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

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

In [2]:
```
df = pd.read_csv("/kaggle/input/western-europe-power-consumption/fr.csv")
df.head()
```

Out[2]:

	start	end	load
0	2015-01-01 00:00:00+00:00	2015-01-01 01:00:00+00:00	70929.0
1	2015-01-01 01:00:00+00:00	2015-01-01 02:00:00+00:00	69773.0
2	2015-01-01 02:00:00+00:00	2015-01-01 03:00:00+00:00	66417.0
3	2015-01-01 03:00:00+00:00	2015-01-01 04:00:00+00:00	64182.0
4	2015-01-01 04:00:00+00:00	2015-01-01 05:00:00+00:00	63859.0

In [3]:
```
df['start'] = pd.to_datetime(df['start'])
df['start'] = df['start'].dt.tz_localize(None)
df = df.rename(columns={'start' : 'DateTime'})
df = df.drop('end',axis=1)
df = df[(df["DateTime"] >= '2015-01-01') & (df['DateTime'] < '2020-05-31')]
df = df.sort_values(ascending=True,by='DateTime')
df.head()
```

Out[3]:

	DateTime	load
0	2015-01-01 00:00:00	70929.0
1	2015-01-01 01:00:00	69773.0
2	2015-01-01 02:00:00	66417.0
3	2015-01-01 03:00:00	64182.0
4	2015-01-01 04:00:00	63859.0

In [4]:
```
df.tail()
```

Out[4]:

	DateTime	load
47378	2020-05-30 19:00:00	37730.0
47379	2020-05-30 20:00:00	40453.0
47380	2020-05-30 21:00:00	41831.0
47381	2020-05-30 22:00:00	38758.0
47382	2020-05-30 23:00:00	36048.0

In [5]:
```
df1 = df.set_index('DateTime')
df1.head()
```

Out[5]:

	load
DateTime	
2015-01-01 00:00:00	70929.0
2015-01-01 01:00:00	69773.0
2015-01-01 02:00:00	66417.0
2015-01-01 03:00:00	64182.0
2015-01-01 04:00:00	63859.0

In [6]:
```
df_day = df1.resample('D').mean()
df_day.head()
```

Out[6]:

	load
DateTime	
2015-01-01	66370.208333
2015-01-02	68225.916667
2015-01-03	62944.166667
2015-01-04	58506.875000
2015-01-05	71173.000000

In [7]:
```
df2 = df[['DateTime', 'load']].rename(columns={"DateTime": "ds", "load": "y"})
df2.head()
```

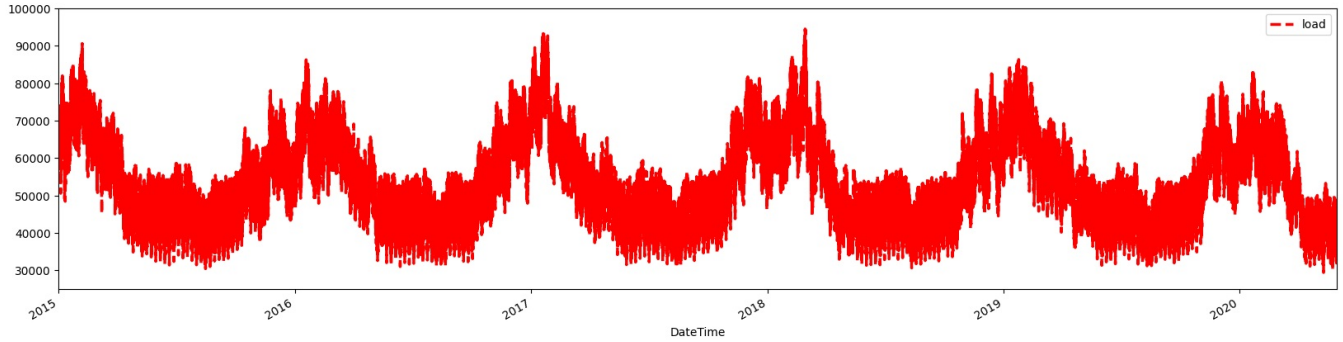
Out[7]:

	ds	y
0	2015-01-01 00:00:00	70929.0
1	2015-01-01 01:00:00	69773.0
2	2015-01-01 02:00:00	66417.0
3	2015-01-01 03:00:00	64182.0
4	2015-01-01 04:00:00	63859.0

In [8]:
```
print("Starting Date  : ", df['DateTime'].min())
print("End Date       : ", df['DateTime'].max())
```

```
Starting Date  :  2015-01-01 00:00:00
End Date       :  2020-05-30 23:00:00
```

In [9]:
```
dfx= df1.copy('Deep')
dfx.plot(figsize=(20,5),color='red',lw=2.5,style="--",xlim=['2015-01-01', '2020-05-31'],ylim=[25000,100000]);
```

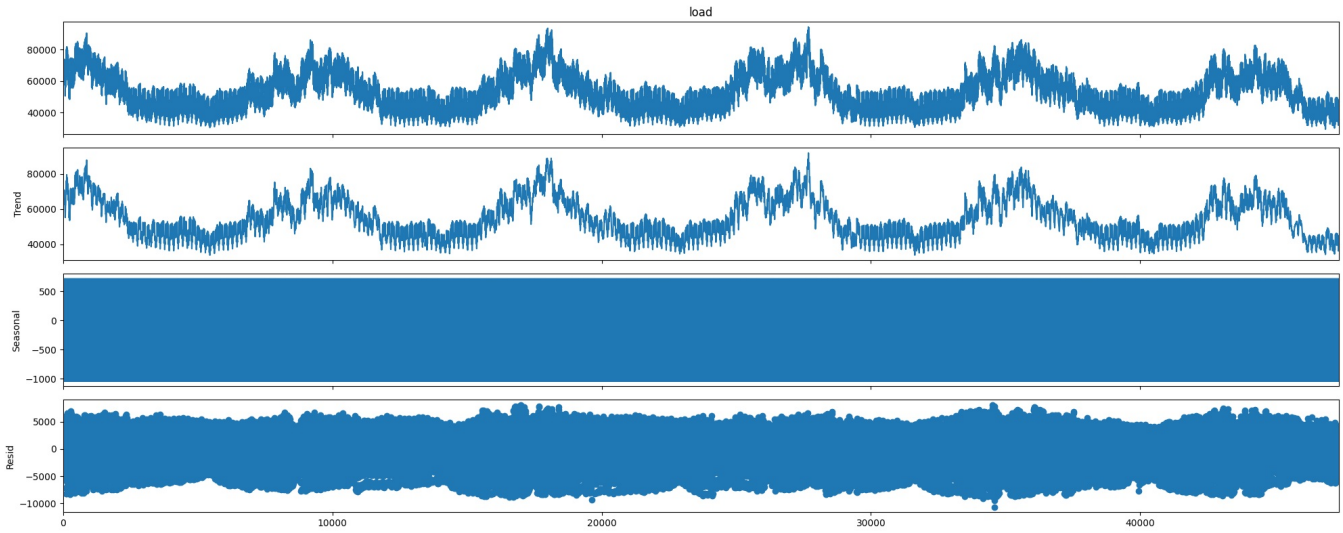


In [10]:
```
df_month = df1.resample('M').mean()
df_month = df_month.reset_index('DateTime')
df_month.head()
```

Out[10]:

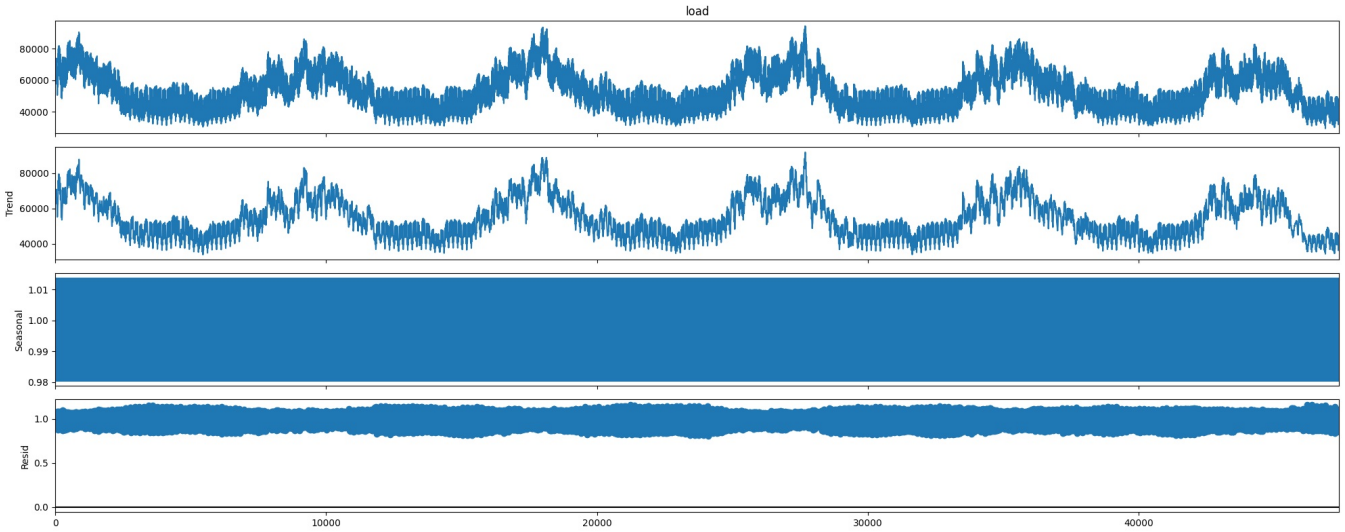
	DateTime	load
0	2015-01-31	69523.116935
1	2015-02-28	71374.474702
2	2015-03-31	60624.438172
3	2015-04-30	50608.841667
4	2015-05-31	45209.040377

In [11]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

from pylab import rcParams
rcParams['figure.figsize'] = 20, 8

decomposition = seasonal_decompose(df["load"],period =12, model = 'additive')
figure = decomposition.plot()
```



In [12]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

from pylab import rcParams
rcParams['figure.figsize'] = 20, 8
```

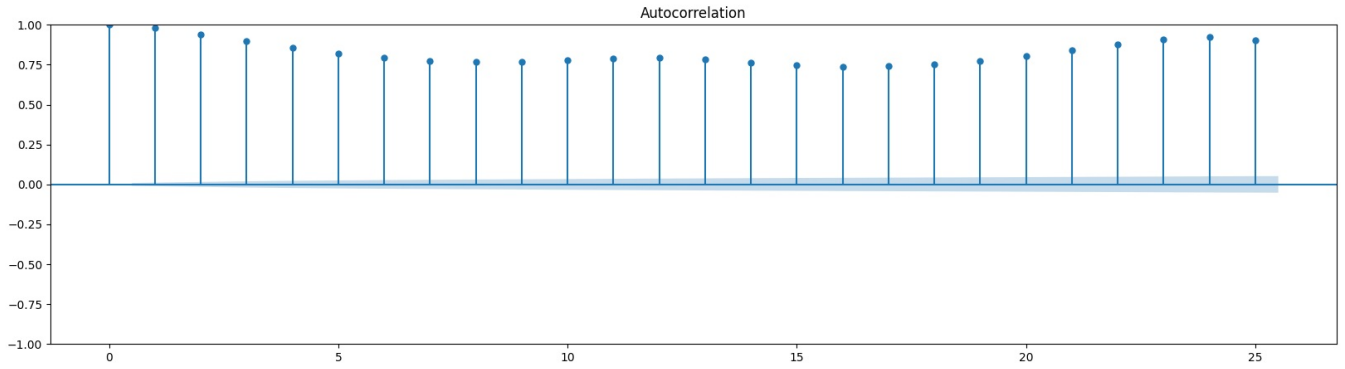
```
decomposition = seasonal_decompose(df["load"],period =12, model = 'multiplicative')
figure = decomposition.plot()
```



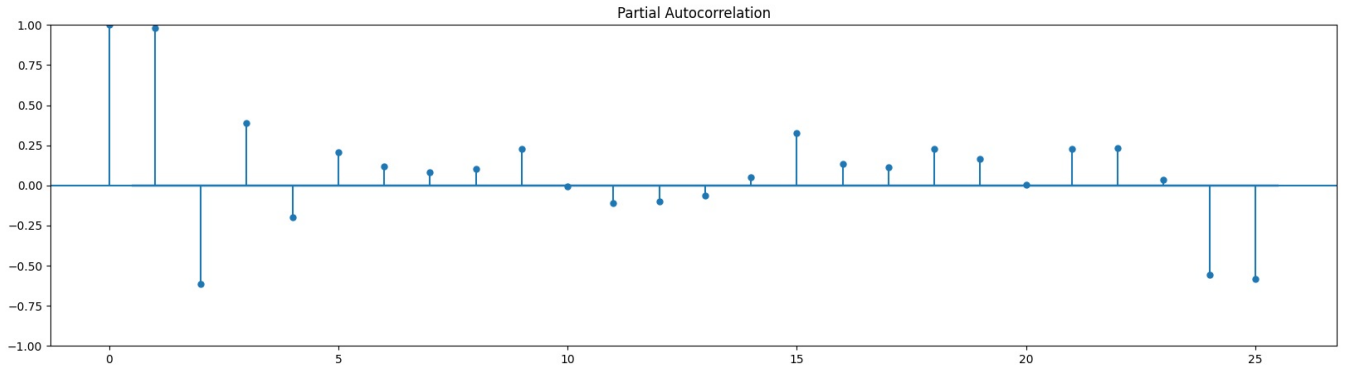
In [13]:
```
from statsmodels.tsa.stattools import adfuller
result = adfuller(df1['load'])
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF Statistic: -7.822472
p-value: 0.000000
Critical Values:
        1%: -3.430
        5%: -2.862
        10%: -2.567
```

In [14]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from pylab import rcParams
rcParams['figure.figsize'] = 20, 5

plot_acf(df1['load'], lags = 25); print()
```



In [15]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from pylab import rcParams
rcParams['figure.figsize'] = 20, 5

plot_pacf(df1['load'], lags = 25); print()
```



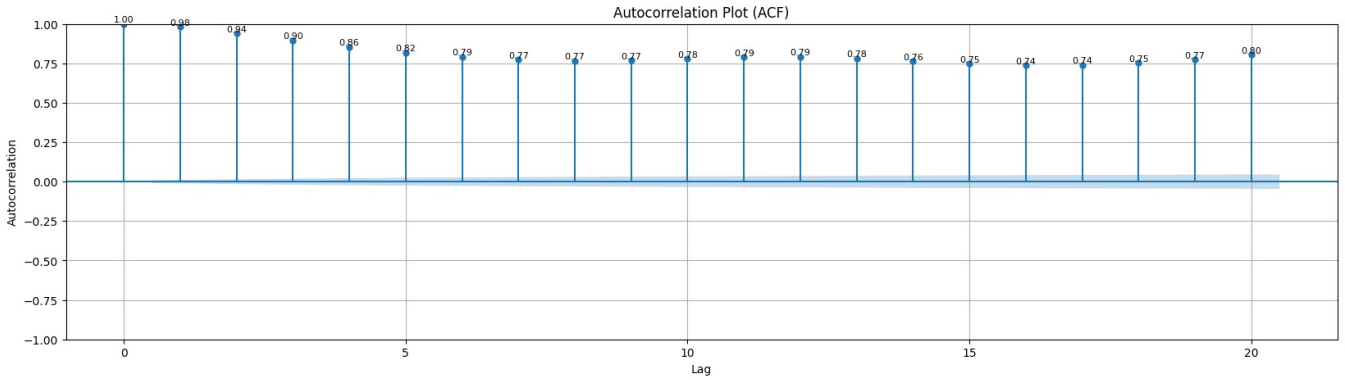
In [16]:
```
df.head()
```

Out[16]:

	DateTime	load
0	2015-01-01 00:00:00	70929.0
1	2015-01-01 01:00:00	69773.0
2	2015-01-01 02:00:00	66417.0
3	2015-01-01 03:00:00	64182.0
4	2015-01-01 04:00:00	63859.0

