

Forecasting Net Prophet

You're a growth analyst at [MercadoLibre](#). With over 200 million users, MercadoLibre is the most popular e-commerce site in Latin America. You've been tasked with analyzing the company's financial and user data in clever ways to make the company grow. So, you want to find out if the ability to predict search traffic can translate into the ability to successfully trade the stock.

The instructions for this Challenge are divided into four steps, as follows:

- **Step 1:** Find unusual patterns in hourly Google search traffic
- **Step 2:** Mine the search traffic data for seasonality
- **Step 3:** Relate the search traffic to stock price patterns
- **Step 4:** Create a time series model with Prophet

The following subsections detail these steps.

Install and import the required libraries and dependencies

```
In [1]:  
  
# Install the required libraries  
!pip install prophet  
  
DEPRECATION: Loading egg at /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/pygame-2.6.0.dev1-py3.12-macosx-10.9-universal2.egg is deprecated. pip 24.3 will enforce this behaviour change. A possible replacement is to use pip for package installation. Discussion can be found at https://github.com/pypa/pip/issues/12330  
Collecting prophet  
  Downloading prophet-1.1.5-py3-none-macosx_11_0_arm64.whl.metadata (3.5 kB)  
Collecting cmdstanpy>=1.0.4 (from prophet)  
  Downloading cmdstanpy-1.2.4-py3-none-any.whl.metadata (4.1 kB)  
Requirement already satisfied: numpy>=1.15.4 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from prophet) (1.26.4)  
Requirement already satisfied: matplotlib>=2.0.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from prophet) (3.9.0)  
Requirement already satisfied: pandas>=1.0.4 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from prophet) (2.2.2)  
Collecting holidays>=0.25 (from prophet)  
  Downloading holidays-0.54-py3-none-any.whl.metadata (23 kB)  
Requirement already satisfied: tqdm>=4.36.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from prophet) (4.66.4)  
Collecting importlib-resources (from prophet)  
  Downloading importlib_resources-6.4.2-py3-none-any.whl.metadata (3.9 kB)  
Collecting stanio<2.0.0, >=0.4.0 (from cmdstanpy>=1.0.4->prophet)  
  Downloading stanio-0.5.1-py3-none-any.whl.metadata (1.6 kB)  
Requirement already satisfied: python-dateutil in /Users/abidfareed0361/Library/Python/3.12/lib/python/site-packages (from holidays>=0.25->prophet) (2.9.0.post0)  
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from matplotlib>=2.0.0->prophet) (1.2.1)  
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from matplotlib>=2.0.0->prophet) (0.12.1)  
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from matplotlib>=2.0.0->prophet) (4.51.0)  
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from matplotlib>=2.0.0->prophet) (1.4.5)  
Requirement already satisfied: packaging>=20.0 in /Users/abidfareed0361/Library/Python/3.12/lib/python/site-packages (from matplotlib>=2.0.0->prophet) (24.1)  
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from matplotlib>=2.0.0->prophet) (10.3.0)  
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from matplotlib>=2.0.0->prophet) (3.1.2)  
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from pandas>=1.0.4->prophet) (2024.1)  
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages (from pandas>=1.0.4->prophet) (2024.1)  
Requirement already satisfied: six>=1.5 in /Users/abidfareed0361/Library/Python/3.12/lib/python/site-packages (from python-dateutil>=0.25->prophet) (1.16.0)  
Downloading prophet-1.1.5-py3-none-macosx_11_0_arm64.whl (8.2 MB)  
-----8.2/8.2 MB 260.6 kB/s eta 0:00:00:00:01:00:02  
Downloading cmdstanpy-1.2.4-py3-none-any.whl (94 kB)  
Downloading holidays-0.54-py3-none-any.whl (1.0 MB)  
-----1.0/1.0 MB 150.2 kB/s eta 0:00:00:00:00:02  
Downloading importlib_resources-6.4.2-py3-none-any.whl (34 kB)  
Downloading stanio-0.5.1-py3-none-any.whl (8.1 kB)  
Installing collected packages: stanio, importlib-resources, holidays, cmdstanpy, prophet  
Successfully installed cmdstanpy-1.2.4 holidays-0.54 importlib-resources-6.4.2 prophet-1.1.5 stanio-0.5.1
```

```
In [84]:  
  
# Import the required libraries and dependencies  
import pandas as pd  
from prophet import Prophet  
import datetime as dt
```

