Data verwerking

een praktijkvoorbeeld

Situatie

Wens: mogelijkheid voor maand/jaar rapportages (standaard en uitgebreid) met betrekking tot een PV installatie (een “plant”)

Data: in MySQL (table “yields”)

Uit te werken:

1. Keuze van rapport (standaard, uitgebreid, maand, jaar)
2. Data selectie (SQL-query)
3. Pandas DataFrame(s)?
4. Grafieken
5. PDF rapportage met tekst en grafieken

Onthou: gebruik tooling waarvoor deze geschikt zijn: (ga bv. niet met dataframe.plot een pdf maken…)

Specificaties

- Jaar of maand rapportages
  - Standaard rapport: weergave van alleen opbrengst (“yield”)
  - Uitgebreid rapport: weergave van opbrengst, temperatuur, instraling en Performance Ratio (PR).

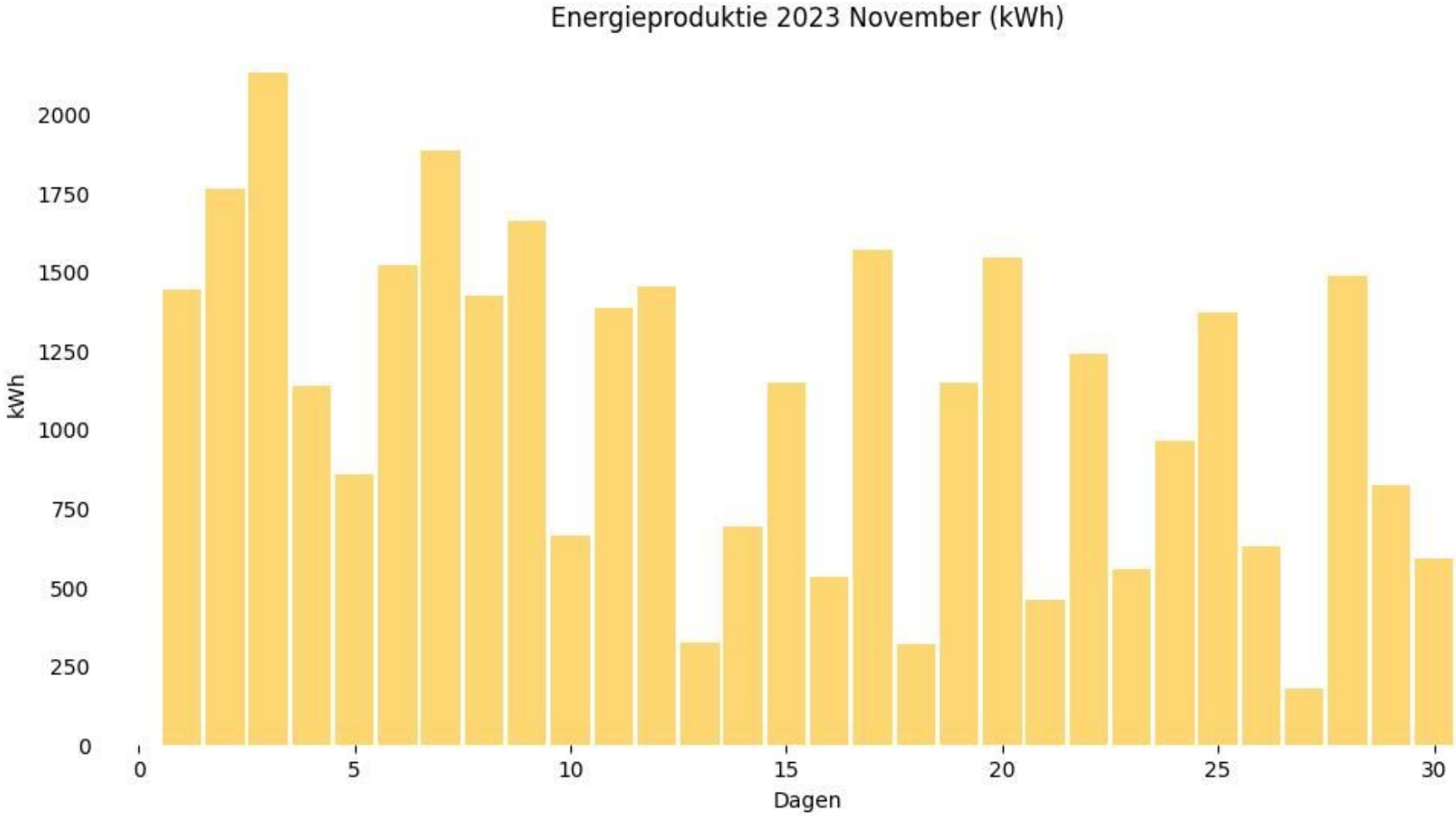
Bij selectie van meerdere jaren de data per plant in 1 plaatje weergeven (makkelijk om zo bv. 2 jaren te vergelijken)