ARIMA

In [17]:
```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import acf

# Plot ACF
fig, ax = plt.subplots(figsize=(20, 5))
plot_acf(df1['load'], lags=20, alpha=0.05, ax=ax)
ax.set_xlabel('Lag')
ax.set_ylabel('Autocorrelation')
ax.set_title('Autocorrelation Plot (ACF)')
ax.grid(True)

# Add autocorrelation values on each bar
acf_values = acf(df1['load'], nlags=20, fft=False)
for lag, acf_value in enumerate(acf_values):
    ax.text(lag, acf_value, f'{acf_value:.2f}', ha='center', va='bottom', fontsize=8, color='black')

plt.show()
```

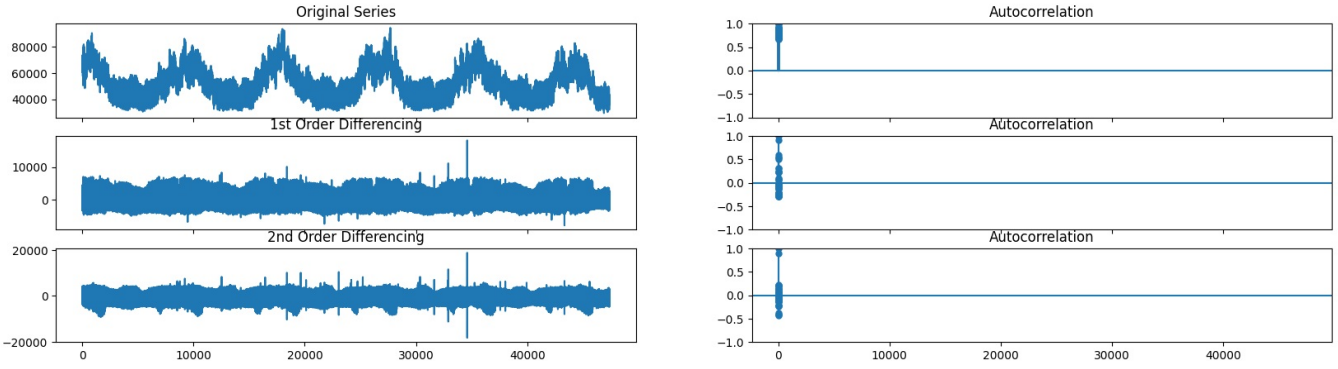


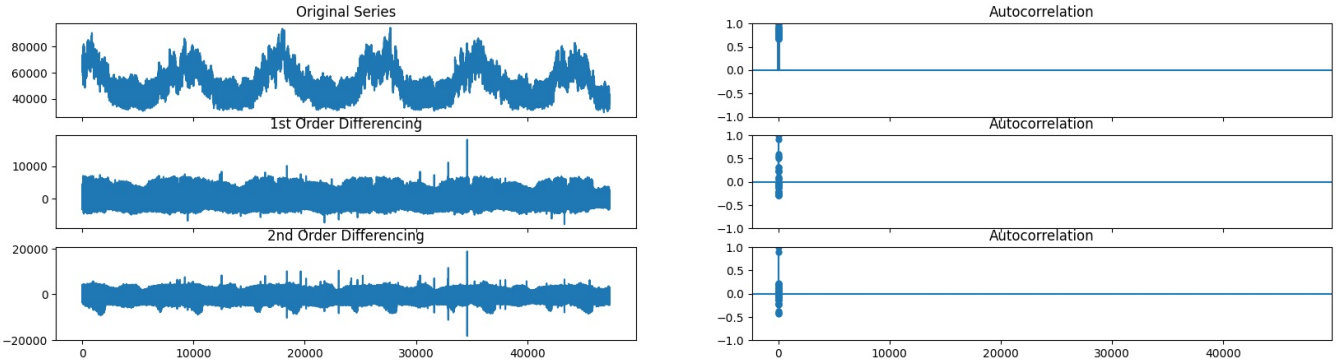
In [18]:
```
from statsmodels.tsa.stattools import acf
x_acf = pd.DataFrame(acf(df1['load']))
print(x_acf)
```

	0
0	1.000000
1	0.981533
2	0.940955
3	0.896942
4	0.855664
5	0.819189
6	0.791462
7	0.774461
8	0.766856
9	0.768995
10	0.779449
11	0.790146
12	0.792038
13	0.782197
14	0.764046
15	0.746835
16	0.738583
17	0.740588
18	0.752262
19	0.774415
20	0.804670
21	0.839128
22	0.876053
23	0.910044
24	0.923774
25	0.904081
26	0.864368
27	0.822067
28	0.782659
29	0.747921
30	0.721548
31	0.705429
32	0.698377
33	0.700829
34	0.711287
35	0.721782
36	0.723574
37	0.713920
38	0.696025
39	0.678814
40	0.670142
41	0.671361
42	0.681848
43	0.702249
44	0.730238
45	0.762121
46	0.796367

In [19]:
```
# auttocorrelation check difference values
from statsmodels.graphics.tsaplots import plot_acf
fig, ax = plt.subplots(3,2, sharex=True)
ax[0,0].plot(df['load'])
ax[0,0].set_title('Original Series')
plot_acf(df['load'], ax=ax[0,1])

# 1st Differencing
ax[1,0].plot(df['load'].diff()); ax[1,0].set_title('1st Order Differencing')
plot_acf(df['load'].diff().dropna(), ax=ax[1,1])

# 2nd Differencing
ax[2,0].plot(df['load'].diff().diff()); ax[2,0].set_title('2nd Order Differencing')
plot_acf(df['load'].diff().diff().dropna(), ax=ax[2,1])
```

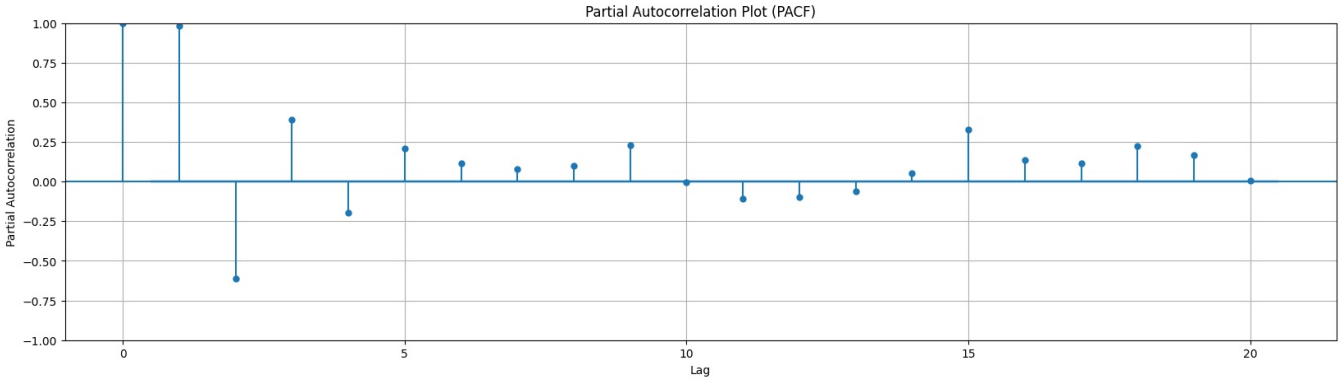
Out[19]:





In [20]:
```
from statsmodels.graphics.tsaplots import plot_pacf

# Plot PACF
plt.figure(figsize=(12, 6))
plot_pacf(df['load'], lags=20)
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.title('Partial Autocorrelation Plot (PACF)')
plt.grid(True)
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



The basic idea is that if we are dealing with a series that can be represented as a stationary process with polynomial trend, we can model it jointly (instead of manually differencing d times and then flipping back).

Sideline point worth mentioning is that a non-seasonal ARIMA encapsulates other models as special cases (to quote a line from several textbooks I used in university: verifying those assertions is left as an exercise to the reader ;-)

- ARIMA(0,0,0) is white noise
- ARIMA(0,1,0) is random walk
- ARIMA(p,0,0) is AR(p)
- ARIMA(0,0,q) is MA(q)
- ARIMA(0, 1, 1) is equivalent to single (basic) exponential smoothing model
- ARIMA(0, 2, 2) is equivalent to Holt's linear method with additive errors, i.e. double exponential smoothing

SARIMA

The "I" in ARIMA corresponds to the "integrated" component, which is a formal way of saying we are incorporating the trend into our setup starting with a stationary ARMA(p,q) process. Following a similar logic, we can incorporate the seasonal component and allow it to follow the same type of dynamic: a seasonal ARIMA (SARIMA) model can be denoted as ARIMA(p,d,q)(P,D,Q)m, where:

- m refers to the number of periods in each season
- (lowercase) p,d,q refer to the definition of the ARIMA part
- (uppercase) P,D,Q refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

The last part is not entirely obvious at an intuitive model (at least it wasn't to me when I learned it). A good intuition to think about is that we want to identify components of the seasonal part the way we would for the series itself:

- we isolate the seasonal component, e.g. through seasonal decomposition

- to establish the value of D we test the seasonal component - is it stationary? is the differenced version stationary?
- the seasonal autoregressive value P can be established by inspecting the PACF of the seasonal component
- the seasonal moving average value Q can be established by inspect the ACF of the seasonal component

In [21]:
```
#import statsmodels.api as sm

#p, d, q = 1, 1, 3
#model = sm.tsa.statespace.SARIMAX(df['load'], order=(p, d, q), seasonal_order=(p, d, q, 12))
#model = model.fit()
#model.summary()
```

In [22]:
```
#start_date = df.index[-1]
#end_date = start_date + pd.DateOffset(days=365)

#predictions = model.predict(start=start_date, end=end_date)
```

In [23]:
```
#import plotly.graph_objs as go

#trace_observed = go.Scatter(x=df.index, y=df['load'], mode='lines', name='Observed')
#trace_predicted = go.Scatter(x=predictions.index, y=predictions, mode='lines', name='Predicted')

#layout = go.Layout(title='SARIMA Model',
#                   xaxis=dict(title='Date'),
#                   yaxis=dict(title='load'))

#fig = go.Figure(data=[trace_observed, trace_predicted], layout=layout)
#fig.show()
```

Fb- Prophet

In [24]:
```
df1.head()
```

Out[24]:

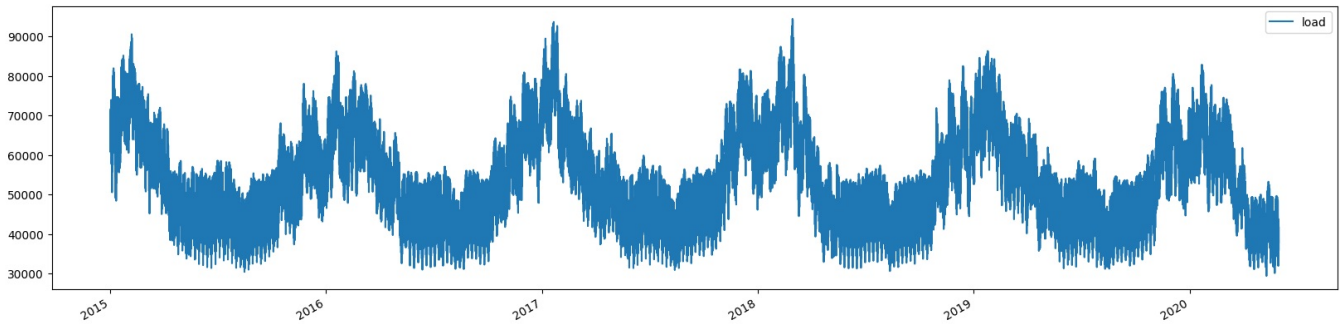
DateTime	load
2015-01-01 00:00:00	70929.0
2015-01-01 01:00:00	69773.0
2015-01-01 02:00:00	66417.0
2015-01-01 03:00:00	64182.0
2015-01-01 04:00:00	63859.0

In [25]:
```
dfx = df.copy('Deep')
dfx = dfx.rename(columns={'DateTime' : 'ds', 'load' : 'y'})
dfx.head()
```

Out[25]:

	ds	y
0	2015-01-01 00:00:00	70929.0
1	2015-01-01 01:00:00	69773.0
2	2015-01-01 02:00:00	66417.0
3	2015-01-01 03:00:00	64182.0
4	2015-01-01 04:00:00	63859.0

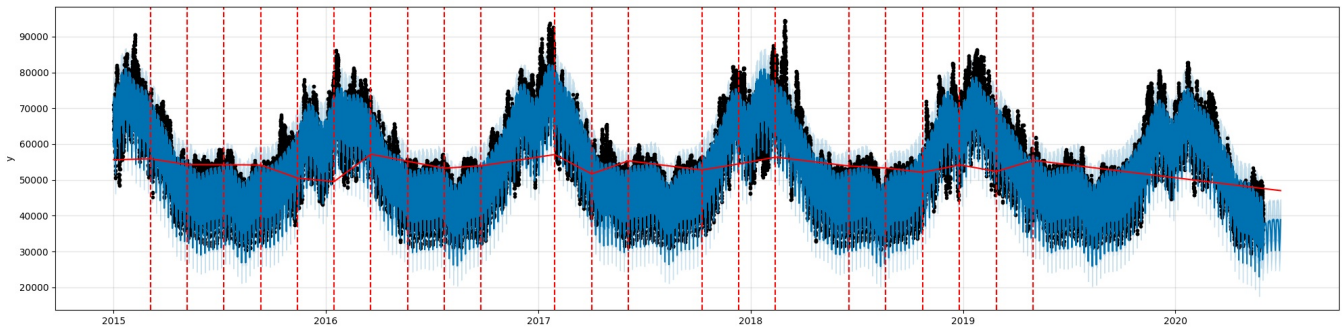
In [26]:
```
df1.plot(figsize=(20,5), xlabel = '');
```



In [27]:
```
from prophet import Prophet
from prophet.diagnostics import cross_validation, performance_metrics
from prophet.plot import plot_yearly, add_changepoints_to_plot

# automatic detection of changepoints
m = Prophet()
m.fit(dfx)
```

```
future = m.make_future_dataframe(periods= 30)
forecast = m.predict(future)
fig = m.plot(forecast, figsize=(20, 5), xlabel = '')
a = add_changepoints_to_plot(fig.gca(), m, forecast)
```

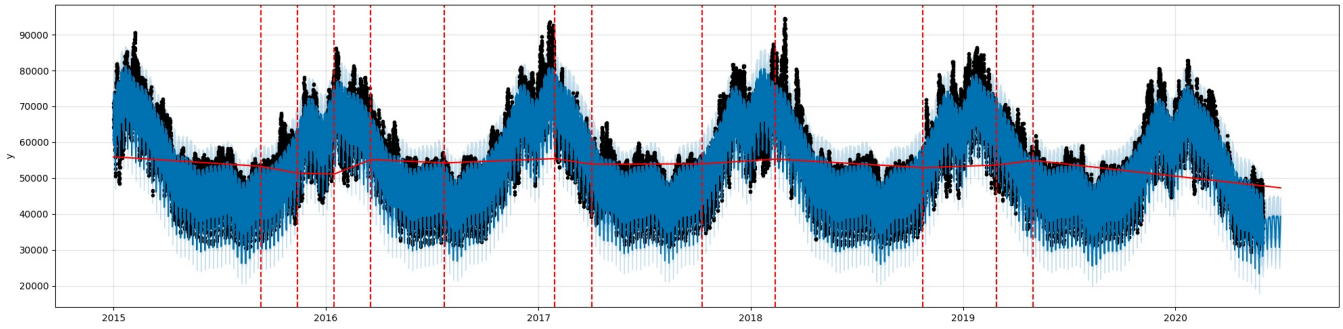
```
11:22:27 - cmdstanpy - INFO - Chain [1] start processing
11:23:12 - cmdstanpy - INFO - Chain [1] done processing
```



In [28]:

```
m = Prophet(changepoint_prior_scale = 0.01)
m.fit(dfx)
future = m.make_future_dataframe(periods= 30)
forecast = m.predict(future)
fig = m.plot(forecast, figsize=(20,5), xlabel = '')
a = add_changepoints_to_plot(fig.gca(), m, forecast)
```

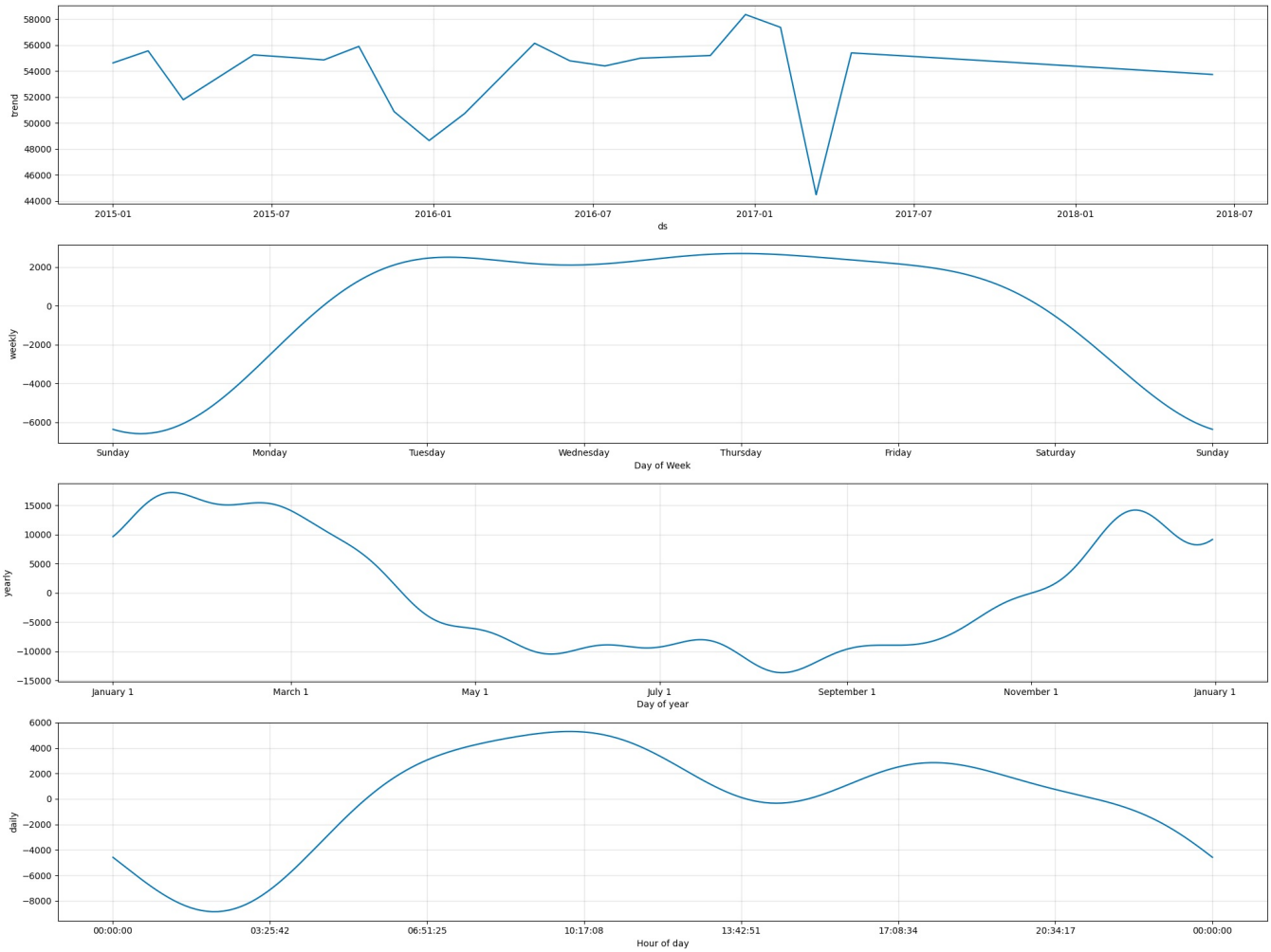
```
11:23:24 - cmdstanpy - INFO - Chain [1] start processing
11:24:08 - cmdstanpy - INFO - Chain [1] done processing
```



In [29]:

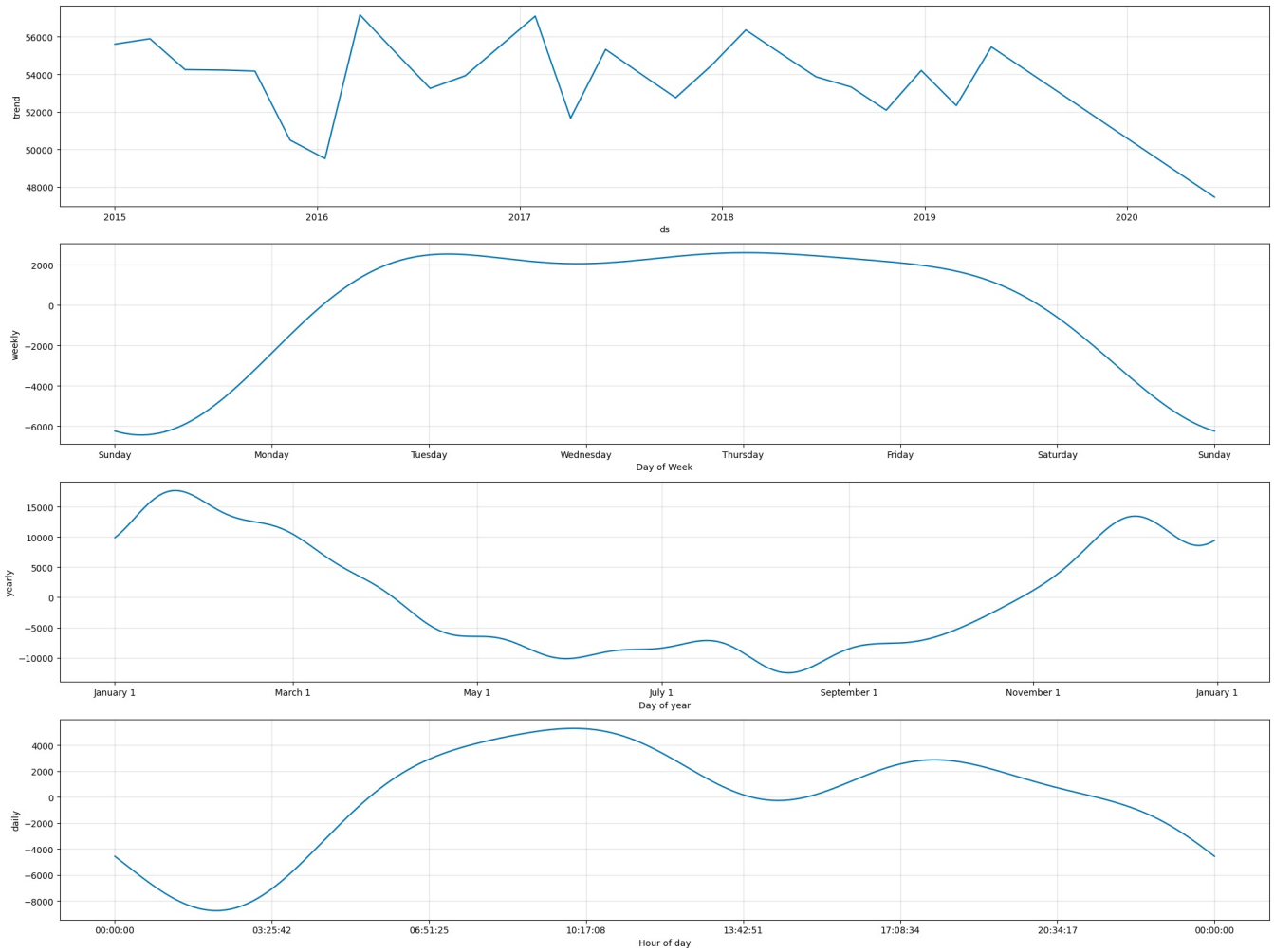
```
# we reduce the dataset size for speed - the only requirement while doing is to keep at least two complete cycl
# for each seasonality we intend to fit
m = Prophet().fit(dfx.iloc[:30000])
future = m.make_future_dataframe(periods = 24, freq = 'H')
forecast = m.predict(future)
m.plot_components(forecast, figsize=(20,15))
print()
```

```
11:24:19 - cmdstanpy - INFO - Chain [1] start processing
11:24:47 - cmdstanpy - INFO - Chain [1] done processing
```



In [30]:
```
# we proceed in a similar manner as before
m = Prophet().fit(dfx)
future = m.make_future_dataframe(periods = 7, freq = 'D')
forecast = m.predict(future)
m.plot_components(forecast, figsize=(20,15))
print()
```

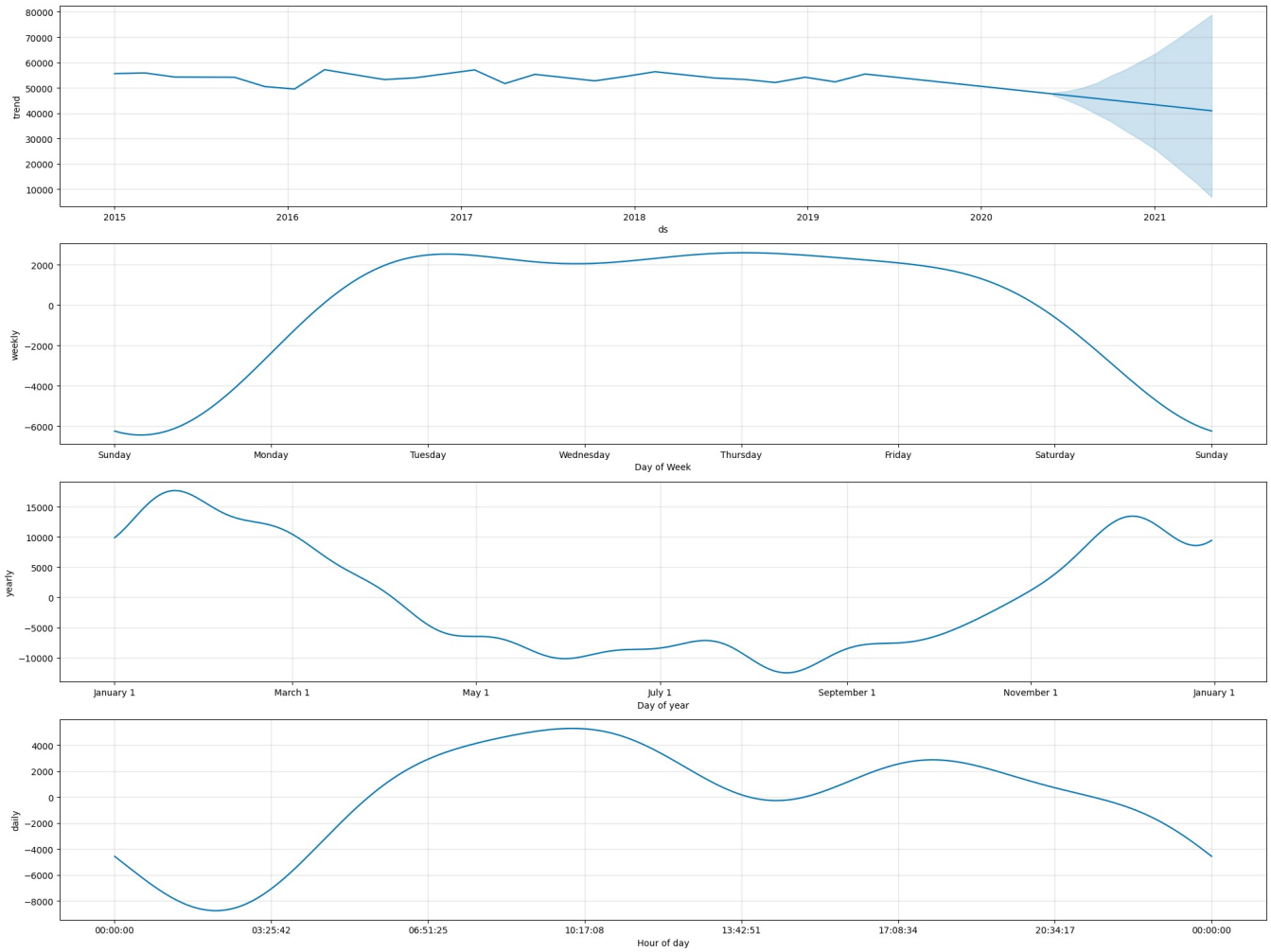
```
11:24:56 - cmdstanpy - INFO - Chain [1] start processing
11:25:42 - cmdstanpy - INFO - Chain [1] done processing
```



In [31]:

```
m = Prophet().fit(dfx)
future = m.make_future_dataframe(periods = 12, freq = 'M')
forecast = m.predict(future)
m.plot_components(forecast, figsize=(20,15))
print()
```

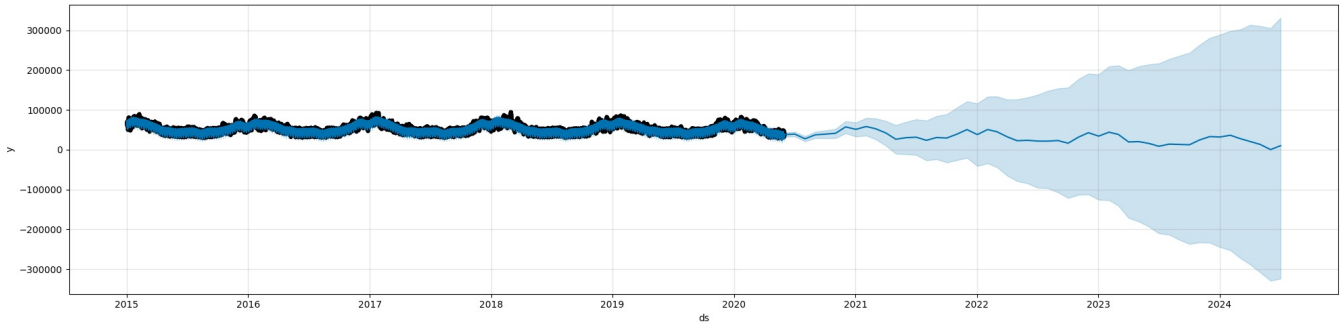
```
11:25:54 - cmdstanpy - INFO - Chain [1] start processing
11:26:40 - cmdstanpy - INFO - Chain [1] done processing
```



In [32]:

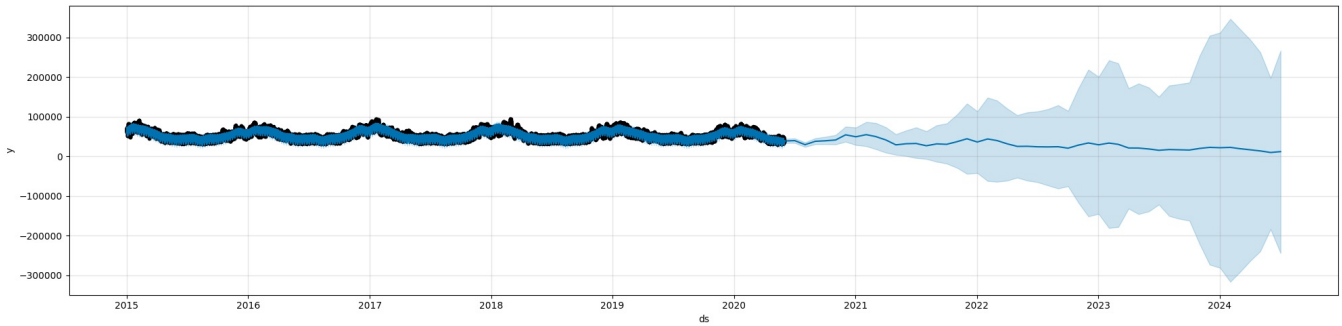
```
m = Prophet(seasonality_mode ='additive')
m.fit(dfx)
future = m.make_future_dataframe(50, freq='MS')
forecast = m.predict(future)
fig = m.plot(forecast, figsize=(20,5))
plt.show()
```