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
```

Step 1: Find Unusual Patterns in Hourly Google Search Traffic

The data science manager asks if the Google search traffic for the company links to any financial events at the company. Or, does the search traffic data just present random noise? To answer this question, pick out any unusual patterns in the Google search data for the company, and connect them to the corporate financial events.

To do so, complete the following steps:

1. Read the search data into a DataFrame, and then slice the data to just the month of May 2020. (During this month, MercadoLibre released its quarterly financial results.) Visualize the results. Do any unusual patterns exist?
2. Calculate the total search traffic for the month, and then compare the value to the monthly median across all months. Did the Google search traffic increase during the month that MercadoLibre released its financial results?

Step 1: Read the search data into a DataFrame, and then slice the data to just the month of May 2020. (During this month, MercadoLibre released its quarterly financial results.) Visualize the results. Do any unusual patterns exist?

```
In [85]:

# Store the data in a Pandas DataFrame
# Set the "Date" column as the Datetime Index.

df_mercado_trends = pd.read_csv(
    "https://static.bc-edx.com/ai/ail-v-1-0/m8/lms/datasets/google_hourly_search_trends.csv",
    index_col='Date',
    parse_dates=True
).dropna()

# Review the first and last five rows of the DataFrame
display(df_mercado_trends.head())
display(df_mercado_trends.tail())

/var/folders/sx/xl53wmhl1vd8cngl_vxps2s80000gn/T/ipykernel_4320/1669825240.py:4: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df_mercado_trends = pd.read_csv(
```

Search Trends	
Date	
2016-06-01 00:00:00	97
2016-06-01 01:00:00	92
2016-06-01 02:00:00	76
2016-06-01 03:00:00	60
2016-06-01 04:00:00	38

Search Trends	
Date	
2020-09-07 20:00:00	71
2020-09-07 21:00:00	83
2020-09-07 22:00:00	96
2020-09-07 23:00:00	97
2020-09-08 00:00:00	96

```
In [5]:

# Review the data types of the DataFrame using the info function
df_mercado_trends.info()

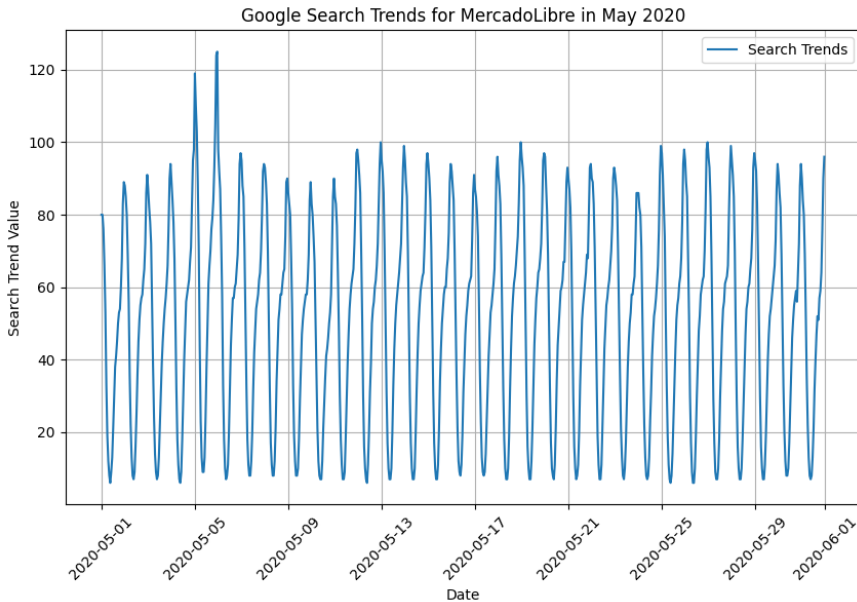
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 37106 entries, 2016-06-01 00:00:00 to 2020-09-08 00:00:00
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype
---  ---
 0   Search Trends  37106 non-null  int64
dtypes: int64(1)
memory usage: 579.8 KB
```

