:00+02:00	0.00
5	2015-06-18T05:00:00+02:00	2015-06-18T06:00:00+02:00	2015-06-17T20:08:00+02:00	0.00
6	2015-06-18T06:00:00+02:00	2015-06-18T07:00:00+02:00	2015-06-17T20:37:00+02:00	4.06
7	2015-06-18T07:00:00+02:00	2015-06-18T08:00:00+02:00	2015-06-17T20:29:00+02:00	230.37
8	2015-06-18T08:00:00+02:00	2015-06-18T09:00:00+02:00	2015-06-17T20:29:00+02:00	697.28
9	2015-06-18T09:00:00+02:00	2015-06-18T10:00:00+02:00	2015-06-17T21:04:00+02:00	1406.28

```
typeE=['WIND_ONSHORE', 'WIND_OFFSHORE', 'SOLAR']
def getSol(e,s,type):
    headers = {
    'Authorization': f'Bearer {token}'}
    url = 'https://digital.iservices.rte-france.com/open_api/generation_forecast/v2/forecasts'
    params = {
        'production_type': typeE[type],
        'start_date': e,
        'end_date': s,
        'type':'D-1'}
    response = requests.get(url, headers=headers, params=params)
    forecast=response.json()
    print(forecast)
    forecast_values=forecast['forecasts'][0]
    fore=pd.DataFrame(forecast_values['values'])
    return fore
```

```
getSol('2015-06-01T00:00:00Z','2015-06-03T00:00:00Z',2)
```

{'forecasts': [{'start_date': '2015-06-02T00:00:00+02:00', 'end_date': '2015-06-03T00:00:00+02:00', 'type': 'D-1', 'production_type

	start_date	end_date	updated_date	value
0	2015-06-02T00:00:00+02:00	2015-06-02T01:00:00+02:00	2015-06-01T23:33:00+02:00	0.00
1	2015-06-02T01:00:00+02:00	2015-06-02T02:00:00+02:00	2015-06-01T23:03:00+02:00	0.00
2	2015-06-02T02:00:00+02:00	2015-06-02T03:00:00+02:00	2015-06-01T22:41:00+02:00	0.00
3	2015-06-02T03:00:00+02:00	2015-06-02T04:00:00+02:00	2015-06-01T22:36:00+02:00	0.00
4	2015-06-02T04:00:00+02:00	2015-06-02T05:00:00+02:00	2015-06-01T22:35:00+02:00	0.00
5	2015-06-02T05:00:00+02:00	2015-06-02T06:00:00+02:00	2015-06-01T22:34:00+02:00	0.00
6	2015-06-02T06:00:00+02:00	2015-06-02T07:00:00+02:00	2015-06-01T22:28:00+02:00	1.65
7	2015-06-02T07:00:00+02:00	2015-06-02T08:00:00+02:00	2015-06-01T22:27:00+02:00	187.40
8	2015-06-02T08:00:00+02:00	2015-06-02T09:00:00+02:00	2015-06-01T22:26:00+02:00	656.06
9	2015-06-02T09:00:00+02:00	2015-06-02T10:00:00+02:00	2015-06-01T23:32:00+02:00	1412.25
10	2015-06-02T10:00:00+02:00	2015-06-02T11:00:00+02:00	2015-06-01T23:31:00+02:00	2187.45
11	2015-06-02T11:00:00+02:00	2015-06-02T12:00:00+02:00	2015-06-01T23:25:00+02:00	2772.51
12	2015-06-02T12:00:00+02:00	2015-06-02T13:00:00+02:00	2015-06-01T23:24:00+02:00	3100.20
13	2015-06-02T13:00:00+02:00	2015-06-02T14:00:00+02:00	2015-06-01T23:22:00+02:00	3227.11
14	2015-06-02T14:00:00+02:00	2015-06-02T15:00:00+02:00	2015-06-01T23:14:00+02:00	3174.83
15	2015-06-02T15:00:00+02:00	2015-06-02T16:00:00+02:00	2015-06-01T23:13:00+02:00	3099.82
16	2015-06-02T16:00:00+02:00	2015-06-02T17:00:00+02:00	2015-06-01T23:12:00+02:00	2867.17
17	2015-06-02T17:00:00+02:00	2015-06-02T18:00:00+02:00	2015-06-01T23:06:00+02:00	2488.27
18	2015-06-02T18:00:00+02:00	2015-06-02T19:00:00+02:00	2015-06-01T23:04:00+02:00	1891.85
19	2015-06-02T19:00:00+02:00	2015-06-02T20:00:00+02:00	2015-06-01T22:58:00+02:00	1141.07
20	2015-06-02T20:00:00+02:00	2015-06-02T21:00:00+02:00	2015-06-01T22:57:00+02:00	490.33
21	2015-06-02T21:00:00+02:00	2015-06-02T22:00:00+02:00	2015-06-01T22:56:00+02:00	90.23
22	2015-06-02T22:00:00+02:00	2015-06-02T23:00:00+02:00	2015-06-01T22:44:00+02:00	0.00
23	2015-06-02T23:00:00+02:00	2015-06-03T00:00:00+02:00	2015-06-01T22:43:00+02:00	0.00

```
s='2017-12-31T00:00:00Z'
e='2018-01-02T00:00:00Z'

dates=pd.date_range(start="2018-01-01", end="2025-03-07", freq="D")
df_dates = pd.DataFrame({"Date": dates})
df_dates.head()
```

	Date
0	2018-01-01
1	2018-01-02
2	2018-01-03
3	2018-01-04
4	2018-01-05

```
dates[0]
e=dates[0].strftime('%Y-%m-%dT%H:%M:%SZ')
s=dates[2].strftime('%Y-%m-%dT%H:%M:%SZ')
print(e,s)
getSol(e,s,2)
```

```
2018-01-01T00:00:00Z 2018-01-03T00:00:00Z
{'forecasts': [{'start_date': '2018-01-02T00:00:00+01:00', 'end_date': '2018-01-03T00:00:00+01:00', 'type': 'D-1', 'production_type
```