```
11:26:52 - cmdstanpy - INFO - Chain [1] start processing
11:27:37 - cmdstanpy - INFO - Chain [1] done processing
```



In [33]:
```
m = Prophet(seasonality_mode ='multiplicative')
m.fit(dfx)
future = m.make_future_dataframe(50, freq='MS')
forecast = m.predict(future)
fig = m.plot(forecast, figsize=(20,5))
plt.show()
```

```
11:27:48 - cmdstanpy - INFO - Chain [1] start processing
11:28:36 - cmdstanpy - INFO - Chain [1] done processing
```



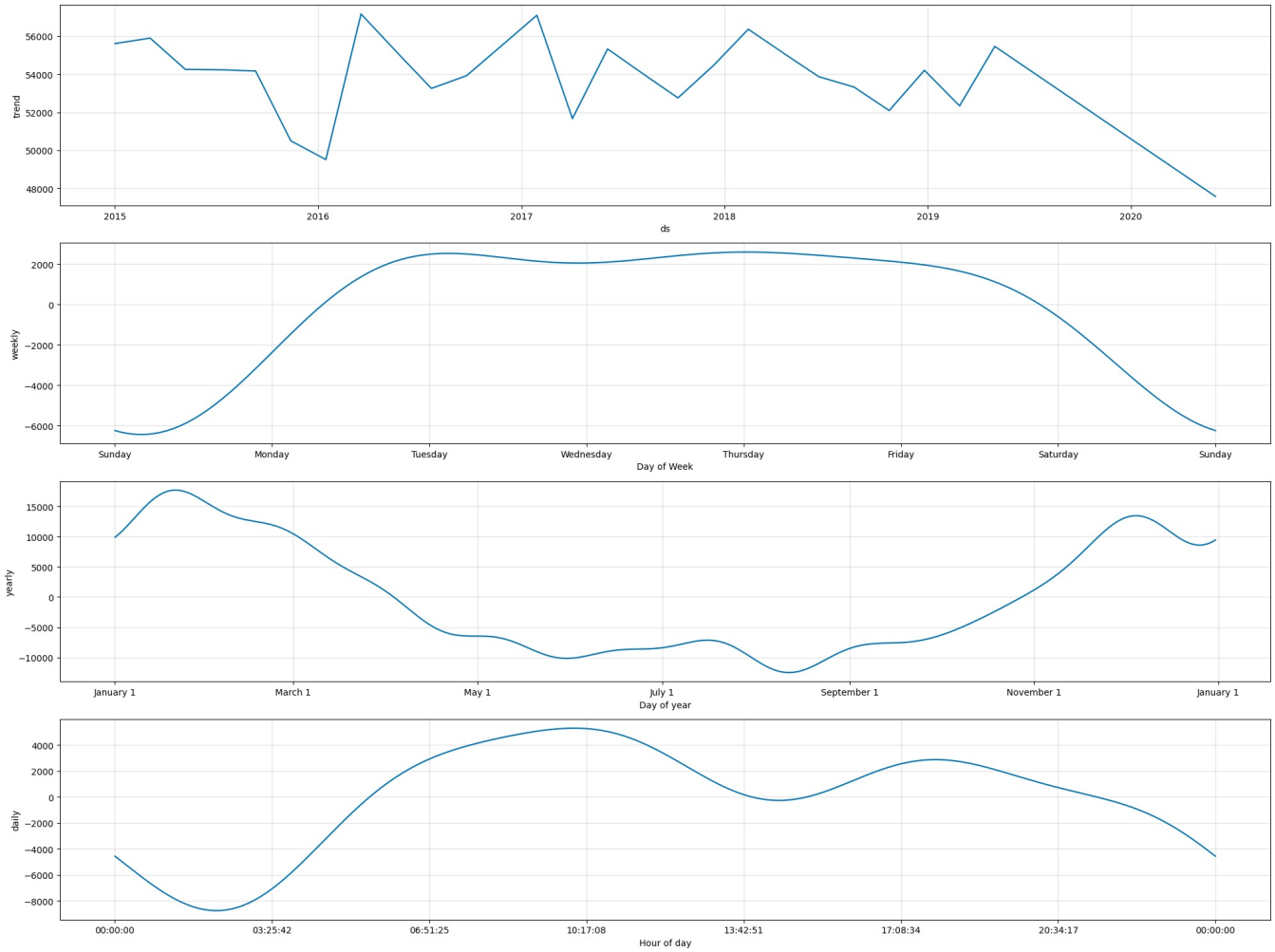
In [34]:
```
dfx.head()
```

Out[34]:

	ds	y
0	2015-01-01 00:00:00	70929.0
1	2015-01-01 01:00:00	69773.0
2	2015-01-01 02:00:00	66417.0
3	2015-01-01 03:00:00	64182.0
4	2015-01-01 04:00:00	63859.0

In [35]:
```
first_valid = np.where(~np.isnan(dfx['y']))[0][0]

m = Prophet().fit(dfx)
future = m.make_future_dataframe(periods = 24, freq = 'H')
forecast = m.predict(future)
m.plot_components(forecast, figsize=(20,15))
print()
```

```
11:28:47 - cmdstanpy - INFO - Chain [1] start processing
11:29:32 - cmdstanpy - INFO - Chain [1] done processing
```

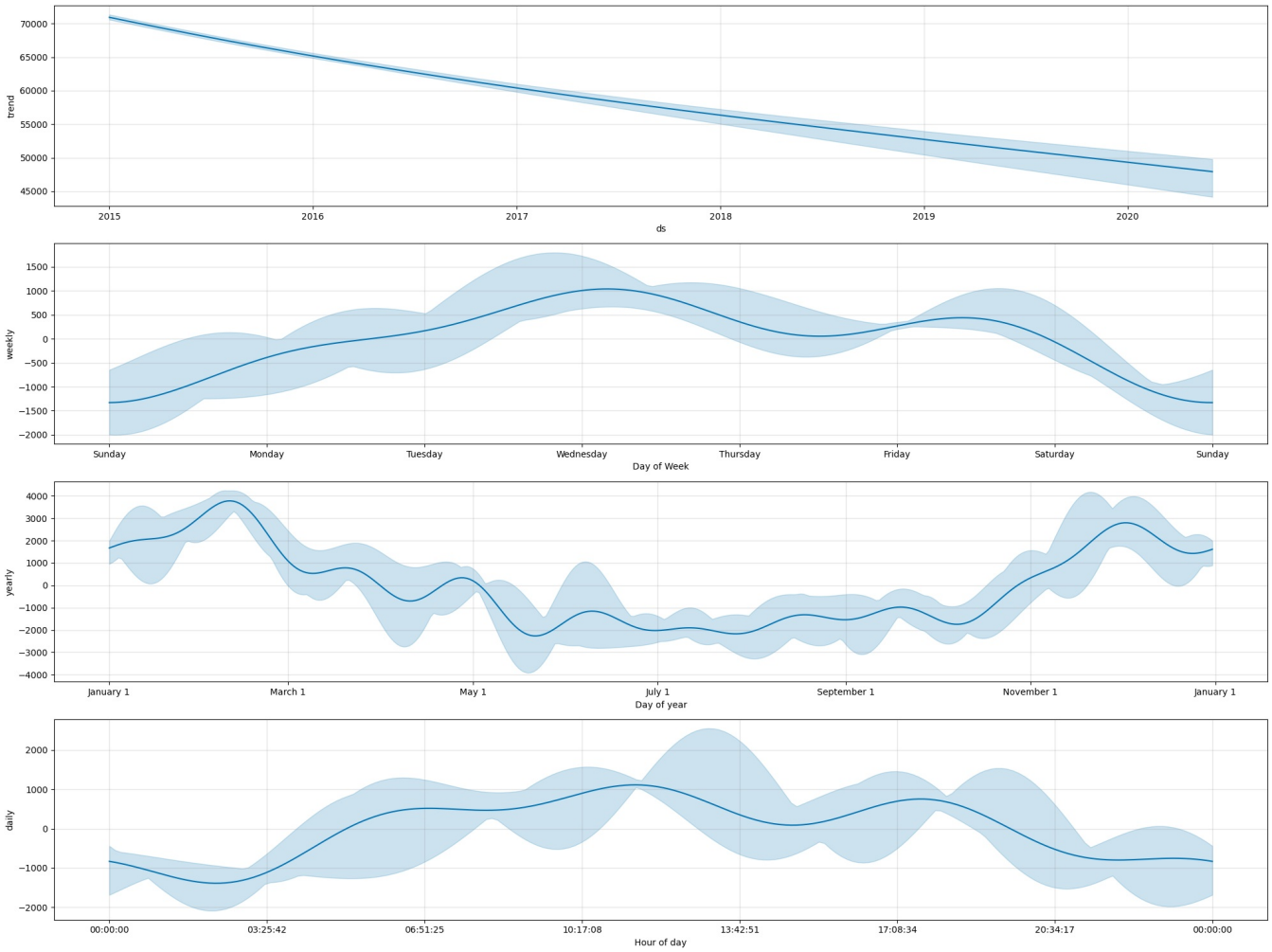


In [36]:
```
m = Prophet(mcmc_samples = 10).fit(dfx)
future = m.make_future_dataframe(periods = 24, freq = 'H')
forecast = m.predict(future)
m.plot_components(forecast, figsize=(20,15))
print()
```

```
11:29:42 - cmdstanpy - INFO - CmdStan installation /opt/conda/lib/python3.10/site-packages/prophet/stan_model/c
mdstan-2.26.1 missing makefile, cannot get version.
11:29:42 - cmdstanpy - INFO - Cannot determine whether version is before 2.28.
11:29:44 - cmdstanpy - INFO - CmdStan start processing
```

```
11:29:47 - cmdstanpy - INFO - CmdStan done processing.
11:29:47 - cmdstanpy - WARNING - Non-fatal error during sampling:
Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/stan/prophe
t.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Matrix of independent variables is inf, but must be finite! (in '/projec
t/python/stan/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/stan/prophe
t.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Matrix of independent variables is inf, but must be finite! (in '/projec
t/python/stan/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/stan/prophe
t.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Matrix of independent variables is inf, but must be finite! (in '/projec
t/python/stan/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/stan/prophe
t.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Matrix of independent variables is inf, but must be finite! (in '/projec
t/python/stan/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
        Exception: normal_id_glm_lpdf: Scale vector is 0, but must be positive finite! (in '/project/python/sta
n/prophet.stan', line 137, column 2 to line 142, column 4)
Consider re-running with show_console=True if the above output is unclear!
```

```
11:29:48 - cmdstanpy - WARNING - Some chains may have failed to converge.
        Chain 1 had 5 divergent transitions (100.0%)
        Chain 2 had 5 divergent transitions (100.0%)
        Chain 3 had 5 divergent transitions (100.0%)
        Chain 4 had 5 divergent transitions (100.0%)
        Use the "diagnose()" method on the CmdStanMCMC object to see further information.
```



Holidays

In [37]:
```
m = Prophet()
m.add_country_holidays(country_name='FR')
m.fit(dfx)
# List the holiday names
m.train_holiday_names
```

```
11:30:02 - cmdstanpy - INFO - Chain [1] start processing
11:30:47 - cmdstanpy - INFO - Chain [1] done processing
```

Out[37]:
```
0            Jour de l'an
1         Fête du Travail
2     Fête de la Victoire
3          Fête nationale
4               Armistice
5         Lundi de Pâques
6      Lundi de Pentecôte
7               Ascension
8             Assomption
9               Toussaint
10                  Noël
dtype: object
```

In [38]:
```
future
```

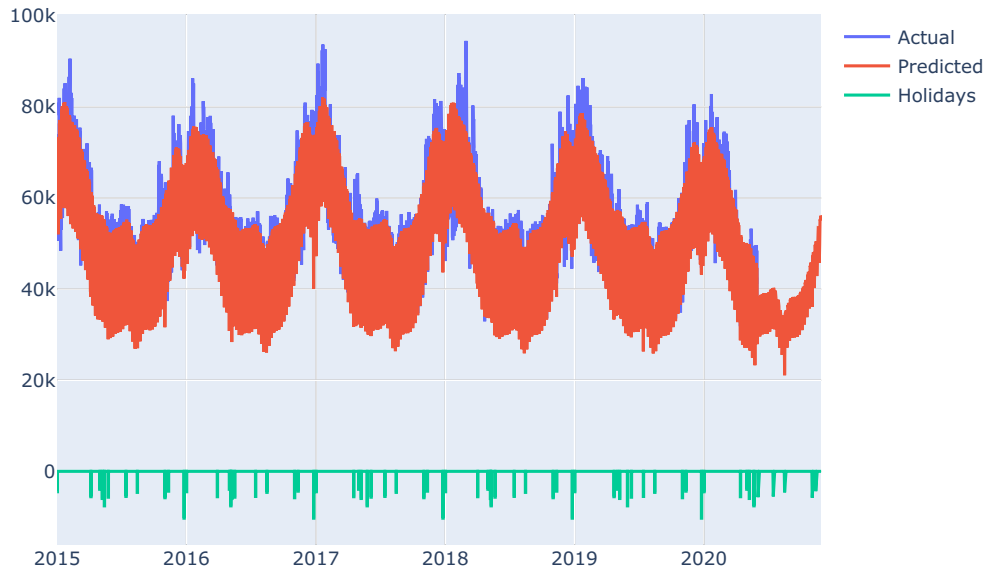
Out[38]:

	ds
0	2015-01-01 00:00:00
1	2015-01-01 01:00:00
2	2015-01-01 02:00:00
3	2015-01-01 03:00:00
4	2015-01-01 04:00:00
...	...
47402	2020-05-31 19:00:00
47403	2020-05-31 20:00:00
47404	2020-05-31 21:00:00
47405	2020-05-31 22:00:00
47406	2020-05-31 23:00:00

47407 rows × 1 columns

In [39]:

```
import plotly.graph_objs as go
# as before, we create a dataframe holding the dates for the entire forecast horizon - 180 days ahead
future = m.make_future_dataframe(periods=180, freq='D')
# generate the actual forecast
forecast = m.predict(future)

fig = go.Figure()
# full disclosure: I am changing the plotting style for this one, because I don't know how to overlay the two g

fig.add_trace(go.Scatter(x=dfx['ds'], y=dfx['y'], name='Actual',))
fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['yhat'], name='Predicted',))
fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['holidays'], name='Holidays',))
fig.show()
```

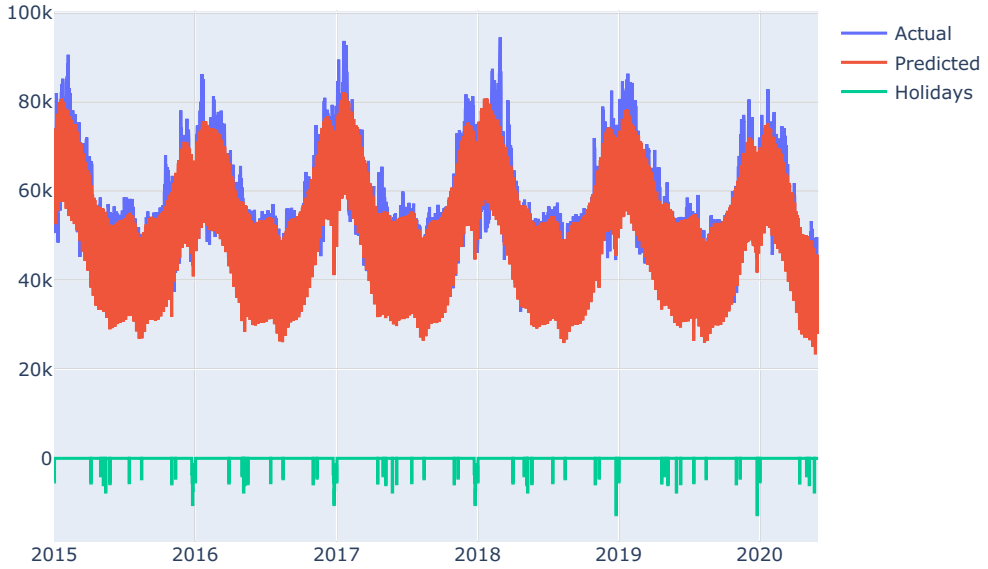


In [40]:

```
christmas = pd.DataFrame({
  'holiday': 'Christmas',
  'ds': pd.to_datetime(['2015-12-24','2016-12-24','2017-12-24','2021-12-24','2022-12-24']),  # Corrected years
  'lower_window': -1,
  'upper_window': 7,
})

m = Prophet(holidays=christmas)
m.add_country_holidays(country_name='FR')
m.fit(dfx)
future = m.make_future_dataframe(periods=0, freq='D')
forecast = m.predict(future)

fig = go.Figure()
# Create and style traces
fig.add_trace(go.Scatter(x=dfx['ds'], y=dfx['y'], name='Actual',))
```

```
fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['yhat'], name='Predicted',))
fig.add_trace(go.Scatter(x=forecast['ds'], y=forecast['holidays'], name='Holidays',))
fig.show()
```

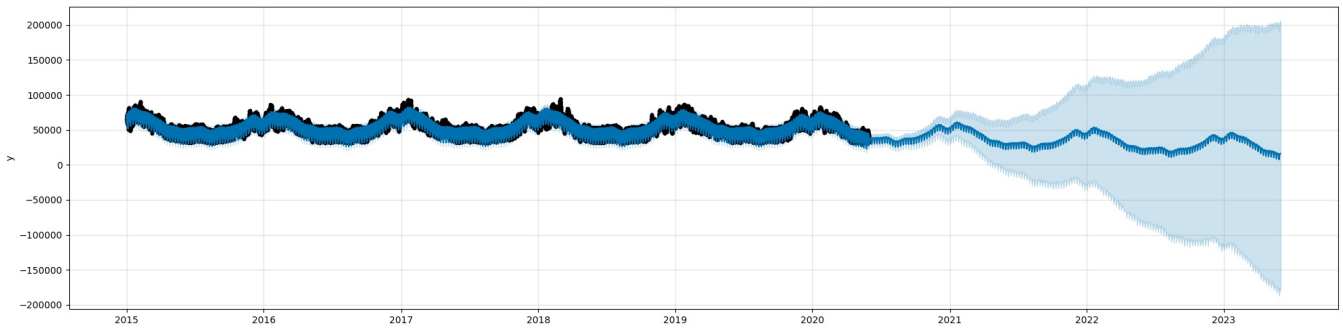
```
11:31:02 - cmdstanpy - INFO - Chain [1] start processing
11:32:05 - cmdstanpy - INFO - Chain [1] done processing
```



Outliers

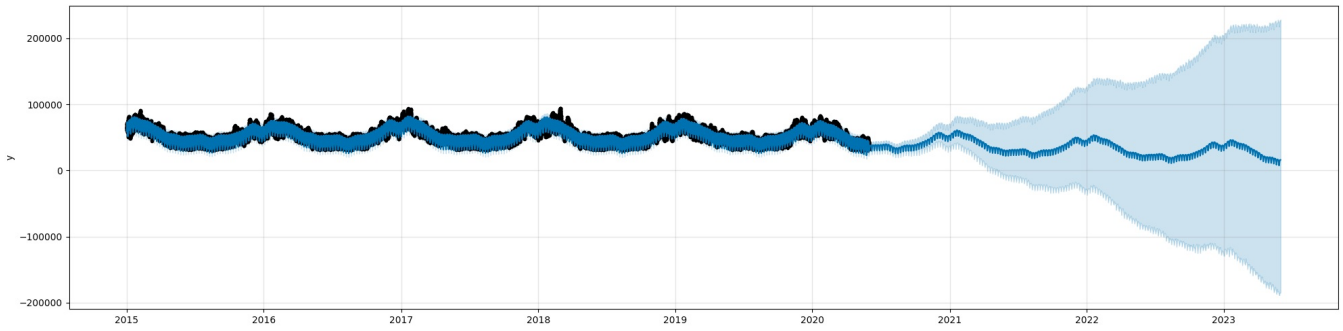
In [41]:
```
m = Prophet()
m.fit(dfx)
future = m.make_future_dataframe(periods=1096)
forecast = m.predict(future)
fig = m.plot(forecast, figsize=(20,5), xlabel = '')
```

```
11:32:19 - cmdstanpy - INFO - Chain [1] start processing
11:33:03 - cmdstanpy - INFO - Chain [1] done processing
```



In [42]:
```
df.loc[(dfx['ds'] > '2018-01-01') & (dfx['ds'] < '2018-01-01'), 'y'] = None
model = Prophet().fit(dfx)
fig = model.plot(model.predict(future), figsize=(20,5), xlabel = '')
plt.show()
```

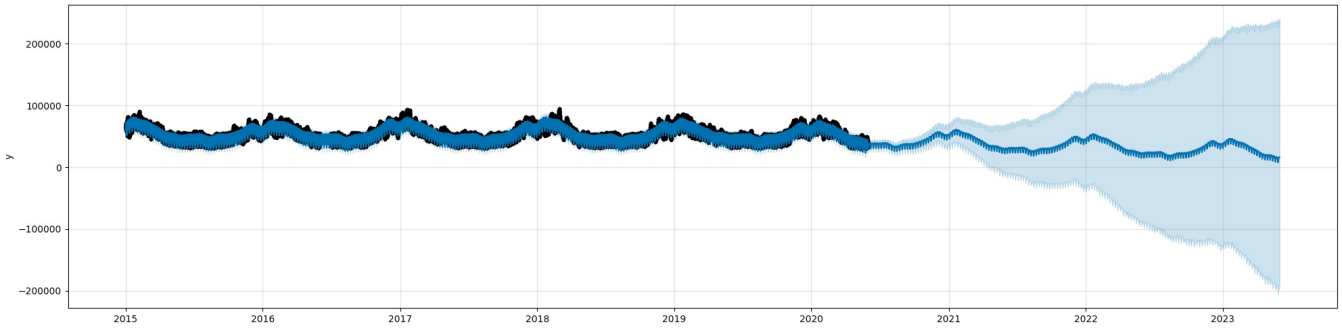
```
11:33:16 - cmdstanpy - INFO - Chain [1] start processing
11:34:01 - cmdstanpy - INFO - Chain [1] done processing
```



In [43]:
```
dfx.loc[(dfx['ds'] > '2016-06-01') & (dfx['ds'] < '2016-06-30'), 'y'] = None
```

```
m = Prophet().fit(dfx)
forecast = m.predict(future)
fig = m.plot(forecast, figsize=(20,5), xlabel = '')
plt.show()
```

```
11:34:13 - cmdstanpy - INFO - Chain [1] start processing
11:35:01 - cmdstanpy - INFO - Chain [1] done processing
```



Performance evaluation

In [44]:
```
%%time
df_cv = cross_validation(m, initial = '1000 days', period = '30 days', horizon='30 days')
```

```
11:35:15 - cmdstanpy - INFO - Chain [1] start processing
11:35:50 - cmdstanpy - INFO - Chain [1] done processing
11:35:52 - cmdstanpy - INFO - Chain [1] start processing
11:36:13 - cmdstanpy - INFO - Chain [1] done processing
11:36:15 - cmdstanpy - INFO - Chain [1] start processing
11:36:34 - cmdstanpy - INFO - Chain [1] done processing
11:36:36 - cmdstanpy - INFO - Chain [1] start processing
11:37:28 - cmdstanpy - INFO - Chain [1] done processing
11:37:30 - cmdstanpy - INFO - Chain [1] start processing
11:37:59 - cmdstanpy - INFO - Chain [1] done processing
11:38:01 - cmdstanpy - INFO - Chain [1] start processing
11:38:35 - cmdstanpy - INFO - Chain [1] done processing
11:38:37 - cmdstanpy - INFO - Chain [1] start processing
11:39:15 - cmdstanpy - INFO - Chain [1] done processing
11:39:18 - cmdstanpy - INFO - Chain [1] start processing
11:39:58 - cmdstanpy - INFO - Chain [1] done processing
11:40:01 - cmdstanpy - INFO - Chain [1] start processing
11:40:36 - cmdstanpy - INFO - Chain [1] done processing
11:40:38 - cmdstanpy - INFO - Chain [1] start processing
11:41:19 - cmdstanpy - INFO - Chain [1] done processing
11:41:22 - cmdstanpy - INFO - Chain [1] start processing
11:42:00 - cmdstanpy - INFO - Chain [1] done processing
11:42:03 - cmdstanpy - INFO - Chain [1] start processing
11:42:46 - cmdstanpy - INFO - Chain [1] done processing
11:42:49 - cmdstanpy - INFO - Chain [1] start processing
11:43:37 - cmdstanpy - INFO - Chain [1] done processing
11:43:39 - cmdstanpy - INFO - Chain [1] start processing
11:44:19 - cmdstanpy - INFO - Chain [1] done processing
11:44:22 - cmdstanpy - INFO - Chain [1] start processing
11:45:05 - cmdstanpy - INFO - Chain [1] done processing
11:45:08 - cmdstanpy - INFO - Chain [1] start processing
11:45:29 - cmdstanpy - INFO - Chain [1] done processing
11:45:32 - cmdstanpy - INFO - Chain [1] start processing
11:46:27 - cmdstanpy - INFO - Chain [1] done processing
11:46:30 - cmdstanpy - INFO - Chain [1] start processing
11:47:04 - cmdstanpy - INFO - Chain [1] done processing
11:47:07 - cmdstanpy - INFO - Chain [1] start processing
11:48:06 - cmdstanpy - INFO - Chain [1] done processing
11:48:09 - cmdstanpy - INFO - Chain [1] start processing
11:49:03 - cmdstanpy - INFO - Chain [1] done processing
11:49:06 - cmdstanpy - INFO - Chain [1] start processing
11:49:49 - cmdstanpy - INFO - Chain [1] done processing
11:49:52 - cmdstanpy - INFO - Chain [1] start processing
11:50:48 - cmdstanpy - INFO - Chain [1] done processing
11:50:51 - cmdstanpy - INFO - Chain [1] start processing
11:51:31 - cmdstanpy - INFO - Chain [1] done processing
11:51:34 - cmdstanpy - INFO - Chain [1] start processing
11:52:39 - cmdstanpy - INFO - Chain [1] done processing
11:52:42 - cmdstanpy - INFO - Chain [1] start processing
11:53:47 - cmdstanpy - INFO - Chain [1] done processing
11:53:50 - cmdstanpy - INFO - Chain [1] start processing
11:54:33 - cmdstanpy - INFO - Chain [1] done processing
11:54:37 - cmdstanpy - INFO - Chain [1] start processing
11:56:03 - cmdstanpy - INFO - Chain [1] done processing
11:56:06 - cmdstanpy - INFO - Chain [1] start processing
11:56:57 - cmdstanpy - INFO - Chain [1] done processing
11:57:00 - cmdstanpy - INFO - Chain [1] start processing
11:57:38 - cmdstanpy - INFO - Chain [1] done processing
11:57:42 - cmdstanpy - INFO - Chain [1] start processing

11:58:32 - cmdstanpy - INFO - Chain [1] done processing
11:58:35 - cmdstanpy - INFO - Chain [1] start processing
11:59:26 - cmdstanpy - INFO - Chain [1] done processing
11:59:29 - cmdstanpy - INFO - Chain [1] start processing
12:00:14 - cmdstanpy - INFO - Chain [1] done processing
CPU times: user 1min 35s, sys: 10.1 s, total: 1min 45s
Wall time: 25min 4s
```

In [45]:
```
df_cv.head(10)
```

Out[45]:

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2017-10-14 00:00:00	45985.920428	40915.899778	50343.826670	40841.0	2017-10-13 23:00:00
1	2017-10-14 01:00:00	43126.352534	38235.520638	47814.010555	38322.0	2017-10-13 23:00:00
2	2017-10-14 02:00:00	41462.887654	36832.273841	46047.931178	37116.0	2017-10-13 23:00:00
3	2017-10-14 03:00:00	41974.082429	36979.516090	46431.043805	37776.0	2017-10-13 23:00:00
4	2017-10-14 04:00:00	44486.757450	39819.806833	49057.534527	39725.0	2017-10-13 23:00:00
5	2017-10-14 05:00:00	47796.154677	42998.231318	52500.415869	42203.0	2017-10-13 23:00:00
6	2017-10-14 06:00:00	50576.896156	45768.433103	55646.226693	44032.0	2017-10-13 23:00:00
7	2017-10-14 07:00:00	52251.269991	47421.566352	57034.628808	46258.0	2017-10-13 23:00:00
8	2017-10-14 08:00:00	53075.558629	48300.751713	57745.139577	46989.0	2017-10-13 23:00:00
9	2017-10-14 09:00:00	53494.306507	49100.526267	58287.173859	47391.0	2017-10-13 23:00:00

In [46]:
```
df_p = performance_metrics(df_cv)
df_p.head(5)
```

Out[46]:

	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	3 days 01:00:00	1.890077e+07	4347.501704	3374.094764	0.062788	0.050428	0.062564	0.774549
1	3 days 02:00:00	1.890435e+07	4347.912973	3380.026526	0.062929	0.050813	0.062695	0.774889
2	3 days 03:00:00	1.889782e+07	4347.161764	3382.756209	0.063008	0.050966	0.062761	0.773434
3	3 days 04:00:00	1.889078e+07	4346.352879	3384.256093	0.063055	0.050966	0.062805	0.772809
4	3 days 05:00:00	1.889216e+07	4346.511017	3385.414789	0.063077	0.050954	0.062836	0.773149

Full pipeline

In [47]:
```
!!pip install xlrd
```

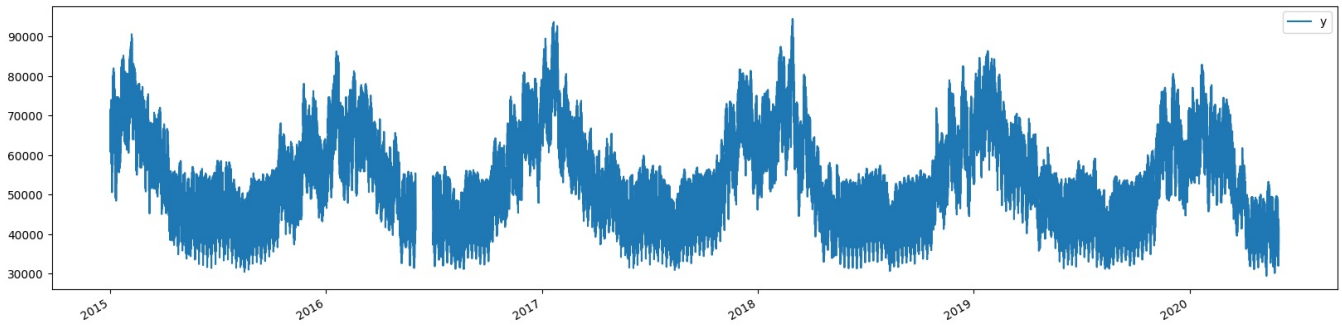
Out[47]:
```
['Collecting xlrd',
 '  Downloading xlrd-2.0.1-py2.py3-none-any.whl.metadata (3.4 kB)',
 'Downloading xlrd-2.0.1-py2.py3-none-any.whl (96 kB)',
 '\x1b[?25l   \x1b[90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━\x1b[0m \x1b[32m0.0/96.5 kB\x1b[0m \x1b[31m?\x1b[
0m eta \x1b[36m-:--:--\x1b[0m',
 '\x1b[2K   \x1b[90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━\x1b[0m \x1b[32m96.5/96.5 kB\x1b[0m \x1b[31m2.9 MB/
s\x1b[0m eta \x1b[36m0:00:00\x1b[0m',
 "\x1b[?25h\x1b[33mWARNING: Error parsing requirements for aiohttp: [Errno 2] No such file or directory: '/opt/
conda/lib/python3.10/site-packages/aiohttp-3.9.1.dist-info/METADATA'\x1b[0m\x1b[33m",
 '\x1b[0mInstalling collected packages: xlrd',
 'Successfully installed xlrd-2.0.1']
```

In [48]:
```
dfx.head()
```

Out[48]:

	ds	y
0	2015-01-01 00:00:00	70929.0
1	2015-01-01 01:00:00	69773.0
2	2015-01-01 02:00:00	66417.0
3	2015-01-01 03:00:00	64182.0
4	2015-01-01 04:00:00	63859.0

In [49]:
```
dfx.set_index('ds').plot(figsize=(20,5), xlabel = '');
```



In [50]:
```
min(dfx['ds']), max(dfx['ds'])
```

Out[50]:
```
(Timestamp('2015-01-01 00:00:00'), Timestamp('2020-05-30 23:00:00'))
```

In [51]:
```
df_train = dfx.loc[dfx['ds'] < '2019-05-01']
df_valid = dfx.loc[dfx['ds'] >= '2019-05-01']
print(df_train.shape, df_valid.shape)
```