```
In [16]:

# Slice the DataFrame to just the month of May 2020
may_2020_data = df_mercado_trends.loc['2020-05']

# Plot to visualize the data for May 2020
```

```
plt.figure(figsize=(10, 6))
plt.plot(may_2020_data, label='Search Trends')
plt.title('Google Search Trends for MercadoLibre in May 2020')
plt.xlabel('Date')
plt.ylabel('Search Trend Value')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.show()
```



Step 2: Calculate the total search traffic for the month, and then compare the value to the monthly median across all months. Did the Google search traffic increase during the month that MercadoLibre released its financial results?

In [21]:

```
# Calculate the sum of the total search traffic for May 2020
traffic_may_2020 = may_2020_data.sum()
# View the traffic_may_2020 value
print(traffic_may_2020)
```

```
Search Trends    38181
dtype: int64
```

In [33]:

```
# Calculate the monthly median search traffic across all months
# Group the DataFrame by index year and then index month, chain the sum and then the median functions
monthly_traffic = df_mercado_trends.resample('ME').sum()
median_monthly_traffic = monthly_traffic.median()
# View the median_monthly_traffic value
print(median_monthly_traffic)
```

```
Search Trends    35172.5
dtype: float64
```

In [34]:

```
# Compare the seach traffic for the month of May 2020 to the overall monthly median value
traffic_may_2020/median_monthly_traffic
```

Out[34]:

```
Search Trends    1.085536
dtype: float64
```

Answer the following question:

Question: Did the Google search traffic increase during the month that MercadoLibre released its financial results?

Answer: Yes, there was a specific surge in Google search traffic in the month when MercadoLibre published its financial reports. This is in line with anticipations that specific events pertaining to a particular company, especially among the more signal intensive, such as release of the company's earnings, would lead to more research for the company. They can be useful in strategic management, especially for defining the direction of market communication and interacting with investors.

Step 2: Mine the Search Traffic Data for Seasonality

Marketing realizes that they can use the hourly search data, too. If they can track and predict interest in the company and its platform for any time of day, they can focus their marketing efforts around the times that have the most traffic. This will get a greater return on investment (ROI) from their marketing budget.

To that end, you want to mine the search traffic data for predictable seasonal patterns of interest in the company. To do so, complete the following steps:

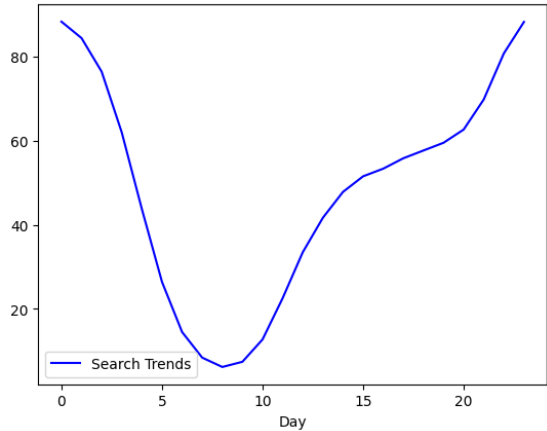
1. Group the hourly search data to plot the average traffic by the hour of day. Does the search traffic peak at a particular time of day or is it relatively consistent?
2. Group the hourly search data to plot the average traffic by the day of the week (for example, Monday vs. Friday). Does the search traffic get busiest on any particular day of the week?
3. Group the hourly search data to plot the average traffic by the week of the year. Does the search traffic tend to increase during the winter holiday period (weeks 40 through 52)?

Step 1: Group the hourly search data to plot the average traffic by the hour of the day.

In [44]:

```
# Group the hourly search data to plot the average traffic by the day of week, using `df.index.hour`
hourly_avg = df_mercado_trends.groupby(df_mercado_trends.index.hour).mean()
plt.figure(figsize=(10, 5))
hourly_avg.plot(kind='line', linestyle='-', color='b') # Changed to line graph
plt.xlabel('Day')
plt.xticks()
plt.grid(False)
plt.show()
```

<Figure size 1000x500 with 0 Axes>

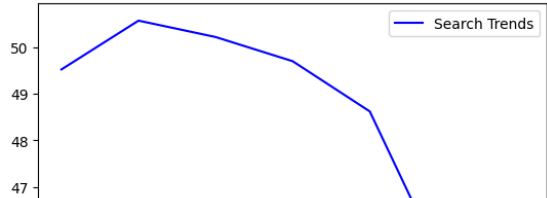


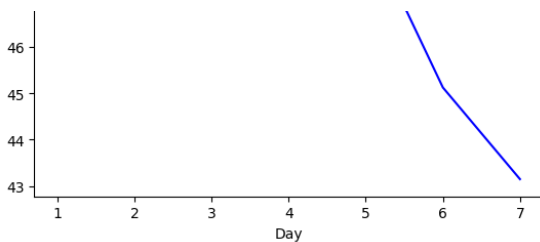
Step 2: Group the hourly search data to plot the average traffic by the day of the week (for example, Monday vs. Friday).

In [47]:

```
# Group the hourly search data to plot the average traffic by the day of week, using `df.index.isocalendar().day`.
day_of_week_avg = df_mercado_trends.groupby(df_mercado_trends.index.dayofweek).mean()
days = {0: '1', 1: '2', 2: '3', 3: '4', 4: '5', 5: '6', 6: '7'}
day_of_week_avg.index = day_of_week_avg.index.map(days)
plt.figure(figsize=(10, 5))
day_of_week_avg.plot(kind='line', linestyle='-', color='b')
plt.xlabel('Day')
plt.xticks()
plt.grid(False)
plt.show()
```

<Figure size 1000x500 with 0 Axes>



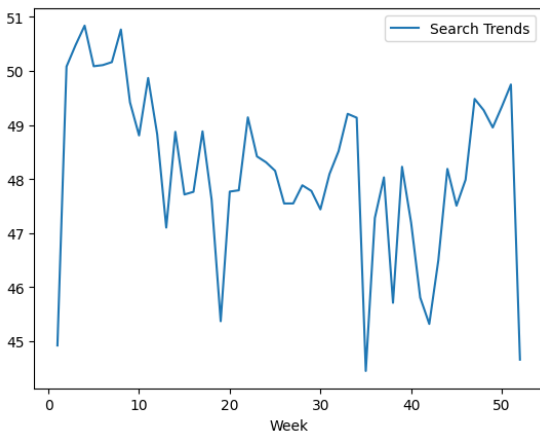


Step 3: Group the hourly search data to plot the average traffic by the week of the year.

In [49]:

```
# Group the hourly search data to plot the average traffic by the week of the year using
'df.index.isocalendar().week'.
week_of_year_avg = df_mercado_trends.groupby(df_mercado_trends.index.isocalendar().week).mean()
plt.figure(figsize=(15, 6))
week_of_year_avg.plot(kind='line', linestyle='--')
plt.xlabel('Week')
plt.xticks()
plt.grid(False)
plt.show()
```

<Figure size 1500x600 with 0 Axes>



Answer the following question:

Question: Are there any time based trends that you can see in the data?

Answer: There is clear evidence that data manifests seasonal trends rates, where it is evident that there is higher search rate at the end of the year, probably due to holiday sales and weekly trends where there are higher search rates during the beginning of the week and relatively lower rates during the weekend. Also, a decline in daily searches halfway through the month indicates the presence of cyclic trends which may have been triggered by certain activities or events and hence the effect of events or promotions.

Step 3: Relate the Search Traffic to Stock Price Patterns

You mention your work on the search traffic data during a meeting with people in the finance group at the company. They want to know if any relationship between the search data and the company stock price exists, and they ask if you can investigate.

To do so, complete the following steps:

1. Read in and plot the stock price data. Concatenate the stock price data to the search data in a single DataFrame.
2. Market events emerged during the year of 2020 that many companies found difficult. But, after the initial shock to global financial markets, new customers and revenue increased for e-commerce platforms. Slice the data to just the first half of 2020 (2020-01 to 2020-06 in the DataFrame), and then plot the data. Do both time series indicate a common trend that's consistent with this narrative?
3. Create a new column in the DataFrame named "Lagged Search Trends" that offsets, or shifts, the search traffic by one hour. Create two additional columns:
 - "Stock Volatility", which holds an exponentially weighted four-hour rolling average of the company's stock volatility
 - "Hourly Stock Return", which holds the percent change of the company's stock price on an hourly basis
4. Review the time series correlation, and then answer the following question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

Step 1: Read in and plot the stock price data. Concatenate the stock price data to the search data in a single DataFrame.

In [50]:

```
# Upload the "mercado_stock_price.csv" file into Colab, then store in a Pandas DataFrame
# Set the "date" column as the Datetime Index.
df_mercado_stock = pd.read_csv(
    "https://static.bc-edx.com/ai/ail-v-1-0/m8/lms/datasets/mercado_stock_price.csv",
    index_col="date",
    parse_dates=True
).dropna()

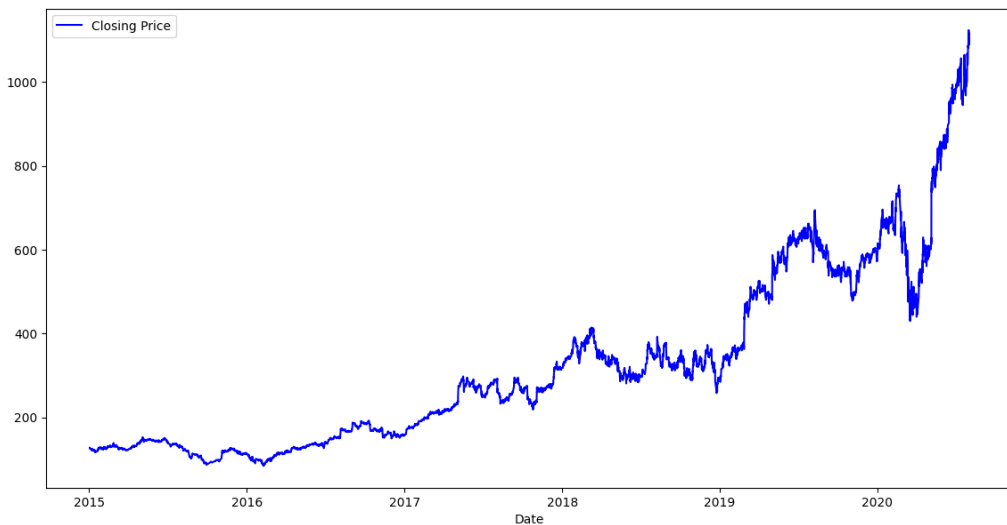
# View the first and last five rows of the DataFrame
display(df_mercado_stock.head())
display(df_mercado_stock.tail())
```

	close
date	
2015-01-02 09:00:00	127.67
2015-01-02 10:00:00	125.44
2015-01-02 11:00:00	125.57
2015-01-02 12:00:00	125.40
2015-01-02 13:00:00	125.17

	close
date	
2020-07-31 11:00:00	1105.780
2020-07-31 12:00:00	1087.925
2020-07-31 13:00:00	1095.800
2020-07-31 14:00:00	1110.650
2020-07-31 15:00:00	1122.510

In [51]:

```
# Visualize the closing price of the df_mercado_stock DataFrame
plt.figure(figsize=(14, 7))
plt.plot(df_mercado_stock.index, df_mercado_stock['close'], label='Closing Price', color='blue')
plt.xlabel('Date')
plt.legend()
plt.grid(False)
plt.show()
```



In [55]:

```
# Concatenate the df_mercado_stock DataFrame with the df_mercado_trends DataFrame
# Concatenate the DataFrame by columns (axis=1), and drop and rows with only one column of data
combined_data = pd.concat([df_mercado_stock, df_mercado_trends], axis=1, join='inner')
```