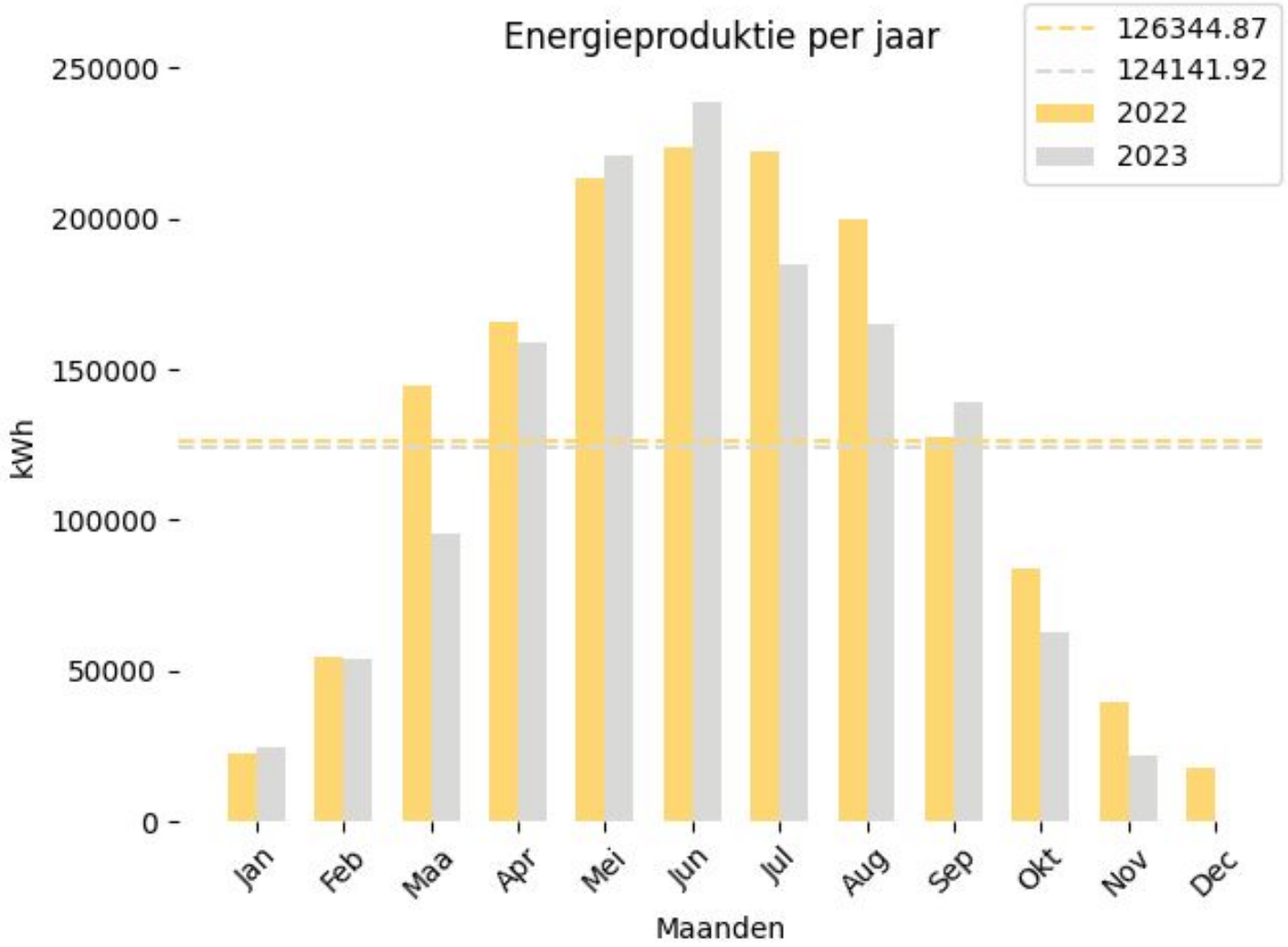
De plaatjes zijn dan een jaar-overzicht met maanden of maand-overzichten met dagen (waarbij dus meerdere jaren in 1 plaatje verwerkt worden).