```
(37897, 2) (9486, 2)
```

In [52]:
```
import itertools
param_grid = {
    # tuning those parameters can potentially improve the performance of our model
    'changepoint_prior_scale': [0.001, 0.1],
#    'seasonality_prior_scale': [0.01, 1.0],
#    'holidays_prior_scale': [0.01, 0.1],
    'seasonality_mode': ['additive', 'multiplicative'],
}

# Generate all combinations of parameters
all_params = [dict(zip(param_grid.keys(), v)) for v in itertools.product(*param_grid.values())]
rmses = []

# Quick peek at what our combinations look like
all_params
```

Out[52]:
```
[{'changepoint_prior_scale': 0.001, 'seasonality_mode': 'additive'},
 {'changepoint_prior_scale': 0.001, 'seasonality_mode': 'multiplicative'},
 {'changepoint_prior_scale': 0.1, 'seasonality_mode': 'additive'},
 {'changepoint_prior_scale': 0.1, 'seasonality_mode': 'multiplicative'}]
```

In [53]:
```
%%time

for params in all_params:
    m = Prophet(**params).fit(dfx)
    df_cv = cross_validation(m, initial = '100 days', period = '30 days', horizon='30 days', parallel="processe
    df_p = performance_metrics(df_cv, rolling_window=1)
    rmses.append(df_p['rmse'].values[0])

# Find the best parameters
tuning_results = pd.DataFrame(all_params)
tuning_results['rmse'] = rmses
print(tuning_results)

# Python
best_params = all_params[np.argmin(rmses)]
print(best_params)
```

```
12:00:37 - cmdstanpy - INFO - Chain [1] start processing
12:00:48 - cmdstanpy - INFO - Chain [1] done processing
12:00:55 - cmdstanpy - INFO - Chain [1] start processing
12:00:55 - cmdstanpy - INFO - Chain [1] start processing
12:00:55 - cmdstanpy - INFO - Chain [1] start processing
12:00:55 - cmdstanpy - INFO - Chain [1] done processing
12:00:55 - cmdstanpy - INFO - Chain [1] start processing
12:00:55 - cmdstanpy - INFO - Chain [1] done processing
12:00:56 - cmdstanpy - INFO - Chain [1] done processing
12:00:56 - cmdstanpy - INFO - Chain [1] done processing
12:00:57 - cmdstanpy - INFO - Chain [1] start processing
12:00:57 - cmdstanpy - INFO - Chain [1] start processing
12:00:57 - cmdstanpy - INFO - Chain [1] done processing
12:00:58 - cmdstanpy - INFO - Chain [1] start processing
12:00:58 - cmdstanpy - INFO - Chain [1] done processing
12:00:58 - cmdstanpy - INFO - Chain [1] start processing
12:00:58 - cmdstanpy - INFO - Chain [1] done processing
12:00:59 - cmdstanpy - INFO - Chain [1] done processing
12:00:59 - cmdstanpy - INFO - Chain [1] start processing
12:01:00 - cmdstanpy - INFO - Chain [1] start processing
12:01:00 - cmdstanpy - INFO - Chain [1] done processing
12:01:00 - cmdstanpy - INFO - Chain [1] start processing
12:01:01 - cmdstanpy - INFO - Chain [1] start processing
12:01:01 - cmdstanpy - INFO - Chain [1] done processing
12:01:02 - cmdstanpy - INFO - Chain [1] done processing
12:01:02 - cmdstanpy - INFO - Chain [1] start processing
12:01:03 - cmdstanpy - INFO - Chain [1] start processing
12:01:04 - cmdstanpy - INFO - Chain [1] start processing
12:01:04 - cmdstanpy - INFO - Chain [1] done processing
12:01:04 - cmdstanpy - INFO - Chain [1] done processing
12:01:05 - cmdstanpy - INFO - Chain [1] done processing
12:01:06 - cmdstanpy - INFO - Chain [1] done processing
12:01:06 - cmdstanpy - INFO - Chain [1] start processing
12:01:07 - cmdstanpy - INFO - Chain [1] start processing
12:01:07 - cmdstanpy - INFO - Chain [1] start processing
12:01:08 - cmdstanpy - INFO - Chain [1] start processing
12:01:08 - cmdstanpy - INFO - Chain [1] done processing
12:01:09 - cmdstanpy - INFO - Chain [1] done processing
12:01:10 - cmdstanpy - INFO - Chain [1] done processing
12:01:10 - cmdstanpy - INFO - Chain [1] done processing
12:01:11 - cmdstanpy - INFO - Chain [1] start processing
12:01:12 - cmdstanpy - INFO - Chain [1] start processing
12:01:12 - cmdstanpy - INFO - Chain [1] start processing
12:01:13 - cmdstanpy - INFO - Chain [1] start processing
12:01:15 - cmdstanpy - INFO - Chain [1] done processing
12:01:15 - cmdstanpy - INFO - Chain [1] done processing
12:01:16 - cmdstanpy - INFO - Chain [1] done processing
12:01:17 - cmdstanpy - INFO - Chain [1] done processing
12:01:18 - cmdstanpy - INFO - Chain [1] start processing
12:01:18 - cmdstanpy - INFO - Chain [1] start processing
12:01:18 - cmdstanpy - INFO - Chain [1] start processing
12:01:20 - cmdstanpy - INFO - Chain [1] done processing
12:01:21 - cmdstanpy - INFO - Chain [1] start processing
12:01:22 - cmdstanpy - INFO - Chain [1] done processing
12:01:23 - cmdstanpy - INFO - Chain [1] done processing
12:01:23 - cmdstanpy - INFO - Chain [1] start processing
12:01:25 - cmdstanpy - INFO - Chain [1] done processing
12:01:25 - cmdstanpy - INFO - Chain [1] start processing
12:01:27 - cmdstanpy - INFO - Chain [1] start processing
12:01:28 - cmdstanpy - INFO - Chain [1] start processing
12:01:29 - cmdstanpy - INFO - Chain [1] done processing
12:01:30 - cmdstanpy - INFO - Chain [1] done processing
12:01:33 - cmdstanpy - INFO - Chain [1] start processing
12:01:34 - cmdstanpy - INFO - Chain [1] start processing
12:01:34 - cmdstanpy - INFO - Chain [1] done processing
12:01:35 - cmdstanpy - INFO - Chain [1] done processing
12:01:38 - cmdstanpy - INFO - Chain [1] start processing
```

```
12:01:38 - cmdstanpy - INFO - Chain [1] done processing
12:01:39 - cmdstanpy - INFO - Chain [1] start processing
12:01:40 - cmdstanpy - INFO - Chain [1] done processing
12:01:42 - cmdstanpy - INFO - Chain [1] start processing
12:01:42 - cmdstanpy - INFO - Chain [1] done processing
12:01:43 - cmdstanpy - INFO - Chain [1] done processing
12:01:44 - cmdstanpy - INFO - Chain [1] start processing
12:01:46 - cmdstanpy - INFO - Chain [1] start processing
12:01:46 - cmdstanpy - INFO - Chain [1] start processing
12:01:47 - cmdstanpy - INFO - Chain [1] done processing
12:01:49 - cmdstanpy - INFO - Chain [1] done processing
12:01:51 - cmdstanpy - INFO - Chain [1] start processing
12:01:54 - cmdstanpy - INFO - Chain [1] start processing
12:01:54 - cmdstanpy - INFO - Chain [1] done processing
12:01:58 - cmdstanpy - INFO - Chain [1] start processing
12:01:59 - cmdstanpy - INFO - Chain [1] done processing
12:01:59 - cmdstanpy - INFO - Chain [1] done processing
12:02:00 - cmdstanpy - INFO - Chain [1] done processing
12:02:03 - cmdstanpy - INFO - Chain [1] start processing
12:02:04 - cmdstanpy - INFO - Chain [1] start processing
12:02:04 - cmdstanpy - INFO - Chain [1] start processing
12:02:05 - cmdstanpy - INFO - Chain [1] done processing
12:02:10 - cmdstanpy - INFO - Chain [1] start processing
12:02:11 - cmdstanpy - INFO - Chain [1] done processing
12:02:13 - cmdstanpy - INFO - Chain [1] done processing
12:02:15 - cmdstanpy - INFO - Chain [1] done processing
12:02:16 - cmdstanpy - INFO - Chain [1] start processing
12:02:18 - cmdstanpy - INFO - Chain [1] start processing
12:02:19 - cmdstanpy - INFO - Chain [1] start processing
12:02:19 - cmdstanpy - INFO - Chain [1] done processing
12:02:23 - cmdstanpy - INFO - Chain [1] done processing
12:02:24 - cmdstanpy - INFO - Chain [1] start processing
12:02:24 - cmdstanpy - INFO - Chain [1] done processing
12:02:28 - cmdstanpy - INFO - Chain [1] start processing
12:02:30 - cmdstanpy - INFO - Chain [1] start processing
12:02:30 - cmdstanpy - INFO - Chain [1] done processing
12:02:35 - cmdstanpy - INFO - Chain [1] start processing
12:02:36 - cmdstanpy - INFO - Chain [1] done processing
12:02:40 - cmdstanpy - INFO - Chain [1] done processing
12:02:41 - cmdstanpy - INFO - Chain [1] start processing
12:02:41 - cmdstanpy - INFO - Chain [1] done processing
12:02:46 - cmdstanpy - INFO - Chain [1] start processing
12:02:47 - cmdstanpy - INFO - Chain [1] start processing
12:02:50 - cmdstanpy - INFO - Chain [1] done processing
12:02:51 - cmdstanpy - INFO - Chain [1] done processing
12:02:55 - cmdstanpy - INFO - Chain [1] start processing
12:02:56 - cmdstanpy - INFO - Chain [1] start processing
12:03:01 - cmdstanpy - INFO - Chain [1] done processing
12:03:03 - cmdstanpy - INFO - Chain [1] done processing
12:03:06 - cmdstanpy - INFO - Chain [1] done processing
12:03:07 - cmdstanpy - INFO - Chain [1] start processing
12:03:08 - cmdstanpy - INFO - Chain [1] done processing
12:03:08 - cmdstanpy - INFO - Chain [1] start processing
12:03:12 - cmdstanpy - INFO - Chain [1] start processing
12:03:13 - cmdstanpy - INFO - Chain [1] start processing
12:03:19 - cmdstanpy - INFO - Chain [1] done processing
12:03:20 - cmdstanpy - INFO - Chain [1] done processing
12:03:28 - cmdstanpy - INFO - Chain [1] done processing
12:03:28 - cmdstanpy - INFO - Chain [1] done processing
12:03:32 - cmdstanpy - INFO - Chain [1] start processing
12:03:41 - cmdstanpy - INFO - Chain [1] done processing
12:03:47 - cmdstanpy - INFO - Chain [1] start processing
12:03:47 - cmdstanpy - INFO - Chain [1] start processing
12:03:47 - cmdstanpy - INFO - Chain [1] start processing
12:03:48 - cmdstanpy - INFO - Chain [1] done processing
12:03:48 - cmdstanpy - INFO - Chain [1] start processing
12:03:48 - cmdstanpy - INFO - Chain [1] done processing
12:03:48 - cmdstanpy - INFO - Chain [1] done processing
12:03:49 - cmdstanpy - INFO - Chain [1] done processing
12:03:49 - cmdstanpy - INFO - Chain [1] start processing
12:03:50 - cmdstanpy - INFO - Chain [1] start processing
12:03:50 - cmdstanpy - INFO - Chain [1] start processing
12:03:50 - cmdstanpy - INFO - Chain [1] done processing
12:03:51 - cmdstanpy - INFO - Chain [1] start processing
12:03:51 - cmdstanpy - INFO - Chain [1] done processing
12:03:51 - cmdstanpy - INFO - Chain [1] done processing
12:03:52 - cmdstanpy - INFO - Chain [1] done processing
12:03:52 - cmdstanpy - INFO - Chain [1] start processing
12:03:53 - cmdstanpy - INFO - Chain [1] start processing
12:03:53 - cmdstanpy - INFO - Chain [1] done processing
12:03:53 - cmdstanpy - INFO - Chain [1] start processing
12:03:54 - cmdstanpy - INFO - Chain [1] start processing
12:03:55 - cmdstanpy - INFO - Chain [1] done processing
12:03:55 - cmdstanpy - INFO - Chain [1] done processing
12:03:56 - cmdstanpy - INFO - Chain [1] start processing
12:03:56 - cmdstanpy - INFO - Chain [1] done processing
12:03:57 - cmdstanpy - INFO - Chain [1] start processing
12:03:58 - cmdstanpy - INFO - Chain [1] start processing
12:03:58 - cmdstanpy - INFO - Chain [1] start processing
```