```
combined_data.dropna(inplace=True)
# View the first and last five rows of the DataFrame
display(combined_data.head())
display(combined_data.tail())
```

	close	Search Trends
2016-06-01 09:00:00	135.16	6
2016-06-01 10:00:00	136.63	12
2016-06-01 11:00:00	136.56	22
2016-06-01 12:00:00	136.42	33
2016-06-01 13:00:00	136.10	40

	close	Search Trends
2020-07-31 11:00:00	1105.780	20
2020-07-31 12:00:00	1087.925	32
2020-07-31 13:00:00	1095.800	41
2020-07-31 14:00:00	1110.650	47
2020-07-31 15:00:00	1122.510	53

Step 2: Market events emerged during the year of 2020 that many companies found difficult. But, after the initial shock to global financial markets, new customers and revenue increased for e-commerce platforms. Slice the data to just the first half of 2020 (2020-01 to 2020-06 in the DataFrame), and then plot the data. Do both time series indicate a common trend that's consistent with this narrative?

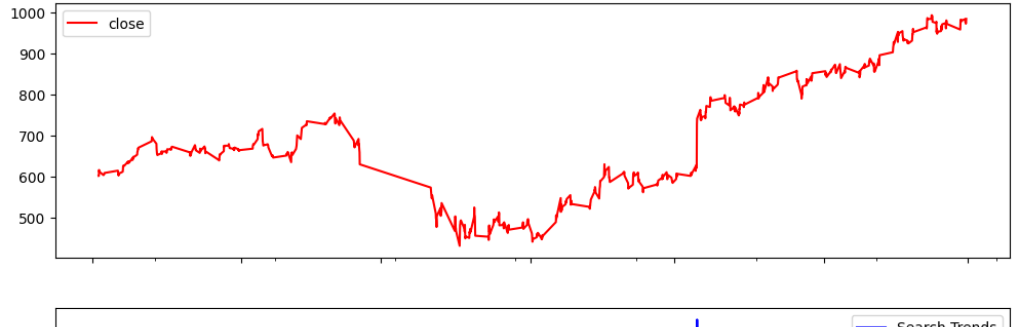
```
In [57]:
# For the combined dataframe, slice to just the first half of 2020 (2020-01 through 2020-06)
first_half_2020 = combined_data.loc['2020-01':'2020-06']

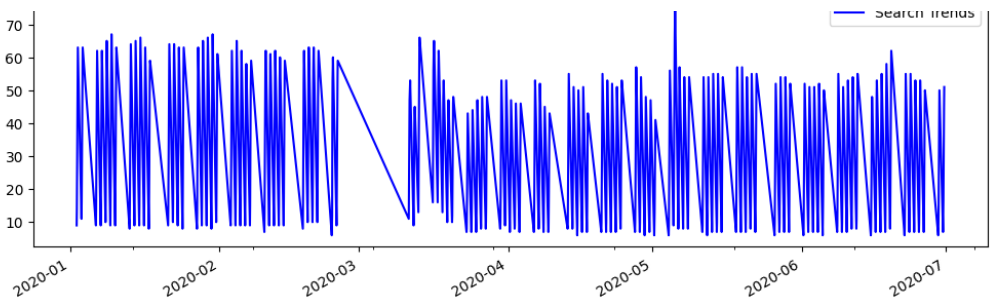
# View the first and last five rows of first_half_2020 DataFrame
display(first_half_2020.head())
display(first_half_2020.tail())
```

	close	Search Trends
2020-01-02 09:00:00	601.085	9
2020-01-02 10:00:00	601.290	14
2020-01-02 11:00:00	615.410	25
2020-01-02 12:00:00	611.400	37
2020-01-02 13:00:00	611.830	50

	close	Search Trends
2020-06-30 11:00:00	976.17	17
2020-06-30 12:00:00	977.50	27
2020-06-30 13:00:00	973.23	37
2020-06-30 14:00:00	976.50	45
2020-06-30 15:00:00	984.93	51