Naast plaatjes nog wat tekst mbt plant-data / adres-data / footer

Tevens het bedrijfslogo op de eerste pagina

Voor nu: jaren zijn kalenderjaren


Energieproduktie 2023 November (kWh)
kWh
2000
1750
1500
1250
1000
750
500
250
0
0
5
10
15
20
25
30
Dagen


Energieproduktie per jaar
126344.87
124141.92
2022
2023
250000
200000
150000
100000
50000
0
kWh
Jan
Feb
Maa
Apr
Mei
Jun
Jul
Aug
Sep
Okt
Nov
Dec
Maanden

Data in Mysql

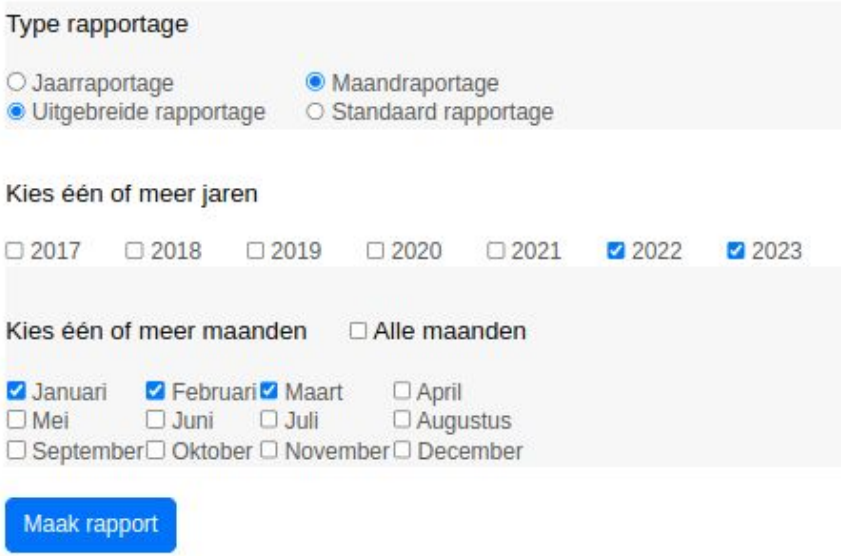
Inhoud table yields:

day	month	year	plantid	daily_yield	pr_rad_meter	pr_t_meter	poa_irradiance	avg_mtr_ambtemp
01	09	2022	906	2965.32	83.7	83.3	2308.2	16.98
02	09	2022	906	4169.44	83.4	84.1	3256.6	19.32
03	09	2022	906	6180.11	82.3	84.7	4892.6	19.88
...	...	...	...	...	...	...	...	...