	start_date	end_date	updated_date	value
0	2018-01-02T00:00:00+01:00	2018-01-02T01:00:00+01:00	2018-01-01T19:44:00+01:00	0.00
1	2018-01-02T01:00:00+01:00	2018-01-02T02:00:00+01:00	2018-01-01T19:33:00+01:00	0.00
2	2018-01-02T02:00:00+01:00	2018-01-02T03:00:00+01:00	2018-01-01T19:24:00+01:00	0.00
3	2018-01-02T03:00:00+01:00	2018-01-02T04:00:00+01:00	2018-01-01T19:23:00+01:00	0.00
4	2018-01-02T04:00:00+01:00	2018-01-02T05:00:00+01:00	2018-01-01T19:23:00+01:00	0.00
5	2018-01-02T05:00:00+01:00	2018-01-02T06:00:00+01:00	2018-01-01T19:19:00+01:00	0.00
6	2018-01-02T06:00:00+01:00	2018-01-02T07:00:00+01:00	2018-01-01T19:19:00+01:00	0.00
7	2018-01-02T07:00:00+01:00	2018-01-02T08:00:00+01:00	2018-01-01T19:18:00+01:00	0.00
8	2018-01-02T08:00:00+01:00	2018-01-02T09:00:00+01:00	2018-01-01T19:18:00+01:00	0.00
9	2018-01-02T09:00:00+01:00	2018-01-02T10:00:00+01:00	2018-01-01T19:44:00+01:00	116.90
10	2018-01-02T10:00:00+01:00	2018-01-02T11:00:00+01:00	2018-01-01T19:43:00+01:00	645.52
11	2018-01-02T11:00:00+01:00	2018-01-02T12:00:00+01:00	2018-01-01T19:43:00+01:00	1199.20
12	2018-01-02T12:00:00+01:00	2018-01-02T13:00:00+01:00	2018-01-01T19:39:00+01:00	1436.39
13	2018-01-02T13:00:00+01:00	2018-01-02T14:00:00+01:00	2018-01-01T19:39:00+01:00	1539.05
14	2018-01-02T14:00:00+01:00	2018-01-02T15:00:00+01:00	2018-01-01T19:38:00+01:00	1327.40
15	2018-01-02T15:00:00+01:00	2018-01-02T16:00:00+01:00	2018-01-01T19:38:00+01:00	896.37
16	2018-01-02T16:00:00+01:00	2018-01-02T17:00:00+01:00	2018-01-01T19:34:00+01:00	366.72
17	2018-01-02T17:00:00+01:00	2018-01-02T18:00:00+01:00	2018-01-01T19:33:00+01:00	20.22
18	2018-01-02T18:00:00+01:00	2018-01-02T19:00:00+01:00	2018-01-01T19:33:00+01:00	0.00
19	2018-01-02T19:00:00+01:00	2018-01-02T20:00:00+01:00	2018-01-01T19:29:00+01:00	0.00
20	2018-01-02T20:00:00+01:00	2018-01-02T21:00:00+01:00	2018-01-01T19:29:00+01:00	0.00
21	2018-01-02T21:00:00+01:00	2018-01-02T22:00:00+01:00	2018-01-01T19:28:00+01:00	0.00
22	2018-01-02T22:00:00+01:00	2018-01-02T23:00:00+01:00	2018-01-01T19:28:00+01:00	0.00
23	2018-01-02T23:00:00+01:00	2018-01-03T00:00:00+01:00	2018-01-01T19:24:00+01:00	0.00

```
def getData(ty):
    'ty le type denergie'
    L=[]
    for i in range(len(dates)-2):
        e=dates[i].strftime('%Y-%m-%dT%H:%M:%SZ')
        s=dates[i+2].strftime('%Y-%m-%dT%H:%M:%SZ')
        fore=getSol(e,s,ty)
        L.append(fore)

    concatenated_df = pd.concat(L, ignore_index=True)
    return concatenated_df

```

```
solar=getData(2)

```

```
{'forecasts': [{'start_date': '2018-01-02T00:00:00+01:00', 'end_date': '2018-01-03T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-03T00:00:00+01:00', 'end_date': '2018-01-04T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-04T00:00:00+01:00', 'end_date': '2018-01-05T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-05T00:00:00+01:00', 'end_date': '2018-01-06T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-06T00:00:00+01:00', 'end_date': '2018-01-07T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-07T00:00:00+01:00', 'end_date': '2018-01-08T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-08T00:00:00+01:00', 'end_date': '2018-01-09T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-09T00:00:00+01:00', 'end_date': '2018-01-10T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-10T00:00:00+01:00', 'end_date': '2018-01-11T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-11T00:00:00+01:00', 'end_date': '2018-01-12T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-12T00:00:00+01:00', 'end_date': '2018-01-13T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-13T00:00:00+01:00', 'end_date': '2018-01-14T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-14T00:00:00+01:00', 'end_date': '2018-01-15T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-15T00:00:00+01:00', 'end_date': '2018-01-16T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-16T00:00:00+01:00', 'end_date': '2018-01-17T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-17T00:00:00+01:00', 'end_date': '2018-01-18T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-18T00:00:00+01:00', 'end_date': '2018-01-19T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-19T00:00:00+01:00', 'end_date': '2018-01-20T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-20T00:00:00+01:00', 'end_date': '2018-01-21T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-21T00:00:00+01:00', 'end_date': '2018-01-22T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-22T00:00:00+01:00', 'end_date': '2018-01-23T00:00:00+01:00', 'type': 'D-1', 'production_ty
```

{'forecasts': [{'start_date': '2018-01-23T00:00:00+01:00', 'end_date': '2018-01-24T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-24T00:00:00+01:00', 'end_date': '2018-01-25T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-25T00:00:00+01:00', 'end_date': '2018-01-26T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-26T00:00:00+01:00', 'end_date': '2018-01-27T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-27T00:00:00+01:00', 'end_date': '2018-01-28T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-28T00:00:00+01:00', 'end_date': '2018-01-29T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-29T00:00:00+01:00', 'end_date': '2018-01-30T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-30T00:00:00+01:00', 'end_date': '2018-01-31T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-31T00:00:00+01:00', 'end_date': '2018-02-01T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-01T00:00:00+01:00', 'end_date': '2018-02-02T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-02T00:00:00+01:00', 'end_date': '2018-02-03T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-03T00:00:00+01:00', 'end_date': '2018-02-04T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-04T00:00:00+01:00', 'end_date': '2018-02-05T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-05T00:00:00+01:00', 'end_date': '2018-02-06T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-06T00:00:00+01:00', 'end_date': '2018-02-07T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-07T00:00:00+01:00', 'end_date': '2018-02-08T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-08T00:00:00+01:00', 'end_date': '2018-02-09T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-09T00:00:00+01:00', 'end_date': '2018-02-10T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-10T00:00:00+01:00', 'end_date': '2018-02-11T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-11T00:00:00+01:00', 'end_date': '2018-02-12T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-12T00:00:00+01:00', 'end_date': '2018-02-13T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-13T00:00:00+01:00', 'end_date': '2018-02-14T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-14T00:00:00+01:00', 'end_date': '2018-02-15T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-15T00:00:00+01:00', 'end_date': '2018-02-16T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-16T00:00:00+01:00', 'end_date': '2018-02-17T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-17T00:00:00+01:00', 'end_date': '2018-02-18T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-18T00:00:00+01:00', 'end_date': '2018-02-19T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-19T00:00:00+01:00', 'end_date': '2018-02-20T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-20T00:00:00+01:00', 'end_date': '2018-02-21T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-21T00:00:00+01:00', 'end_date': '2018-02-22T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-22T00:00:00+01:00', 'end_date': '2018-02-23T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-23T00:00:00+01:00', 'end_date': '2018-02-24T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-24T00:00:00+01:00', 'end_date': '2018-02-25T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-25T00:00:00+01:00', 'end_date': '2018-02-26T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-26T00:00:00+01:00', 'end_date': '2018-02-27T00:00:00+01:00', 'type': 'D-1', 'production_ty

```
1 solar.rename(columns={'value':'Solar'}, inplace=True)
2
```

```
1 windOn=getData(0)
2
3
```