```
12:03:58 - cmdstanpy - INFO - Chain [1] done processing
12:04:00 - cmdstanpy - INFO - Chain [1] done processing
12:04:01 - cmdstanpy - INFO - Chain [1] start processing
12:04:01 - cmdstanpy - INFO - Chain [1] done processing
12:04:02 - cmdstanpy - INFO - Chain [1] done processing
12:04:03 - cmdstanpy - INFO - Chain [1] done processing
12:04:03 - cmdstanpy - INFO - Chain [1] start processing
12:04:03 - cmdstanpy - INFO - Chain [1] start processing
12:04:04 - cmdstanpy - INFO - Chain [1] start processing
12:04:05 - cmdstanpy - INFO - Chain [1] start processing
12:04:06 - cmdstanpy - INFO - Chain [1] done processing
12:04:07 - cmdstanpy - INFO - Chain [1] done processing
12:04:08 - cmdstanpy - INFO - Chain [1] done processing
12:04:09 - cmdstanpy - INFO - Chain [1] done processing
12:04:09 - cmdstanpy - INFO - Chain [1] start processing
12:04:09 - cmdstanpy - INFO - Chain [1] start processing
12:04:11 - cmdstanpy - INFO - Chain [1] start processing
12:04:11 - cmdstanpy - INFO - Chain [1] start processing
12:04:12 - cmdstanpy - INFO - Chain [1] done processing
12:04:13 - cmdstanpy - INFO - Chain [1] done processing
12:04:15 - cmdstanpy - INFO - Chain [1] start processing
12:04:15 - cmdstanpy - INFO - Chain [1] done processing
12:04:16 - cmdstanpy - INFO - Chain [1] done processing
12:04:16 - cmdstanpy - INFO - Chain [1] start processing
12:04:18 - cmdstanpy - INFO - Chain [1] start processing
12:04:19 - cmdstanpy - INFO - Chain [1] done processing
12:04:19 - cmdstanpy - INFO - Chain [1] start processing
12:04:22 - cmdstanpy - INFO - Chain [1] done processing
12:04:22 - cmdstanpy - INFO - Chain [1] start processing
12:04:25 - cmdstanpy - INFO - Chain [1] done processing
12:04:26 - cmdstanpy - INFO - Chain [1] start processing
12:04:26 - cmdstanpy - INFO - Chain [1] done processing
12:04:26 - cmdstanpy - INFO - Chain [1] done processing
12:04:28 - cmdstanpy - INFO - Chain [1] start processing
12:04:29 - cmdstanpy - INFO - Chain [1] start processing
12:04:30 - cmdstanpy - INFO - Chain [1] start processing
12:04:31 - cmdstanpy - INFO - Chain [1] done processing
12:04:35 - cmdstanpy - INFO - Chain [1] start processing
12:04:35 - cmdstanpy - INFO - Chain [1] done processing
12:04:36 - cmdstanpy - INFO - Chain [1] done processing
12:04:39 - cmdstanpy - INFO - Chain [1] start processing
12:04:40 - cmdstanpy - INFO - Chain [1] done processing
12:04:40 - cmdstanpy - INFO - Chain [1] start processing
12:04:44 - cmdstanpy - INFO - Chain [1] start processing
12:04:45 - cmdstanpy - INFO - Chain [1] done processing
12:04:48 - cmdstanpy - INFO - Chain [1] done processing
12:04:49 - cmdstanpy - INFO - Chain [1] start processing
12:04:50 - cmdstanpy - INFO - Chain [1] done processing
12:04:50 - cmdstanpy - INFO - Chain [1] done processing
12:04:52 - cmdstanpy - INFO - Chain [1] start processing
12:04:54 - cmdstanpy - INFO - Chain [1] start processing
12:04:55 - cmdstanpy - INFO - Chain [1] start processing
12:04:55 - cmdstanpy - INFO - Chain [1] done processing
12:04:58 - cmdstanpy - INFO - Chain [1] done processing
12:05:00 - cmdstanpy - INFO - Chain [1] start processing
12:05:01 - cmdstanpy - INFO - Chain [1] done processing
12:05:02 - cmdstanpy - INFO - Chain [1] start processing
12:05:02 - cmdstanpy - INFO - Chain [1] done processing
12:05:05 - cmdstanpy - INFO - Chain [1] start processing
12:05:06 - cmdstanpy - INFO - Chain [1] start processing
12:05:08 - cmdstanpy - INFO - Chain [1] done processing
12:05:13 - cmdstanpy - INFO - Chain [1] start processing
12:05:14 - cmdstanpy - INFO - Chain [1] done processing
12:05:16 - cmdstanpy - INFO - Chain [1] done processing
12:05:16 - cmdstanpy - INFO - Chain [1] done processing
12:05:18 - cmdstanpy - INFO - Chain [1] done processing
12:05:19 - cmdstanpy - INFO - Chain [1] start processing
12:05:20 - cmdstanpy - INFO - Chain [1] start processing
12:05:21 - cmdstanpy - INFO - Chain [1] start processing
12:05:22 - cmdstanpy - INFO - Chain [1] start processing
12:05:30 - cmdstanpy - INFO - Chain [1] done processing
12:05:30 - cmdstanpy - INFO - Chain [1] done processing
12:05:32 - cmdstanpy - INFO - Chain [1] done processing
12:05:34 - cmdstanpy - INFO - Chain [1] done processing
12:05:35 - cmdstanpy - INFO - Chain [1] start processing
12:05:35 - cmdstanpy - INFO - Chain [1] start processing
12:05:37 - cmdstanpy - INFO - Chain [1] start processing
12:05:38 - cmdstanpy - INFO - Chain [1] start processing
12:05:47 - cmdstanpy - INFO - Chain [1] done processing
12:05:48 - cmdstanpy - INFO - Chain [1] done processing
12:05:50 - cmdstanpy - INFO - Chain [1] done processing
12:05:52 - cmdstanpy - INFO - Chain [1] start processing
12:05:53 - cmdstanpy - INFO - Chain [1] start processing
12:05:55 - cmdstanpy - INFO - Chain [1] start processing
12:06:00 - cmdstanpy - INFO - Chain [1] done processing
12:06:06 - cmdstanpy - INFO - Chain [1] start processing
12:06:08 - cmdstanpy - INFO - Chain [1] done processing
12:06:09 - cmdstanpy - INFO - Chain [1] done processing
12:06:12 - cmdstanpy - INFO - Chain [1] done processing
```

```
12:06:14 - cmdstanpy - INFO - Chain [1] start processing
12:06:15 - cmdstanpy - INFO - Chain [1] start processing
12:06:18 - cmdstanpy - INFO - Chain [1] start processing
12:06:20 - cmdstanpy - INFO - Chain [1] done processing
12:06:29 - cmdstanpy - INFO - Chain [1] done processing
12:06:31 - cmdstanpy - INFO - Chain [1] done processing
12:06:31 - cmdstanpy - INFO - Chain [1] done processing
12:06:35 - cmdstanpy - INFO - Chain [1] start processing
12:07:40 - cmdstanpy - INFO - Chain [1] done processing
12:07:46 - cmdstanpy - INFO - Chain [1] start processing
12:07:46 - cmdstanpy - INFO - Chain [1] start processing
12:07:46 - cmdstanpy - INFO - Chain [1] start processing
12:07:46 - cmdstanpy - INFO - Chain [1] start processing
12:07:49 - cmdstanpy - INFO - Chain [1] done processing
12:07:51 - cmdstanpy - INFO - Chain [1] start processing
12:07:52 - cmdstanpy - INFO - Chain [1] done processing
12:07:54 - cmdstanpy - INFO - Chain [1] start processing
12:07:54 - cmdstanpy - INFO - Chain [1] done processing
12:07:56 - cmdstanpy - INFO - Chain [1] done processing
12:07:57 - cmdstanpy - INFO - Chain [1] start processing
12:07:58 - cmdstanpy - INFO - Chain [1] start processing
12:07:58 - cmdstanpy - INFO - Chain [1] done processing
12:08:00 - cmdstanpy - INFO - Chain [1] start processing
12:08:05 - cmdstanpy - INFO - Chain [1] done processing
12:08:07 - cmdstanpy - INFO - Chain [1] start processing
12:08:12 - cmdstanpy - INFO - Chain [1] done processing
12:08:12 - cmdstanpy - INFO - Chain [1] done processing
12:08:13 - cmdstanpy - INFO - Chain [1] done processing
12:08:14 - cmdstanpy - INFO - Chain [1] start processing
12:08:14 - cmdstanpy - INFO - Chain [1] start processing
12:08:15 - cmdstanpy - INFO - Chain [1] start processing
12:08:29 - cmdstanpy - INFO - Chain [1] done processing
12:08:32 - cmdstanpy - INFO - Chain [1] start processing
12:08:36 - cmdstanpy - INFO - Chain [1] done processing
12:08:36 - cmdstanpy - INFO - Chain [1] done processing
12:08:38 - cmdstanpy - INFO - Chain [1] start processing
12:08:38 - cmdstanpy - INFO - Chain [1] start processing
12:09:04 - cmdstanpy - INFO - Chain [1] done processing
12:09:04 - cmdstanpy - INFO - Chain [1] done processing
12:09:07 - cmdstanpy - INFO - Chain [1] start processing
12:09:07 - cmdstanpy - INFO - Chain [1] start processing
12:09:12 - cmdstanpy - INFO - Chain [1] done processing
12:09:13 - cmdstanpy - INFO - Chain [1] done processing
12:09:15 - cmdstanpy - INFO - Chain [1] start processing
12:09:16 - cmdstanpy - INFO - Chain [1] start processing
12:09:42 - cmdstanpy - INFO - Chain [1] done processing
12:09:45 - cmdstanpy - INFO - Chain [1] start processing
12:09:55 - cmdstanpy - INFO - Chain [1] done processing
12:09:57 - cmdstanpy - INFO - Chain [1] done processing
12:09:58 - cmdstanpy - INFO - Chain [1] start processing
12:10:00 - cmdstanpy - INFO - Chain [1] start processing
12:10:11 - cmdstanpy - INFO - Chain [1] done processing
12:10:14 - cmdstanpy - INFO - Chain [1] start processing
12:10:28 - cmdstanpy - INFO - Chain [1] done processing
12:10:30 - cmdstanpy - INFO - Chain [1] done processing
12:10:32 - cmdstanpy - INFO - Chain [1] start processing
12:10:34 - cmdstanpy - INFO - Chain [1] start processing
12:10:44 - cmdstanpy - INFO - Chain [1] done processing
12:10:47 - cmdstanpy - INFO - Chain [1] done processing
12:10:48 - cmdstanpy - INFO - Chain [1] start processing
12:10:50 - cmdstanpy - INFO - Chain [1] start processing
12:11:13 - cmdstanpy - INFO - Chain [1] done processing
12:11:16 - cmdstanpy - INFO - Chain [1] done processing
12:11:16 - cmdstanpy - INFO - Chain [1] start processing
12:11:17 - cmdstanpy - INFO - Chain [1] done processing
12:11:19 - cmdstanpy - INFO - Chain [1] start processing
12:11:21 - cmdstanpy - INFO - Chain [1] start processing
12:11:55 - cmdstanpy - INFO - Chain [1] done processing
12:11:59 - cmdstanpy - INFO - Chain [1] start processing
12:12:25 - cmdstanpy - INFO - Chain [1] done processing
12:12:29 - cmdstanpy - INFO - Chain [1] start processing
12:12:33 - cmdstanpy - INFO - Chain [1] done processing
12:12:36 - cmdstanpy - INFO - Chain [1] start processing
12:12:46 - cmdstanpy - INFO - Chain [1] done processing
12:12:50 - cmdstanpy - INFO - Chain [1] start processing
12:13:08 - cmdstanpy - INFO - Chain [1] done processing
12:13:12 - cmdstanpy - INFO - Chain [1] start processing
12:13:28 - cmdstanpy - INFO - Chain [1] done processing
12:13:32 - cmdstanpy - INFO - Chain [1] start processing
12:13:36 - cmdstanpy - INFO - Chain [1] done processing
12:13:40 - cmdstanpy - INFO - Chain [1] start processing
12:14:19 - cmdstanpy - INFO - Chain [1] done processing
12:14:23 - cmdstanpy - INFO - Chain [1] start processing
12:14:32 - cmdstanpy - INFO - Chain [1] done processing
12:14:36 - cmdstanpy - INFO - Chain [1] start processing
12:14:44 - cmdstanpy - INFO - Chain [1] done processing
12:14:48 - cmdstanpy - INFO - Chain [1] start processing
12:15:11 - cmdstanpy - INFO - Chain [1] done processing
12:15:16 - cmdstanpy - INFO - Chain [1] start processing
```

```
12:15:32 - cmdstanpy - INFO - Chain [1] done processing
12:15:36 - cmdstanpy - INFO - Chain [1] start processing
12:15:48 - cmdstanpy - INFO - Chain [1] done processing
12:15:53 - cmdstanpy - INFO - Chain [1] start processing
12:15:59 - cmdstanpy - INFO - Chain [1] done processing
12:16:03 - cmdstanpy - INFO - Chain [1] start processing
12:17:01 - cmdstanpy - INFO - Chain [1] done processing
12:17:05 - cmdstanpy - INFO - Chain [1] start processing
12:17:07 - cmdstanpy - INFO - Chain [1] done processing
12:17:12 - cmdstanpy - INFO - Chain [1] start processing
12:17:14 - cmdstanpy - INFO - Chain [1] done processing
12:17:19 - cmdstanpy - INFO - Chain [1] start processing
12:18:05 - cmdstanpy - INFO - Chain [1] done processing
12:18:10 - cmdstanpy - INFO - Chain [1] start processing
12:18:28 - cmdstanpy - INFO - Chain [1] done processing
12:18:34 - cmdstanpy - INFO - Chain [1] start processing
12:18:48 - cmdstanpy - INFO - Chain [1] done processing
12:18:53 - cmdstanpy - INFO - Chain [1] start processing
12:19:03 - cmdstanpy - INFO - Chain [1] done processing
12:19:08 - cmdstanpy - INFO - Chain [1] start processing
12:20:06 - cmdstanpy - INFO - Chain [1] done processing
12:20:11 - cmdstanpy - INFO - Chain [1] start processing
12:20:15 - cmdstanpy - INFO - Chain [1] done processing
12:20:20 - cmdstanpy - INFO - Chain [1] start processing
12:20:40 - cmdstanpy - INFO - Chain [1] done processing
12:20:41 - cmdstanpy - INFO - Chain [1] done processing
12:20:45 - cmdstanpy - INFO - Chain [1] start processing
12:20:47 - cmdstanpy - INFO - Chain [1] start processing
12:21:34 - cmdstanpy - INFO - Chain [1] done processing
12:21:40 - cmdstanpy - INFO - Chain [1] start processing
12:22:14 - cmdstanpy - INFO - Chain [1] done processing
12:22:20 - cmdstanpy - INFO - Chain [1] start processing
12:22:29 - cmdstanpy - INFO - Chain [1] done processing
12:22:34 - cmdstanpy - INFO - Chain [1] start processing
12:22:36 - cmdstanpy - INFO - Chain [1] done processing
12:22:42 - cmdstanpy - INFO - Chain [1] start processing
12:23:38 - cmdstanpy - INFO - Chain [1] done processing
12:23:44 - cmdstanpy - INFO - Chain [1] start processing
12:24:12 - cmdstanpy - INFO - Chain [1] done processing
12:24:16 - cmdstanpy - INFO - Chain [1] done processing
12:24:17 - cmdstanpy - INFO - Chain [1] start processing
12:24:39 - cmdstanpy - INFO - Chain [1] done processing
12:25:11 - cmdstanpy - INFO - Chain [1] done processing
12:25:30 - cmdstanpy - INFO - Chain [1] done processing
12:25:34 - cmdstanpy - INFO - Chain [1] start processing
12:26:30 - cmdstanpy - INFO - Chain [1] done processing
12:26:38 - cmdstanpy - INFO - Chain [1] start processing
12:26:38 - cmdstanpy - INFO - Chain [1] start processing
12:26:38 - cmdstanpy - INFO - Chain [1] start processing
12:26:38 - cmdstanpy - INFO - Chain [1] start processing
12:26:47 - cmdstanpy - INFO - Chain [1] done processing
12:26:49 - cmdstanpy - INFO - Chain [1] start processing
12:26:49 - cmdstanpy - INFO - Chain [1] done processing
12:26:51 - cmdstanpy - INFO - Chain [1] start processing
12:26:55 - cmdstanpy - INFO - Chain [1] done processing
12:26:57 - cmdstanpy - INFO - Chain [1] done processing
12:26:58 - cmdstanpy - INFO - Chain [1] start processing
12:26:59 - cmdstanpy - INFO - Chain [1] start processing
12:27:14 - cmdstanpy - INFO - Chain [1] done processing
12:27:15 - cmdstanpy - INFO - Chain [1] done processing
12:27:16 - cmdstanpy - INFO - Chain [1] start processing
12:27:17 - cmdstanpy - INFO - Chain [1] start processing
12:27:19 - cmdstanpy - INFO - Chain [1] done processing
12:27:21 - cmdstanpy - INFO - Chain [1] start processing
12:27:28 - cmdstanpy - INFO - Chain [1] done processing
12:27:30 - cmdstanpy - INFO - Chain [1] start processing
12:27:31 - cmdstanpy - INFO - Chain [1] done processing
12:27:33 - cmdstanpy - INFO - Chain [1] start processing
12:27:36 - cmdstanpy - INFO - Chain [1] done processing
12:27:38 - cmdstanpy - INFO - Chain [1] start processing
12:27:39 - cmdstanpy - INFO - Chain [1] done processing
12:27:41 - cmdstanpy - INFO - Chain [1] start processing
12:27:52 - cmdstanpy - INFO - Chain [1] done processing
12:27:55 - cmdstanpy - INFO - Chain [1] start processing
12:27:55 - cmdstanpy - INFO - Chain [1] done processing
12:27:55 - cmdstanpy - INFO - Chain [1] done processing
12:27:57 - cmdstanpy - INFO - Chain [1] start processing
12:27:57 - cmdstanpy - INFO - Chain [1] start processing
12:28:13 - cmdstanpy - INFO - Chain [1] done processing
12:28:16 - cmdstanpy - INFO - Chain [1] start processing
12:28:16 - cmdstanpy - INFO - Chain [1] done processing
12:28:19 - cmdstanpy - INFO - Chain [1] start processing
12:28:21 - cmdstanpy - INFO - Chain [1] done processing
12:28:24 - cmdstanpy - INFO - Chain [1] start processing
12:28:37 - cmdstanpy - INFO - Chain [1] done processing
12:28:41 - cmdstanpy - INFO - Chain [1] start processing
12:28:56 - cmdstanpy - INFO - Chain [1] done processing
12:28:59 - cmdstanpy - INFO - Chain [1] start processing
12:29:07 - cmdstanpy - INFO - Chain [1] done processing
```