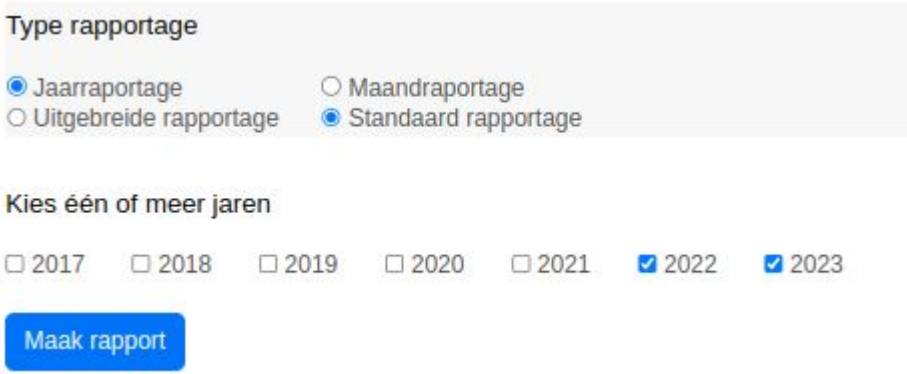
...

Selectie type rapport

maand of jaar rapportage, standaard of uitgebreid, bv.:



of



Selectie: bewaar de form-waarden in een dictionary

```
report_selection = {
   'plantids':                [41,906],
   'years':                   [2022,2023],
   'months':                  ['01','02','03'],
   'month_or_year':           ['month_report'],
   'standard_or_extended:'    [‘standard’]
}
```

Kun je gebruiken in een SQL-query, bv.:

```
f’SELECT plant_id,year,month,sum(daily_yield) FROM yields
WHERE plant_id in (41,906) and year in (2022,2023) GROUP BY
plant_id, year,month'
```

Let op: dit moet met een “prepared statement” (!)

Maar ook: selecteren van data in een DataFrame

- rijwaarden: loc, iloc
  - loc[1:2,'year','month','daily_yield':'avg_mtr_ambtemp']
- kolommen: df['day','daily_yield']
- df.query("year==2023 and plant_id=41")

Groeperen:

- df.groupby(['year','month'])['daily_yield'].sum()

“Dilemma”?

- SQL: veel mogelijkheden (select where in (), group by, sum, avg) :-)
- Pandas: veel mogelijkheden…(loc, query, group by, sum, mean) :-)

Uitgangspunt: ‘Selecteer uit SQL wat je nodig hebt en doe eventuele “nabewerking” met pandas DataFrames zodanig dat je zo makkelijk mogelijk kan plotten’

SQL queries

Maak onderscheid voor jaar/maand en standaard/uitgebreide rapportages, dan krijg je 4 queries:

1) Een standaard jaar-rapportage met de dagelijkse opbrengst **opgeteld per maand**:

```
f'SELECT plant_id,year,month, sum(daily_yield) from yields WHERE plant_id in
{plantids} and year in {years} group by plant_id, year,month'
```

2) Een standaard maand-rapportage met de dagelijkse opbrengst **per dag**:

```
f'SELECT plant_id,day,month,year,daily_yield from yields WHERE plant_id in
{plantids} and year in {years} and month in {months}'
```

Hier geen sum() nodig want de opbrengst is per dag en de weergave ook

3) Een uitgebreide jaar-rapportage met de dagelijkse opbrengst opgeteld per maand en de totale instraling, gemiddelde temperatuur en gemiddelde PR per maand

4) Een uitgebreide maand-rapportage met de dagelijkse opbrengst, de dagelijkse instraling, gemiddelde dag-temperatuur en de PR per dag