{'forecasts': [{'start_date': '2018-01-02T00:00:00+01:00', 'end_date': '2018-01-03T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-03T00:00:00+01:00', 'end_date': '2018-01-04T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-04T00:00:00+01:00', 'end_date': '2018-01-05T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-05T00:00:00+01:00', 'end_date': '2018-01-06T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-06T00:00:00+01:00', 'end_date': '2018-01-07T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-07T00:00:00+01:00', 'end_date': '2018-01-08T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-08T00:00:00+01:00', 'end_date': '2018-01-09T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-09T00:00:00+01:00', 'end_date': '2018-01-10T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-10T00:00:00+01:00', 'end_date': '2018-01-11T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-11T00:00:00+01:00', 'end_date': '2018-01-12T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-12T00:00:00+01:00', 'end_date': '2018-01-13T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-13T00:00:00+01:00', 'end_date': '2018-01-14T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-14T00:00:00+01:00', 'end_date': '2018-01-15T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-15T00:00:00+01:00', 'end_date': '2018-01-16T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-16T00:00:00+01:00', 'end_date': '2018-01-17T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-17T00:00:00+01:00', 'end_date': '2018-01-18T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-18T00:00:00+01:00', 'end_date': '2018-01-19T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-19T00:00:00+01:00', 'end_date': '2018-01-20T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-20T00:00:00+01:00', 'end_date': '2018-01-21T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-21T00:00:00+01:00', 'end_date': '2018-01-22T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-22T00:00:00+01:00', 'end_date': '2018-01-23T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-23T00:00:00+01:00', 'end_date': '2018-01-24T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-24T00:00:00+01:00', 'end_date': '2018-01-25T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-25T00:00:00+01:00', 'end_date': '2018-01-26T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-26T00:00:00+01:00', 'end_date': '2018-01-27T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-27T00:00:00+01:00', 'end_date': '2018-01-28T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-28T00:00:00+01:00', 'end_date': '2018-01-29T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-29T00:00:00+01:00', 'end_date': '2018-01-30T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-30T00:00:00+01:00', 'end_date': '2018-01-31T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-31T00:00:00+01:00', 'end_date': '2018-02-01T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-01T00:00:00+01:00', 'end_date': '2018-02-02T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-02T00:00:00+01:00', 'end_date': '2018-02-03T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-03T00:00:00+01:00', 'end_date': '2018-02-04T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-04T00:00:00+01:00', 'end_date': '2018-02-05T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-05T00:00:00+01:00', 'end_date': '2018-02-06T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-06T00:00:00+01:00', 'end_date': '2018-02-07T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-07T00:00:00+01:00', 'end_date': '2018-02-08T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-08T00:00:00+01:00', 'end_date': '2018-02-09T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-09T00:00:00+01:00', 'end_date': '2018-02-10T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-10T00:00:00+01:00', 'end_date': '2018-02-11T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-11T00:00:00+01:00', 'end_date': '2018-02-12T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-12T00:00:00+01:00', 'end_date': '2018-02-13T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-13T00:00:00+01:00', 'end_date': '2018-02-14T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-14T00:00:00+01:00', 'end_date': '2018-02-15T00:00:00+01:00', 'type': 'D-1', 'production_ty

{'forecasts': [{'start_date': '2018-02-15T00:00:00+01:00', 'end_date': '2018-02-16T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-16T00:00:00+01:00', 'end_date': '2018-02-17T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-17T00:00:00+01:00', 'end_date': '2018-02-18T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-18T00:00:00+01:00', 'end_date': '2018-02-19T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-19T00:00:00+01:00', 'end_date': '2018-02-20T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-20T00:00:00+01:00', 'end_date': '2018-02-21T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-21T00:00:00+01:00', 'end_date': '2018-02-22T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-22T00:00:00+01:00', 'end_date': '2018-02-23T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-23T00:00:00+01:00', 'end_date': '2018-02-24T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-24T00:00:00+01:00', 'end_date': '2018-02-25T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-25T00:00:00+01:00', 'end_date': '2018-02-26T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-26T00:00:00+01:00', 'end_date': '2018-02-27T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-27T00:00:00+01:00', 'end_date': '2018-02-28T00:00:00+01:00', 'type': 'D-1', 'production_ty

```
windOn.rename(columns={'value':'windOn'}, inplace=True)

windOn.head()
```

	start_date	end_date	updated_date	windOn	load_factor
0	2018-01-02T00:00:00+01:00	2018-01-02T01:00:00+01:00	2018-01-01T17:26:00+01:00	8393.0	67
1	2018-01-02T01:00:00+01:00	2018-01-02T02:00:00+01:00	2018-01-01T17:26:00+01:00	8186.0	65
2	2018-01-02T02:00:00+01:00	2018-01-02T03:00:00+01:00	2018-01-01T17:26:00+01:00	7589.0	60
3	2018-01-02T03:00:00+01:00	2018-01-02T04:00:00+01:00	2018-01-01T17:26:00+01:00	6980.0	56
4	2018-01-02T04:00:00+01:00	2018-01-02T05:00:00+01:00	2018-01-01T17:26:00+01:00	6368.0	51

```
windOff=getData(1)