```
12:29:10 - cmdstanpy - INFO - Chain [1] start processing
12:29:10 - cmdstanpy - INFO - Chain [1] done processing
12:29:13 - cmdstanpy - INFO - Chain [1] start processing
12:29:20 - cmdstanpy - INFO - Chain [1] done processing
12:29:23 - cmdstanpy - INFO - Chain [1] start processing
12:29:48 - cmdstanpy - INFO - Chain [1] done processing
12:29:51 - cmdstanpy - INFO - Chain [1] done processing
12:29:51 - cmdstanpy - INFO - Chain [1] done processing
12:29:52 - cmdstanpy - INFO - Chain [1] start processing
12:29:54 - cmdstanpy - INFO - Chain [1] start processing
12:29:54 - cmdstanpy - INFO - Chain [1] start processing
12:30:09 - cmdstanpy - INFO - Chain [1] done processing
12:30:13 - cmdstanpy - INFO - Chain [1] start processing
12:30:56 - cmdstanpy - INFO - Chain [1] done processing
12:31:00 - cmdstanpy - INFO - Chain [1] start processing
12:31:01 - cmdstanpy - INFO - Chain [1] done processing
12:31:05 - cmdstanpy - INFO - Chain [1] start processing
12:31:05 - cmdstanpy - INFO - Chain [1] done processing
12:31:08 - cmdstanpy - INFO - Chain [1] start processing
12:31:21 - cmdstanpy - INFO - Chain [1] done processing
12:31:25 - cmdstanpy - INFO - Chain [1] start processing
12:32:10 - cmdstanpy - INFO - Chain [1] done processing
12:32:14 - cmdstanpy - INFO - Chain [1] start processing
12:32:15 - cmdstanpy - INFO - Chain [1] done processing
12:32:18 - cmdstanpy - INFO - Chain [1] start processing
12:32:31 - cmdstanpy - INFO - Chain [1] done processing
12:32:35 - cmdstanpy - INFO - Chain [1] start processing
12:32:52 - cmdstanpy - INFO - Chain [1] done processing
12:32:57 - cmdstanpy - INFO - Chain [1] start processing
12:33:22 - cmdstanpy - INFO - Chain [1] done processing
12:33:23 - cmdstanpy - INFO - Chain [1] done processing
12:33:26 - cmdstanpy - INFO - Chain [1] done processing
12:33:26 - cmdstanpy - INFO - Chain [1] start processing
12:33:28 - cmdstanpy - INFO - Chain [1] start processing
12:33:30 - cmdstanpy - INFO - Chain [1] start processing
12:34:04 - cmdstanpy - INFO - Chain [1] done processing
12:34:08 - cmdstanpy - INFO - Chain [1] start processing
12:34:28 - cmdstanpy - INFO - Chain [1] done processing
12:34:33 - cmdstanpy - INFO - Chain [1] start processing
12:34:42 - cmdstanpy - INFO - Chain [1] done processing
12:34:47 - cmdstanpy - INFO - Chain [1] start processing
12:34:53 - cmdstanpy - INFO - Chain [1] done processing
12:34:57 - cmdstanpy - INFO - Chain [1] start processing
12:35:54 - cmdstanpy - INFO - Chain [1] done processing
12:35:56 - cmdstanpy - INFO - Chain [1] done processing
12:35:59 - cmdstanpy - INFO - Chain [1] start processing
12:35:59 - cmdstanpy - INFO - Chain [1] done processing
12:36:01 - cmdstanpy - INFO - Chain [1] start processing
12:36:04 - cmdstanpy - INFO - Chain [1] start processing
12:36:23 - cmdstanpy - INFO - Chain [1] done processing
12:36:28 - cmdstanpy - INFO - Chain [1] start processing
12:37:10 - cmdstanpy - INFO - Chain [1] done processing
12:37:14 - cmdstanpy - INFO - Chain [1] done processing
12:37:16 - cmdstanpy - INFO - Chain [1] start processing
12:37:18 - cmdstanpy - INFO - Chain [1] start processing
12:37:55 - cmdstanpy - INFO - Chain [1] done processing
12:37:57 - cmdstanpy - INFO - Chain [1] done processing
12:38:01 - cmdstanpy - INFO - Chain [1] start processing
12:38:02 - cmdstanpy - INFO - Chain [1] start processing
12:38:44 - cmdstanpy - INFO - Chain [1] done processing
12:38:50 - cmdstanpy - INFO - Chain [1] start processing
12:39:22 - cmdstanpy - INFO - Chain [1] done processing
12:39:28 - cmdstanpy - INFO - Chain [1] start processing
12:39:37 - cmdstanpy - INFO - Chain [1] done processing
12:39:43 - cmdstanpy - INFO - Chain [1] start processing
12:40:08 - cmdstanpy - INFO - Chain [1] done processing
12:40:14 - cmdstanpy - INFO - Chain [1] start processing
12:40:39 - cmdstanpy - INFO - Chain [1] done processing
12:40:45 - cmdstanpy - INFO - Chain [1] start processing
12:41:29 - cmdstanpy - INFO - Chain [1] done processing
12:41:35 - cmdstanpy - INFO - Chain [1] start processing
12:41:48 - cmdstanpy - INFO - Chain [1] done processing
12:41:54 - cmdstanpy - INFO - Chain [1] start processing
12:42:21 - cmdstanpy - INFO - Chain [1] done processing
12:42:23 - cmdstanpy - INFO - Chain [1] done processing
12:42:27 - cmdstanpy - INFO - Chain [1] start processing
12:42:29 - cmdstanpy - INFO - Chain [1] start processing
12:43:32 - cmdstanpy - INFO - Chain [1] done processing
12:43:37 - cmdstanpy - INFO - Chain [1] done processing
12:43:49 - cmdstanpy - INFO - Chain [1] done processing
12:43:57 - cmdstanpy - INFO - Chain [1] done processing
```

	changepoint_prior_scale	seasonality_mode	rmse
0	0.001	additive	6533.181535
1	0.001	multiplicative	7887.271391
2	0.100	additive	74818.854160
3	0.100	multiplicative	40687.806439

```
{'changepoint_prior_scale': 0.001, 'seasonality_mode': 'additive'}
CPU times: user 41.2 s, sys: 1.68 s, total: 42.9 s
Wall time: 43min 24s
```

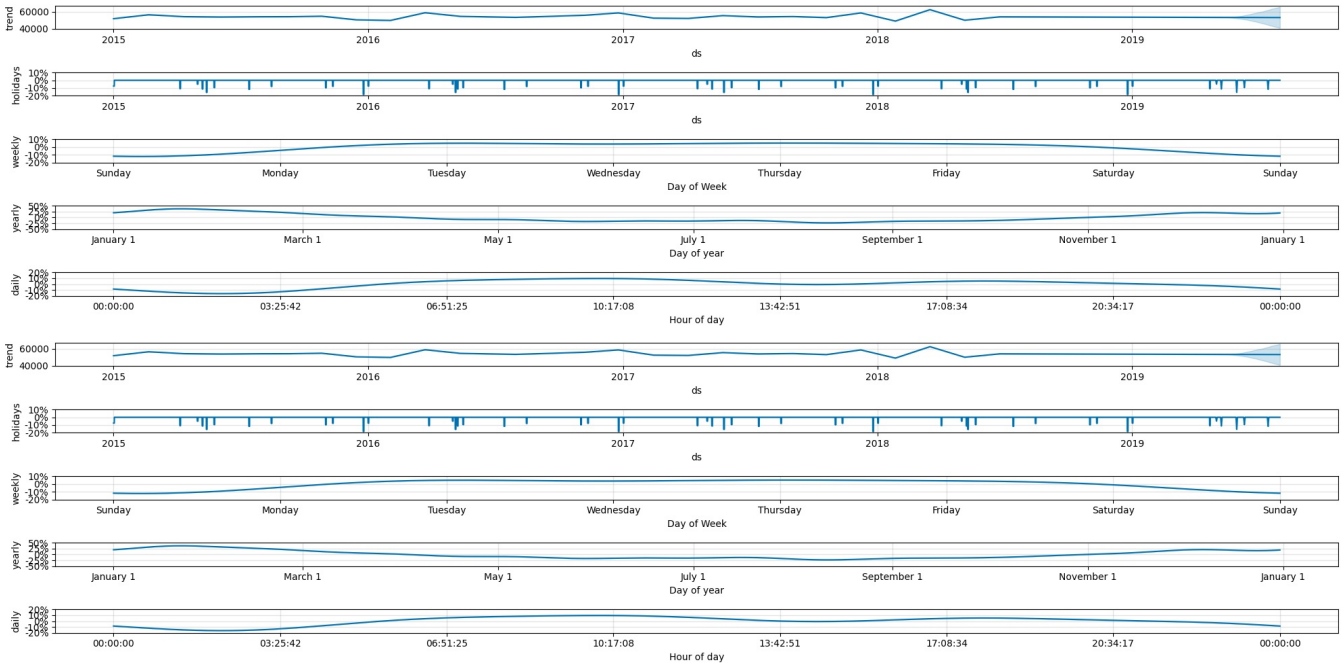
In [54]:
```
m = Prophet(**params)
m.add_country_holidays(country_name='FR')
m.fit(df_train)
```

```
12:44:01 - cmdstanpy - INFO - Chain [1] start processing
12:44:58 - cmdstanpy - INFO - Chain [1] done processing
```

Out[54]:
```
<prophet.forecaster.Prophet at 0x7a259ac47190>
```

In [55]:
```
future = m.make_future_dataframe(periods = 92, freq = 'D')
forecast = m.predict(future)
m.plot_components(forecast, figsize=(20, 5))
```

Out[55]:



In [56]:
```
forecast.columns
```

Out[56]:
```
Index(['ds', 'trend', 'yhat_lower', 'yhat_upper', 'trend_lower', 'trend_upper',
       'Armistice', 'Armistice_lower', 'Armistice_upper', 'Ascension',
       'Ascension_lower', 'Ascension_upper', 'Assomption', 'Assomption_lower',
       'Assomption_upper', 'Fête de la Victoire', 'Fête de la Victoire_lower',
       'Fête de la Victoire_upper', 'Fête du Travail', 'Fête du Travail_lower',
       'Fête du Travail_upper', 'Fête nationale', 'Fête nationale_lower',
       'Fête nationale_upper', 'Jour de l'an', 'Jour de l'an_lower',
       'Jour de l'an_upper', 'Lundi de Pentecôte', 'Lundi de Pentecôte_lower',
       'Lundi de Pentecôte_upper', 'Lundi de Pâques', 'Lundi de Pâques_lower',
       'Lundi de Pâques_upper', 'Noël', 'Noël_lower', 'Noël_upper',
       'Toussaint', 'Toussaint_lower', 'Toussaint_upper', 'daily',
       'daily_lower', 'daily_upper', 'holidays', 'holidays_lower',
       'holidays_upper', 'multiplicative_terms', 'multiplicative_terms_lower',
       'multiplicative_terms_upper', 'weekly', 'weekly_lower', 'weekly_upper',
       'yearly', 'yearly_lower', 'yearly_upper', 'additive_terms',
       'additive_terms_lower', 'additive_terms_upper', 'yhat'],
      dtype='object')
```

In [57]:
```
xfor = forecast[['ds', 'yhat', 'yhat_upper', 'yhat_lower']].loc[forecast['ds'] >= '2019-05-01' ]
xfor.head(10)
```

Out[57]:

	ds	yhat	yhat_upper	yhat_lower
37897	2019-05-01 23:00:00	46607.145521	51315.248362	41700.046799
37898	2019-05-02 23:00:00	48760.714675	54092.667814	43771.565297
37899	2019-05-03 23:00:00	46116.348595	51374.265663	40906.832284
37900	2019-05-04 23:00:00	40193.313151	45208.275693	35191.732635
37901	2019-05-05 23:00:00	43661.603030	49204.884490	38887.061839
37902	2019-05-06 23:00:00	48706.157891	53832.454083	43856.683535
37903	2019-05-07 23:00:00	48027.524794	52802.023655	42657.963788
37904	2019-05-08 23:00:00	42556.948214	47345.427900	37497.989720
37905	2019-05-09 23:00:00	47856.780242	52923.392824	42824.106798
37906	2019-05-10 23:00:00	45057.417908	50084.676728	39863.690378

In [58]:
```
# we use a left outer join because of the missing observations in the original data
xfor = pd.merge(left = xfor, right = dfx, on = 'ds', how = 'left')
```

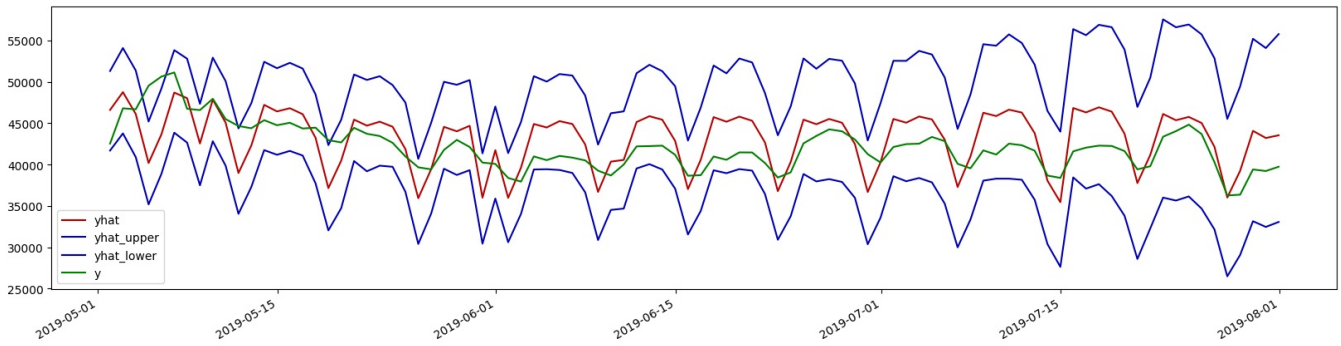
In [59]:
```
xfor.head(10)
```

Out[59]:

	ds	yhat	yhat_upper	yhat_lower	y
0	2019-05-01 23:00:00	46607.145521	51315.248362	41700.046799	42542.0
1	2019-05-02 23:00:00	48760.714675	54092.667814	43771.565297	46807.0
2	2019-05-03 23:00:00	46116.348595	51374.265663	40906.832284	46683.0
3	2019-05-04 23:00:00	40193.313151	45208.275693	35191.732635	49541.0
4	2019-05-05 23:00:00	43661.603030	49204.884490	38887.061839	50642.0
5	2019-05-06 23:00:00	48706.157891	53832.454083	43856.683535	51140.0
6	2019-05-07 23:00:00	48027.524794	52802.023655	42657.963788	46752.0
7	2019-05-08 23:00:00	42556.948214	47345.427900	37497.989720	46588.0
8	2019-05-09 23:00:00	47856.780242	52923.392824	42824.106798	47957.0
9	2019-05-10 23:00:00	45057.417908	50084.676728	39863.690378	45498.0

In [60]:
```
colors = ['#BB0000' , '#0000BB', '#0000BB', 'green']

xfor['ds'] = pd.to_datetime(xfor['ds'])
xfor.set_index('ds').plot(color = colors, figsize=(20,5), xlabel = '')
```

Out[60]:
```
<Axes: >
```



https://www.kaggle.com/code/pythonafroz/a-guide-to-time-series-analysis-part-04/notebook

In [ ]:
```
```