SQL output (in df)

	plant_id	year	month	sum(daily_yield)
0	41	2022	01	22539.870117
1	41	2022	02	54661.020203
2	41	2022	03	144284.659546
3	41	2022	04	165835.390503
4	41	2022	05	213346.811035
5	41	2022	06	223931.860352
6	41	2022	07	222464.569824
7	41	2022	08	199502.629761
8	41	2022	09	127646.380554
9	41	2022	10	84122.960022
10	41	2022	11	39756.389984
11	41	2022	12	18045.880066
12	41	2023	01	24549.230011
13	41	2023	02	53909.560120
14	41	2023	03	95242.019775
15	41	2023	04	158624.920044
16	41	2023	05	221072.576904
17	41	2023	06	238708.159180
			...	
			...	
31	**906**	**2022**	**09**	**40756.059753**
32	**906**	**2022**	**10**	**25987.910278**
33	**906**	**2022**	**11**	**14275.160027**
34	**906**	**2022**	**12**	**6664.079979**
35	**906**	**2023**	**01**	**9012.529907**
36	**906**	**2023**	**02**	**18226.799805**
37	**906**	**2023**	**03**	**29027.460114**
38	**906**	**2023**	**04**	**47806.519836**
39	**906**	**2023**	**05**	**65534.880005**
40	**906**	**2023**	**06**	**72134.989563**
41	**906**	**2023**	**07**	**54620.170166**
42	**906**	**2023**	**08**	**47503.580139**
43	**906**	**2023**	**09**	**43269.010284**
44	**906**	**2023**	**10**	**22348.260162**
45	**906**	**2023**	**11**	**7041.019989**

-> Dataframe df **per plant** (hier van plant met id 906)

	month	yield_2022	yield_2023
0	01	7812.910034	9012.529907
1	02	19087.930061	18226.799805
2	03	48663.310150	29027.460114
3	04	51366.839813	47806.519836
4	05	63244.329834	65534.880005
5	06	66310.399841	72134.989563
6	07	66547.340485	54620.170166
7	08	61662.210571	47503.580139
8	09	40756.059753	43269.010284
9	10	25987.910278	22348.260162
10	11	14275.160027	7041.019989
11	12	6664.079979	NaN

SQL -> Pandas DataFrame(s)

```
sql = f'SELECT plant_id,year,month,sum(daily_yield) from yields WHERE plant_id in
                {plantids} and year in {years} group by plant_id, year,month'
result = db.session.execute(sql)
data = mycursor.fetchall()
df = pd.DataFrame(data)
```

Gebruik df om dataframes **per jaar** (en per plant) te krijgen, die gaan we dan later samenvoegen (mergen) met de maand als gemeenschappelijke kolom.

```
df_years = [] # list van dataframes per jaar
for year in df['year'].drop_duplicates():
  df_years.append(df.query(f"year=={year} and plant_id=={plant_id}")
                  .rename(columns={'sum(daily_yield)': f'yield_{year}'})[['month',f'yield_{year}']])
```

[

	month	yield_2022
0	01	7812.910034
1	02	19087.930061
…		
10	11	14275.160027
11	12	6664.079979

,

	month	yield_2023
0	01	9012.529907
1	02	18226.799805
…		
10	11	7041.019989
11	12	NaN

]

Samenvoegen van de dataframes:

```
from functools import reduce
df_all = reduce(lambda df1, df2: df1.merge(df2, on='month', how="outer"), df_years)
```

Resultaat

	month	yield_2022	yield_2023
0	01	7812.910034	9012.529907
1	02	19087.930061	18226.799805
2	03	48663.310150	29027.460114
3	04	51366.839813	47806.519836
4	05	63244.329834	65534.880005
5	06	66310.399841	72134.989563
6	07	66547.340485	54620.170166
7	08	61662.210571	47503.580139
8	09	40756.059753	43269.010284
9	10	25987.910278	22348.260162
10	11	14275.160027	7041.019989
11	12	6664.079979	NaN

Maken van grafieken

matplotlib, figure, axes, axis, subplots, pyplot, plotly, plotly express, DataFrame.plot(), …