```

{'forecasts': [{'start_date': '2018-01-02T00:00:00+01:00', 'end_date': '2018-01-03T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-03T00:00:00+01:00', 'end_date': '2018-01-04T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-04T00:00:00+01:00', 'end_date': '2018-01-05T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-05T00:00:00+01:00', 'end_date': '2018-01-06T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-06T00:00:00+01:00', 'end_date': '2018-01-07T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-07T00:00:00+01:00', 'end_date': '2018-01-08T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-08T00:00:00+01:00', 'end_date': '2018-01-09T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-09T00:00:00+01:00', 'end_date': '2018-01-10T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-10T00:00:00+01:00', 'end_date': '2018-01-11T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-11T00:00:00+01:00', 'end_date': '2018-01-12T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-12T00:00:00+01:00', 'end_date': '2018-01-13T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-13T00:00:00+01:00', 'end_date': '2018-01-14T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-14T00:00:00+01:00', 'end_date': '2018-01-15T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-15T00:00:00+01:00', 'end_date': '2018-01-16T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-16T00:00:00+01:00', 'end_date': '2018-01-17T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-17T00:00:00+01:00', 'end_date': '2018-01-18T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-18T00:00:00+01:00', 'end_date': '2018-01-19T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-19T00:00:00+01:00', 'end_date': '2018-01-20T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-20T00:00:00+01:00', 'end_date': '2018-01-21T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-21T00:00:00+01:00', 'end_date': '2018-01-22T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-22T00:00:00+01:00', 'end_date': '2018-01-23T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-23T00:00:00+01:00', 'end_date': '2018-01-24T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-24T00:00:00+01:00', 'end_date': '2018-01-25T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-25T00:00:00+01:00', 'end_date': '2018-01-26T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-26T00:00:00+01:00', 'end_date': '2018-01-27T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-27T00:00:00+01:00', 'end_date': '2018-01-28T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-28T00:00:00+01:00', 'end_date': '2018-01-29T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-29T00:00:00+01:00', 'end_date': '2018-01-30T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-30T00:00:00+01:00', 'end_date': '2018-01-31T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-01-31T00:00:00+01:00', 'end_date': '2018-02-01T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-01T00:00:00+01:00', 'end_date': '2018-02-02T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-02T00:00:00+01:00', 'end_date': '2018-02-03T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-03T00:00:00+01:00', 'end_date': '2018-02-04T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-04T00:00:00+01:00', 'end_date': '2018-02-05T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-05T00:00:00+01:00', 'end_date': '2018-02-06T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-06T00:00:00+01:00', 'end_date': '2018-02-07T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-07T00:00:00+01:00', 'end_date': '2018-02-08T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-08T00:00:00+01:00', 'end_date': '2018-02-09T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-09T00:00:00+01:00', 'end_date': '2018-02-10T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-10T00:00:00+01:00', 'end_date': '2018-02-11T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-11T00:00:00+01:00', 'end_date': '2018-02-12T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-12T00:00:00+01:00', 'end_date': '2018-02-13T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-13T00:00:00+01:00', 'end_date': '2018-02-14T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-14T00:00:00+01:00', 'end_date': '2018-02-15T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-15T00:00:00+01:00', 'end_date': '2018-02-16T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-16T00:00:00+01:00', 'end_date': '2018-02-17T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-17T00:00:00+01:00', 'end_date': '2018-02-18T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-18T00:00:00+01:00', 'end_date': '2018-02-19T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-19T00:00:00+01:00', 'end_date': '2018-02-20T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-20T00:00:00+01:00', 'end_date': '2018-02-21T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-21T00:00:00+01:00', 'end_date': '2018-02-22T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-22T00:00:00+01:00', 'end_date': '2018-02-23T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-23T00:00:00+01:00', 'end_date': '2018-02-24T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-24T00:00:00+01:00', 'end_date': '2018-02-25T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-25T00:00:00+01:00', 'end_date': '2018-02-26T00:00:00+01:00', 'type': 'D-1', 'production_ty

```
{'forecasts': [{'start_date': '2018-02-26T00:00:00+01:00', 'end_date': '2018-02-27T00:00:00+01:00', 'type': 'D-1', 'production_ty
{'forecasts': [{'start_date': '2018-02-27T00:00:00+01:00', 'end_date': '2018-02-28T00:00:00+01:00', 'type': 'D-1', 'production_ty
```

```
windOff.rename(columns={'value':'windOff'}, inplace=True)

windOff.head()
```

	start_date	end_date	updated_date	windOff	load_factor
0	2023-08-04T00:00:00+02:00	2023-08-04T01:00:00+02:00	2023-08-03T17:06:00+02:00	290.0	0
1	2023-08-04T01:00:00+02:00	2023-08-04T02:00:00+02:00	2023-08-03T17:06:00+02:00	279.0	0
2	2023-08-04T02:00:00+02:00	2023-08-04T03:00:00+02:00	2023-08-03T17:06:00+02:00	278.5	0
3	2023-08-04T03:00:00+02:00	2023-08-04T04:00:00+02:00	2023-08-03T17:06:00+02:00	280.0	0
4	2023-08-04T04:00:00+02:00	2023-08-04T05:00:00+02:00	2023-08-03T17:06:00+02:00	286.0	0

```
windOff['start_date']=windOff['start_date'].str.replace('+02:00','+01:00')
```

```
windOff.head()
```

	start_date	end_date	updated_date	windOff	load_factor
0	2023-08-04T00:00:00+01:00	2023-08-04T01:00:00+02:00	2023-08-03T17:06:00+02:00	290.0	0
1	2023-08-04T01:00:00+01:00	2023-08-04T02:00:00+02:00	2023-08-03T17:06:00+02:00	279.0	0
2	2023-08-04T02:00:00+01:00	2023-08-04T03:00:00+02:00	2023-08-03T17:06:00+02:00	278.5	0
3	2023-08-04T03:00:00+01:00	2023-08-04T04:00:00+02:00	2023-08-03T17:06:00+02:00	280.0	0
4	2023-08-04T04:00:00+01:00	2023-08-04T05:00:00+02:00	2023-08-03T17:06:00+02:00	286.0	0

```
sorted_histo.head()
```

	Date	Heure	Prix	date	start_date
2	2018-01-01 00:00:00	01:00:00	14.43	2018-01-01	2018-01-01T01:00:00+01:00
102615	2018-01-01 00:00:00	02:00:00	10.26	2018-01-01	2018-01-01T02:00:00+01:00
205228	2018-01-01 00:00:00	03:00:00	7.48	2018-01-01	2018-01-01T03:00:00+01:00
307841	2018-01-01 00:00:00	04:00:00	4.52	2018-01-01	2018-01-01T04:00:00+01:00
410454	2018-01-01 00:00:00	05:00:00	4.24	2018-01-01	2018-01-01T05:00:00+01:00

```
df1 = pd.merge(solar, windOn, on='start_date', how='left')

```

```
df1['start_date']=df1['start_date'].str.replace('+02:00','+01:00')
```

```
df2 = pd.merge(df1,sorted_histo.dropna(), on='start_date', how='outer')
```

```
df2=df2.dropna()
```

```
df3 = pd.merge(df2, windOff, on='start_date', how='left')
```

```
df=df3[['start_date', 'Solar', 'windOn', 'windOff', 'Prix']]
df.head(10)
```

	start_date	Solar	windOn	windOff	Prix
0	2018-01-02T00:00:00+01:00	0.0	8393.0	NaN	13.87
1	2018-01-02T01:00:00+01:00	0.0	8186.0	NaN	12.14
2	2018-01-02T02:00:00+01:00	0.0	7589.0	NaN	5.48
3	2018-01-02T03:00:00+01:00	0.0	6980.0	NaN	5
4	2018-01-02T04:00:00+01:00	0.0	6368.0	NaN	0.99
5	2018-01-02T05:00:00+01:00	0.0	5787.0	NaN	-0.25
6	2018-01-02T06:00:00+01:00	0.0	5206.0	NaN	3.87
7	2018-01-02T07:00:00+01:00	0.0	4631.0	NaN	12.25
8	2018-01-02T08:00:00+01:00	0.0	4406.0	NaN	19.42
9	2018-01-02T09:00:00+01:00	116.9	4180.0	NaN	22.57

```
df['Prix'] = df['Prix'].astype(float)
```

```
C:\Users\adamh\AppData\Local\Temp\ipykernel_3544\1829466283.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
```

```
df = df.fillna(0)