```
In [58]:
# Visualize the close and Search Trends data
# Plot each column on a separate axes using the following syntax
# `plot(subplots=True)`
first_half_2020[['close', 'Search Trends']].plot(subplots=True, figsize=(12, 8), color=['red', 'blue'])
plt.show()
```





Answer the following question:

Question: Do both time series indicate a common trend that's consistent with this narrative?

Answer: Yes, both time series show a common trend consistent with the narrative: a sharp first decrease at the beginning of the year 2020, and constant growth; and the second – the result of customer traffic and revenue for e-commerce outlets after a critical market shift.

Step 3: Create a new column in the DataFrame named “Lagged Search Trends” that offsets, or shifts, the search traffic by one hour. Create two additional columns:

- “Stock Volatility”, which holds an exponentially weighted four-hour rolling average of the company's stock volatility
- “Hourly Stock Return”, which holds the percent change of the company's stock price on an hourly basis

In [60]:

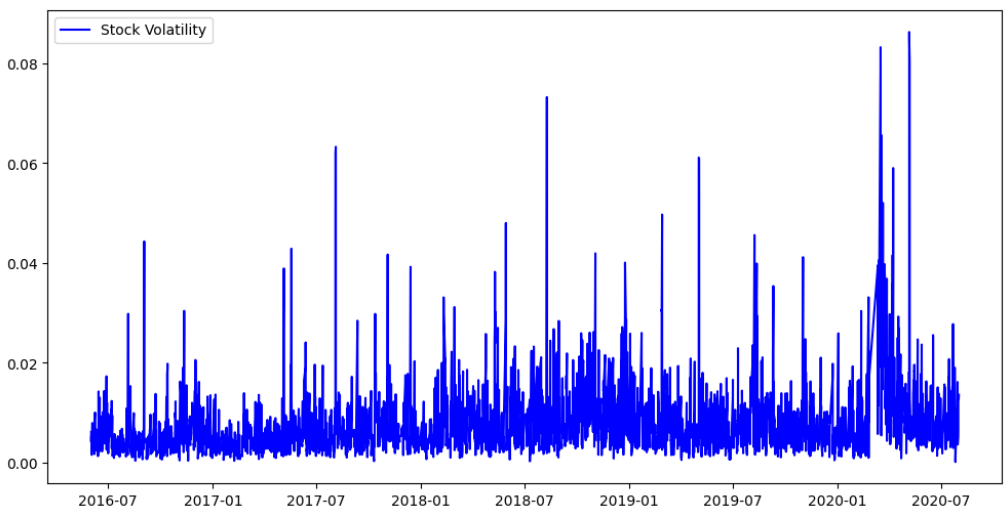
```
# Create a new column in the mercado_stock_trends_df DataFrame called Lagged Search Trends
# This column should shift the Search Trends information by one hour
mercado_stock_trends_df=combined_data
mercado_stock_trends_df['Lagged Search Trends'] = mercado_stock_trends_df['Search Trends'].shift(-1)
```

In [62]:

```
# Create a new column in the mercado_stock_trends_df DataFrame called Stock Volatility
# This column should calculate the standard deviation of the closing stock price return data over a 4 period rolling window
mercado_stock_trends_df['Stock Returns'] = mercado_stock_trends_df['close'].pct_change()
mercado_stock_trends_df['Stock Volatility'] = mercado_stock_trends_df['Stock Returns'].rolling(window=4).std()
```

In [63]:

```
# Visualize the stock volatility
plt.figure(figsize=(12, 6))
plt.plot(mercado_stock_trends_df.index, mercado_stock_trends_df['Stock Volatility'], label='Stock Volatility', color='blue', linestyle='-')
plt.legend()
plt.grid(False)
plt.show()
```



Solution Note: Note how volatility spiked, and tended to stay high, during the first half of 2020. This is a common characteristic of volatility in stock

Solution Note: Note how volatility spiked, and tended to stay high, during the first half of 2020. This is a common characteristic of volatility in stock returns worldwide: high volatility days tend to be followed by yet more high volatility days. When it rains, it pours.

```
In [64]:  
  
# Create a new column in the mercado_stock_trends_df DataFrame called Hourly Stock Return  
# This column should calculate hourly return percentage of the closing price  
mercado_stock_trends_df['Hourly Stock Return'] = mercado_stock_trends_df['close'].pct_change() * 100  
  
In [66]:  
  