- matplotlib: since 2003
  - objecten als figure, axes, axis: de expliciete matplotlib “axes” API interface
  - submodule: pyplot (impliciete API voor matplotlib)
- DataFrame.plot(): 2010??
  - default backend: matplotlib
- plotly: since 2014 interactie (denk aan Dash)
  - plotly express: 2019 (API voor plotly)

Statische plaatjes in een rapport: ik gebruik matplotlib met de core objecten:

- import matplotlib.pyplot as plt
- fig, ax = plt.subplots()

Gebruik van matplotlibs objecten

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

# linegraph 1

ax.plot(df['month'],df['yield_2022'],color='b',label='2022')

# bargraph 2

ax.bar(df['month'],df['yield_2023'],color='g',label='2023')

fig.legend()

fig.savefig(f'{reporting_dir}/{plant_name}/{image_filename}')
```

matplotlib of Dataframe.plot()?

“Subtiele” verschillen:

```
df['Data_col'].plot(kind='bar', ax=ax, color='b', width=1, label='Data1')
```

of

```
ax.bar(df['x-values'],df['Data_col'], color='b', width=1, label='Data1')
```

Beter gewoon de core-components (fig en ax) te gebruiken.

Nu nog een pdf

We hebben de plaatjes. Nu nog wat tekst, header, footer, pagina-nummers…

Je kunt matplotlib (en dus ook DataFrame.plot()) gebruiken om je data als pdf te bewaren maar matplotlib is niet goed met tekst, headers, footers etc…

Ik gebruik **fpdf2** (opvolger van pyfpdf (python port van de FPDF PHP library).

In file `write_pdf.py`:

```
from fpdf import FPDF
class WritePDF(FPDF):
  def header(self):
    ..
  def footer
```

fpdf2 1/3

```
def header(self):
   if self.page_no() == 1:
      self.image(f'{plotdir}/solarcontrol_logo.png', x=120, y=0, w=80)
  else:
     self.cell(0, 10, f'{self.page_no()}/{{nb}}', align="C")
def footer(self):
  # Positioneer cursor 18mm vanaf de onderkant:
  self.set_y(-18)
  if self.page_no() != 1:
    self.set_font("helvetica",size=8)
    self.set_text_color(50, 60, 168)
    self.cell(0, 10, f'Keizer Karelplein 32Q 6511 NH Nijmegen - info@arienssolar.nl', align="C")
```

fpdf2 2/3

```
def overview(self,plantdata):
    self.set_font('segoeui','',16);
    self.cell(0, 10, f"Systeemgegevens")
    self.set_font('segoeui','',12);
    self.ln()
    self.cell(54.5, 8, 'Locatie', border="TLRB", align="L", fill=False)
    self.cell(109, 8, f'{plantdata["street"]} {plantdata["zipcode"]} {plantdata["city"]}', border="TLRB",...)
    self.ln()
    self.cell(54.5, 8, 'Totaal Vermogen', border="LRB", align="L", fill=False)
    self.cell(109, 8, f'{plantdata["capacity"]}', border="LRB", align="L", fill=False)
```

fpdf2 3/3

```
# Maak pdf object

from .write_pdf import WritePDF

pdf = WritePDF(orientation="P", unit="mm", format="A4")


# Create PDF file
def create_pdf_file(self, pdf,plant_name,plant_data):
  pdf.add_page()
  pdf.overview(plant_data)
  pdf.add_page()
  pdf.image(f'{self._reporting_dir}/{plant_name}/yield.png', x=10, w=170)
  pdf.output(f'{self._reporting_dir}/{plant_ref}/Year-Energieproduktie_{plant_ref}.pdf')
```

Referenties

- Pandas DataFrames: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
- Matplotlib: https://matplotlib.org/stable/users/getting_started/
- DeepSeek: https://chat.deepseek.com/coder
- fpdf2: https://py-pdf.github.io/fpdf2/index.html

Bier?