```

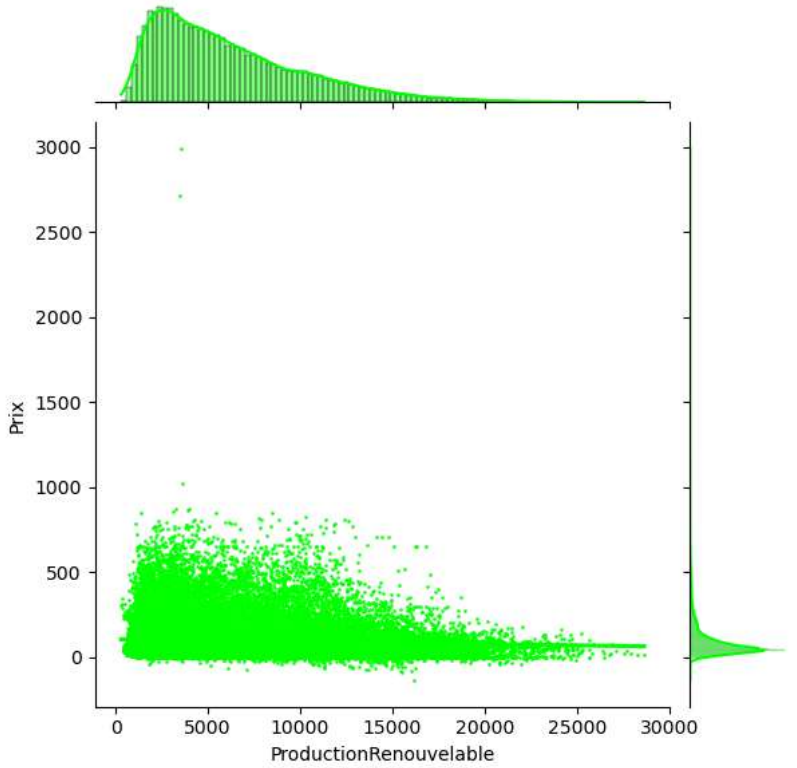
```
df['ProductionRenouvelable']=df['Solar']+df['windOn']+df['windOff']
```

```
df.head()
```

	start_date	Solar	windOn	windOff	Prix	ProductionRenouvelable
0	2018-01-02T00:00:00+01:00	0.0	8393.0	0.0	13.87	8393.0
1	2018-01-02T01:00:00+01:00	0.0	8186.0	0.0	12.14	8186.0
2	2018-01-02T02:00:00+01:00	0.0	7589.0	0.0	5.48	7589.0
3	2018-01-02T03:00:00+01:00	0.0	6980.0	0.0	5.00	6980.0
4	2018-01-02T04:00:00+01:00	0.0	6368.0	0.0	0.99	6368.0

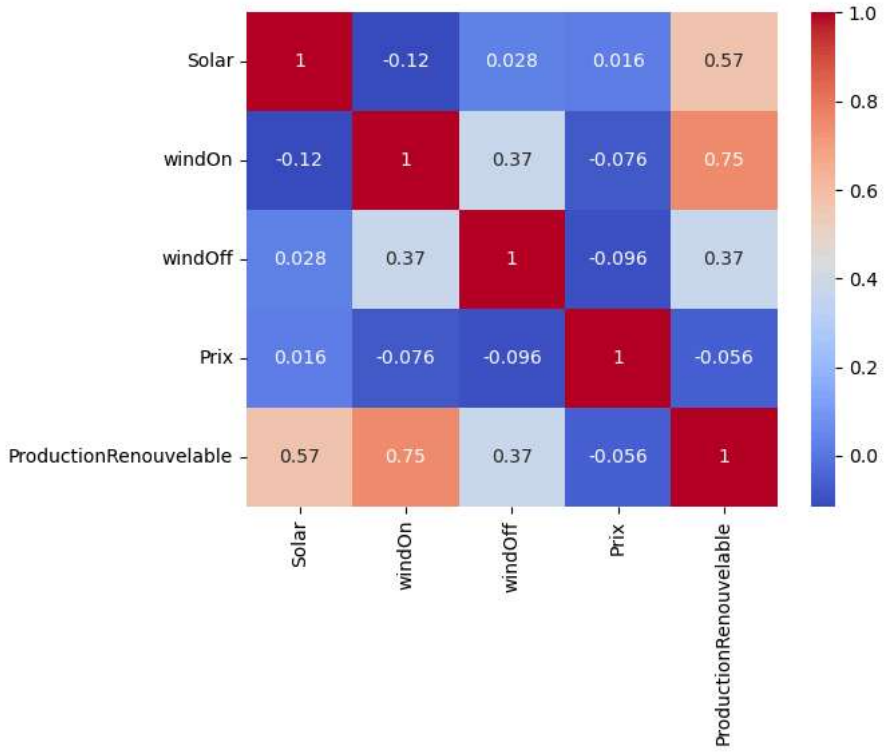
```
sns.jointplot(x="ProductionRenouvelable", y="Prix", data=df, kind="reg",color='lime', scatter_kws={'s': 1})  # petite taille

```

```
<seaborn.axisgrid.JointGrid at 0x1e516ad9f40>
```




```
sns.heatmap(df.iloc[:,1:].corr(), cmap="coolwarm", annot=True)

```

```
<Axes: >
```




```
df['date'] = pd.to_datetime(df['start_date'], errors='coerce')

```

```
px.scatter(df, x="date", y="Prix")
```

```
df
```

	start_date	Solar	windOn	windOff	Prix	ProductionRenouvelable	date
0	2018-01-02T00:00:00+01:00	0.0	8393.00	0.00	13.87	8393.00	2018-01-02 00:00:00+01:00
1	2018-01-02T01:00:00+01:00	0.0	8186.00	0.00	12.14	8186.00	2018-01-02 01:00:00+01:00
2	2018-01-02T02:00:00+01:00	0.0	7589.00	0.00	5.48	7589.00	2018-01-02 02:00:00+01:00
3	2018-01-02T03:00:00+01:00	0.0	6980.00	0.00	5.00	6980.00	2018-01-02 03:00:00+01:00
4	2018-01-02T04:00:00+01:00	0.0	6368.00	0.00	0.99	6368.00	2018-01-02 04:00:00+01:00
...	...	...	...	...	...	...	...
61213	2025-03-06T19:00:00+01:00	0.0	8242.30	918.00	83.08	9160.30	2025-03-06 19:00:00+01:00
61214	2025-03-06T20:00:00+01:00	0.0	9278.01	968.50	99.65	10246.51	2025-03-06 20:00:00+01:00
61215	2025-03-06T21:00:00+01:00	0.0	9717.69	995.75	74.89	10713.44	2025-03-06 21:00:00+01:00
61216	2025-03-06T22:00:00+01:00	0.0	9936.22	1012.50	50.00	10948.72	2025-03-06 22:00:00+01:00
61217	2025-03-06T23:00:00+01:00	0.0	9803.47	1024.00	59.20	10827.47	2025-03-06 23:00:00+01:00

61218 rows × 7 columns

```
df
```

	start_date	Solar	windOn	windOff	Prix	ProductionRenouvelable	date
0	2018-01-02T00:00:00+01:00	0.0	8393.00	0.00	13.87	8393.00	2018-01-02 00:00:00+01:00
1	2018-01-02T01:00:00+01:00	0.0	8186.00	0.00	12.14	8186.00	2018-01-02 01:00:00+01:00
2	2018-01-02T02:00:00+01:00	0.0	7589.00	0.00	5.48	7589.00	2018-01-02 02:00:00+01:00
3	2018-01-02T03:00:00+01:00	0.0	6980.00	0.00	5.00	6980.00	2018-01-02 03:00:00+01:00
4	2018-01-02T04:00:00+01:00	0.0	6368.00	0.00	0.99	6368.00	2018-01-02 04:00:00+01:00
...	...	...	...	...	...	...	...
61213	2025-03-06T19:00:00+01:00	0.0	8242.30	918.00	83.08	9160.30	2025-03-06 19:00:00+01:00
61214	2025-03-06T20:00:00+01:00	0.0	9278.01	968.50	99.65	10246.51	2025-03-06 20:00:00+01:00
61215	2025-03-06T21:00:00+01:00	0.0	9717.69	995.75	74.89	10713.44	2025-03-06 21:00:00+01:00
61216	2025-03-06T22:00:00+01:00	0.0	9936.22	1012.50	50.00	10948.72	2025-03-06 22:00:00+01:00
61217	2025-03-06T23:00:00+01:00	0.0	9803.47	1024.00	59.20	10827.47	2025-03-06 23:00:00+01:00

61218 rows × 7 columns

```
type(df['start_date'].iloc[2])
df['start_date'] = pd.to_datetime(df['start_date'], errors='coerce')


df['Year'] = df['start_date'].dt.year
df['Month'] = df['start_date'].dt.month
df['Day'] = df['start_date'].dt.day
```

```
df_grouped_day = df.groupby(['Year', 'Month', 'Day'])[['Prix','ProductionRenouvelable']].agg(['max', 'min', 'mean','sum']).reset_in
df_grouped_day.head()
```

	Year	Month	Day	Prix				ProductionRenouvelable			
				max	min	mean	sum	max	min	mean	sum
0	2018	1	2	36.67	-0.25	16.522083	396.53	8393.00	4296.9	6396.573750	153517.77
1	2018	1	3	50.08	2.11	30.698750	736.77	13396.72	8006.0	10933.846250	262412.31
2	2018	1	4	57.42	0.55	34.010417	816.25	12551.91	5995.0	8797.032917	211128.79
3	2018	1	5	49.98	24.34	36.208750	869.01	6655.00	2522.0	4633.367083	111200.81
4	2018	1	6	34.34	12.17	22.411250	537.87	3115.00	1210.0	1912.567500	45901.62

```
df_grouped_day.columns = ['_'.join(col).strip() if isinstance(col, tuple) else col for col in df_grouped_day.columns.values]
df_grouped_day.head()
```

	Year_	Month_	Day_	Prix_max	Prix_min	Prix_mean	Prix_sum	ProductionRenouvelable_max	ProductionRenouvelable_min	ProductionR
0	2018	1	2	36.67	-0.25	16.522083	396.53	8393.00	4296.9	
1	2018	1	3	50.08	2.11	30.698750	736.77	13396.72	8006.0	
2	2018	1	4	57.42	0.55	34.010417	816.25	12551.91	5995.0	
3	2018	1	5	49.98	24.34	36.208750	869.01	6655.00	2522.0	
4	2018	1	6	34.34	12.17	22.411250	537.87	3115.00	1210.0	

```
df_grouped_day['Spread']=df_grouped_day['Prix_max']-df_grouped_day['Prix_min']
```

```
df_grouped_day['Year_']=df_grouped_day['Year_'].astype(int)
```

```
df_grouped_day.head()
```

	Year_	Month_	Day_	Prix_max	Prix_min	Prix_mean	Prix_sum	ProductionRenouvelable_max	ProductionRenouvelable_min	ProductionR
0	2018	1	2	36.67	-0.25	16.522083	396.53	8393.00	4296.9	
1	2018	1	3	50.08	2.11	30.698750	736.77	13396.72	8006.0	
2	2018	1	4	57.42	0.55	34.010417	816.25	12551.91	5995.0	
3	2018	1	5	49.98	24.34	36.208750	869.01	6655.00	2522.0	
4	2018	1	6	34.34	12.17	22.411250	537.87	3115.00	1210.0	

```
df_grouped_day['Date'] = pd.to_datetime(
    df_grouped_day.rename(columns={'Year_': 'year', 'Month_': 'month', 'Day_': 'day'})[['year', 'month', 'day']]
)

```

```
import plotly.express as px

fig = px.scatter(
    df,
    x="start_date",
    y="Prix",
    facet_col="Year",
    facet_col_wrap=2  # pour mettre 2 graphiques par ligne, optionnel
)
fig.update_traces(marker=dict(size=1))  # ou 2, ou même 1

# Pour que chaque facette ait son propre axe X (date adaptée)
fig.update_xaxes(matches=None)
fig.update_yaxes(matches=None)

for axis in fig.layout:
    if axis.startswith("xaxis"):
        fig.layout[axis].tickformat = "%b"

fig.update_layout(
    title="Prix par mois pour chaque année",
    height=600
)
fig.show()

```

```
import plotly.express as px


fig = px.scatter(
    df,
    x="start_date",
    y="Prix",
    facet_col="Year",
    facet_col_wrap=2
)
fig.update_traces(marker=dict(size=1))

# Axes personnalisés
fig.update_xaxes(matches=None)
fig.update_yaxes(matches=None)

# Appliquer le format mois ET forcer l'affichage des axes
```

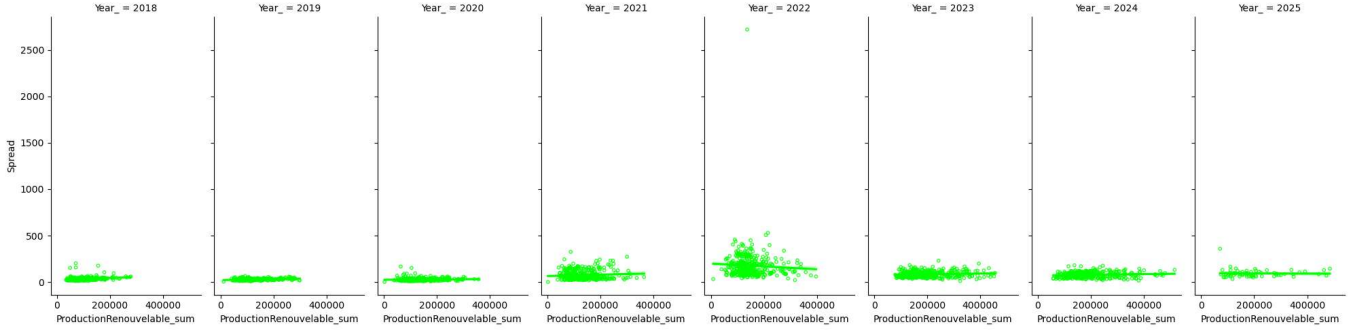
```
for axis in fig.layout:
    if axis.startswith("xaxis"):
        fig.layout[axis].tickformat = "%b"
        fig.layout[axis].showticklabels = True
    if axis.startswith("yaxis"):
        fig.layout[axis].showticklabels = True

fig.update_layout(
    title="Prix par mois pour chaque année",
    height=600
)

fig.show()

```

```
import plotly.express as px

fig = px.scatter(
    df_grouped_day,
    x="Date",
    y="Spread",
    trendline="ols",
    facet_col="Year_",
    facet_col_wrap=2  # pour mettre 2 graphiques par ligne, optionnel
)

# Pour que chaque facette ait son propre axe X (date adaptée)
fig.update_xaxes(matches=None)
fig.update_yaxes(matches=None)

fig.update_layout(title="Spread vs Date par année", height=600)
fig.show()

```

```
px.scatter(df, x="start_date", y="Prix")
```

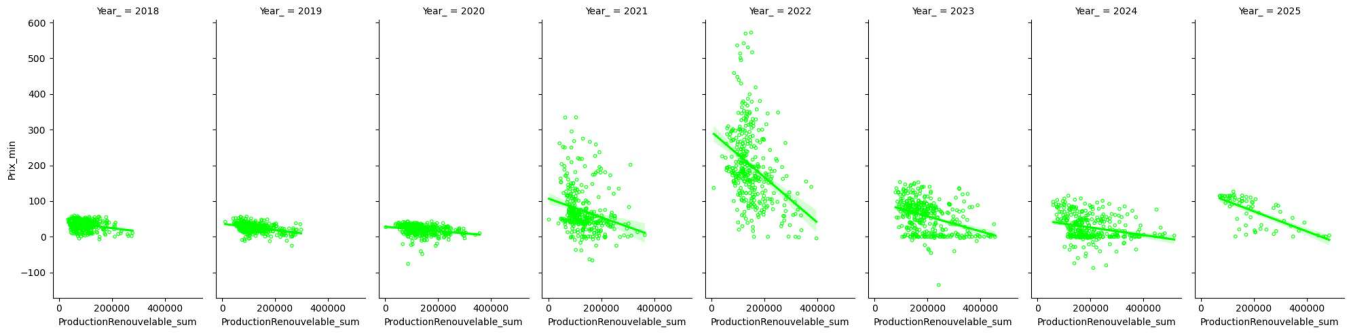
```
sns.lmplot(
    x="ProductionRenouvelable_sum",
    y="Spread",
    data=df_grouped_day,
    col="Year_",  # Facettage par année
    scatter_kws={
        's': 10,              # taille des points (ajustable)
        'facecolors': 'none',  # ronds vides
        'edgecolors': 'lime'   # bordure verte
    },
    line_kws={
        'color': 'lime'        # couleur de la régression
    },
    height=5,
    aspect=0.5
)

```

```
<seaborn.axisgrid.FacetGrid at 0x1e518a7b620>
```



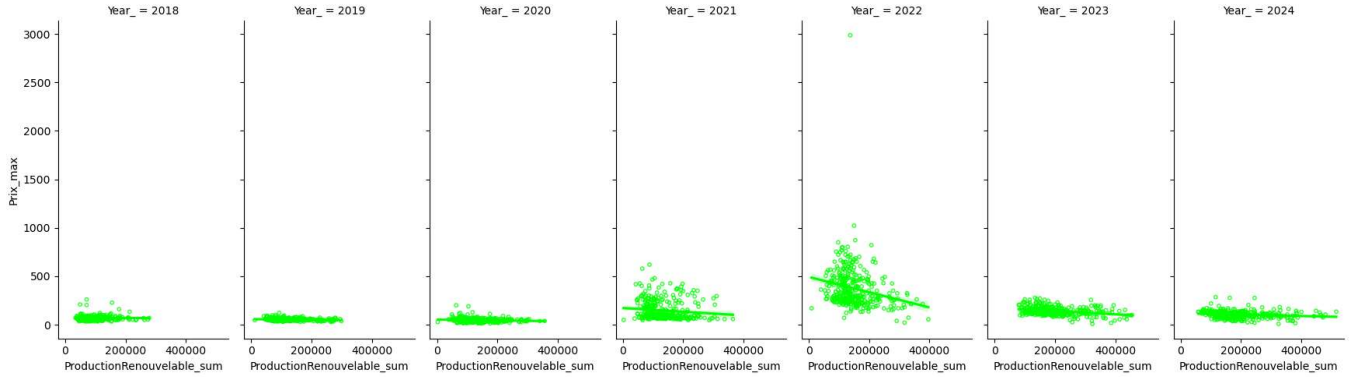
```
sns.lmplot(
    x="ProductionRenouvelable_sum",
    y="Prix_min",
    data=df_grouped_day,
    col="Year_",  # Facettage par année
    scatter_kws={
        's': 10,              # taille des points (ajustable)
        'facecolors': 'none',  # ronds vides
        'edgecolors': 'lime'   # bordure verte
    },
    line_kws={
        'color': 'lime'        # couleur de la régression
    },
    height=5,
    aspect=0.5
)
```

```
<seaborn.axisgrid.FacetGrid at 0x1e528484800>
```

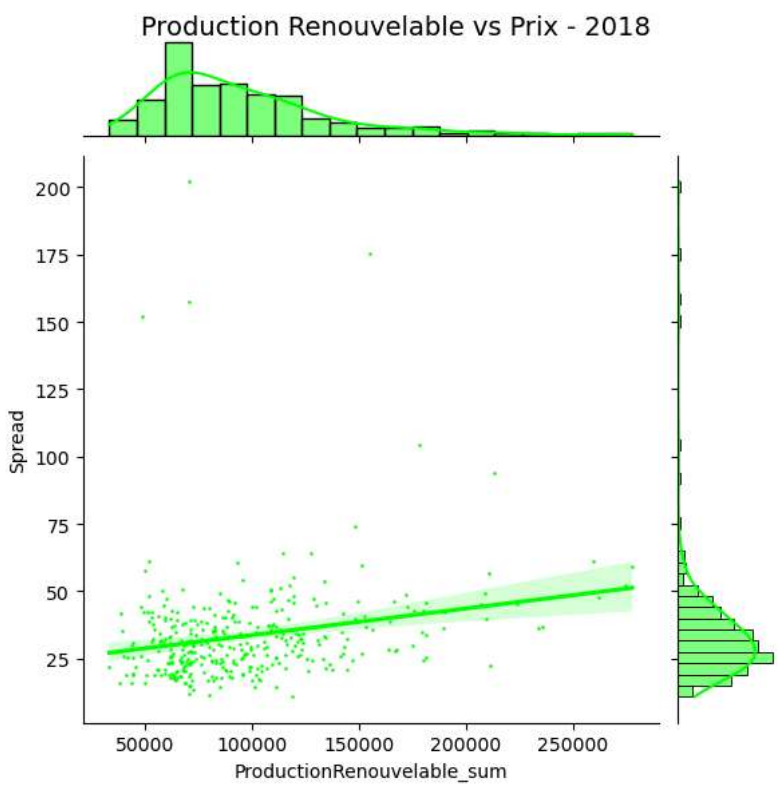


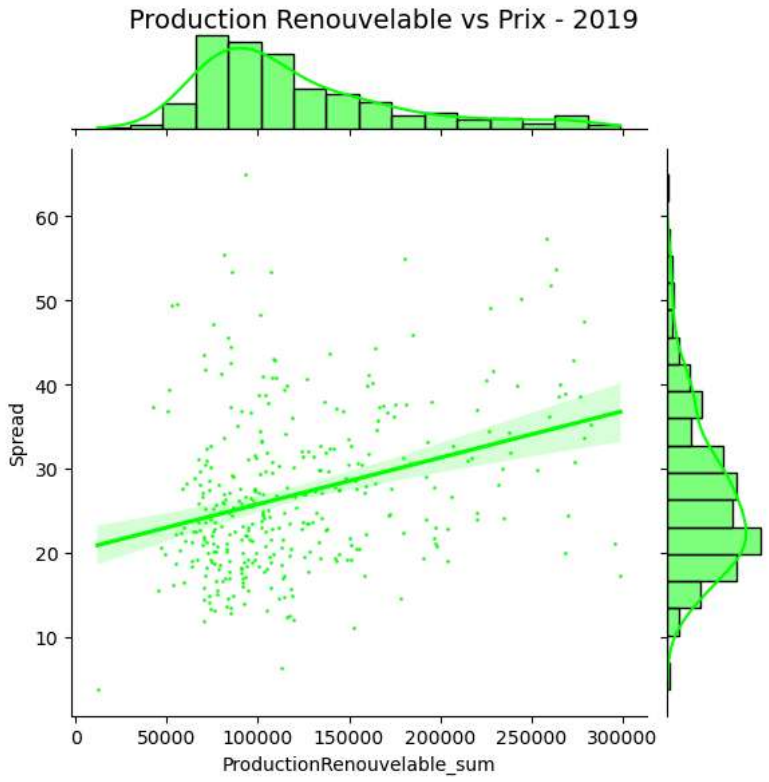

```
df_grouped_day = df_grouped_day[df_grouped_day['Year_'] != 2025]

sns.lmplot(
    x="ProductionRenouvelable_sum",
    y="Prix_max",
    data=df_grouped_day,
    col="Year_",  # Facettage par année
    scatter_kws={
        's': 10,              # taille des points (ajustable)
        'facecolors': 'none',  # ronds vides
        'edgecolors': 'lime'   # bordure verte
    },
    line_kws={
        'color': 'lime'        # couleur de la régression
    },
    height=5,
    aspect=0.5
)
```

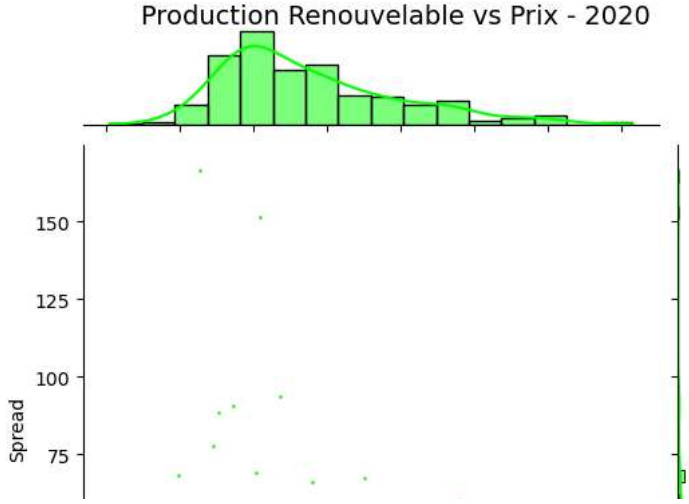
```
<seaborn.axisgrid.FacetGrid at 0x1e518b08560>
```



```
for year in sorted(df_grouped_day['Year_'].unique()):
    df_year = df_grouped_day[df_grouped_day['Year_'] == year]

    # Créer le jointplot
    g = sns.jointplot(
        x="ProductionRenouvelable_sum",
        y="Spread",
        data=df_year,
        kind="reg",
        color='lime',
        scatter_kws={'s': 1}  # petite taille des points
    )

    # Ajouter un titre
    plt.suptitle(f'Production Renouvelable vs Prix - {year}', fontsize=14)
    plt.subplots_adjust(top=0.95)  # Pour ne pas couper le titre
    plt.show()
```


Production Renouvelable vs Prix - 2018
Spread
ProductionRenouvelable_sum
200
175
150
125
100
75
50
25
50000
100000
150000
200000
250000


Production Renouvelable vs Prix - 2019
Spread
ProductionRenouvelable_sum
60
50
40
30
20
10
0
50000
100000
150000
200000
250000
300000


Production Renouvelable vs Prix - 2020
Spread
150
125
100
75