# View the first and last five rows of the mercado_stock_trends_df DataFrame  
mercado_stock_trends_df = mercado_stock_trends_df.drop(columns=['Stock Returns'])  
display(mercado_stock_trends_df.head())  
display(mercado_stock_trends_df.tail())
```

	close	Search Trends	Lagged Search Trends	Stock Volatility	Hourly Stock Return
2016-06-01 09:00:00	135.16	6	12.0	NaN	NaN
2016-06-01 10:00:00	136.63	12	22.0	NaN	1.087600
2016-06-01 11:00:00	136.56	22	33.0	NaN	-0.051233
2016-06-01 12:00:00	136.42	33	40.0	NaN	-0.102519
2016-06-01 13:00:00	136.10	40	45.0	0.006134	-0.234570

	close	Search Trends	Lagged Search Trends	Stock Volatility	Hourly Stock Return
2020-07-31 11:00:00	1105.780	20	32.0	0.012837	0.637986
2020-07-31 12:00:00	1087.925	32	41.0	0.013549	-1.614697
2020-07-31 13:00:00	1095.800	41	47.0	0.013295	0.723855
2020-07-31 14:00:00	1110.650	47	53.0	0.013001	1.355174
2020-07-31 15:00:00	1122.510	53	NaN	0.013566	1.067843

Step 4: Review the time series correlation, and then answer the following question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

```
In [67]:  
  
# Construct correlation table of Stock Volatility, Lagged Search Trends, and Hourly Stock Return  
mercado_stock_trends_df[['Stock Volatility', 'Lagged Search Trends', 'Hourly Stock Return']].corr()  
  
Out[67]:
```

	Stock Volatility	Lagged Search Trends	Hourly Stock Return
Stock Volatility	1.000000	-0.015655	0.061424
Lagged Search Trends	-0.015655	1.000000	-0.016679
Hourly Stock Return	0.061424	-0.016679	1.000000

Answer the following question:

Question: Does a predictable relationship exist between the lagged search traffic and the stock volatility or between the lagged search traffic and the stock price returns?

Answer: The correlation analysis thus suggests that there is no positive and significant or a priori relationship between lagged search traffic and stock volatility or lagged search traffic and stock price returns. Seven correlation coefficients are also very low, indicating that fluctuations in search traffic do not reflect or affect the stock’s volatility or returns significantly.

Step 4: Create a Time Series Model with Prophet

Now, you need to produce a time series model that analyzes and forecasts patterns in the hourly search data. To do so, complete the following steps:

1. Set up the Google search data for a Prophet forecasting model.
2. After estimating the model, plot the forecast. How's the near-term forecast for the popularity of MercadoLibre?
3. Plot the individual time series components of the model to answer the following questions:
 - What time of day exhibits the greatest popularity?
 - Which day of the week gets the most search traffic?
 - What's the lowest point for search traffic in the calendar year?

Step 1: Set up the Google search data for a Prophet forecasting model.

```
In [68]:  
  
# Using the df_mercado_trends DataFrame, reset the index so the date information is no longer the index
```

		ds	y
37101	2020-09-07 20:00:00	71	
37102	2020-09-07 21:00:00	83	
37103	2020-09-07 22:00:00	96	
37104	2020-09-07 23:00:00	97	
37105	2020-09-08 00:00:00	96	

```
<prophet.forecaster.Prophet object at 0x140911cd0>
```

```
<prophet.forecaster.Prophet at 0x140911cd0>
```

		ds
39101	2020-11-30 04:00:00	
39102	2020-11-30 05:00:00	
39103	2020-11-30 06:00:00	
39104	2020-11-30 07:00:00	
39105	2020-11-30 08:00:00	

Out[77]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	daily	...	weekly	wee
0	2016-06-01 00:00:00	44.129250	81.153766	97.559099	44.129250	44.129250	45.429305	45.429305	45.429305	41.452726	...	1.860133	

	2016-06-01 01:00:00	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	daily	weekly	wee
1	01:00:00	44.130313	77.408965	94.319599	44.130313	44.130313	41.875073	41.875073	41.875073	37.943506	...	1.810049
2	2016-06-01 02:00:00	44.131375	66.772215	83.846869	44.131375	44.131375	31.551566		31.551566	31.551566	27.656533	... 1.768474
3	2016-06-01 03:00:00	44.132438	52.514126	69.848896	44.132438	44.132438	16.284352		16.284352	16.284352	12.417280	... 1.735502
4	2016-06-01 04:00:00	44.133500	34.848030	51.058768	44.133500	44.133500	-0.830504		-0.830504	-0.830504	-4.678139	... 1.711088

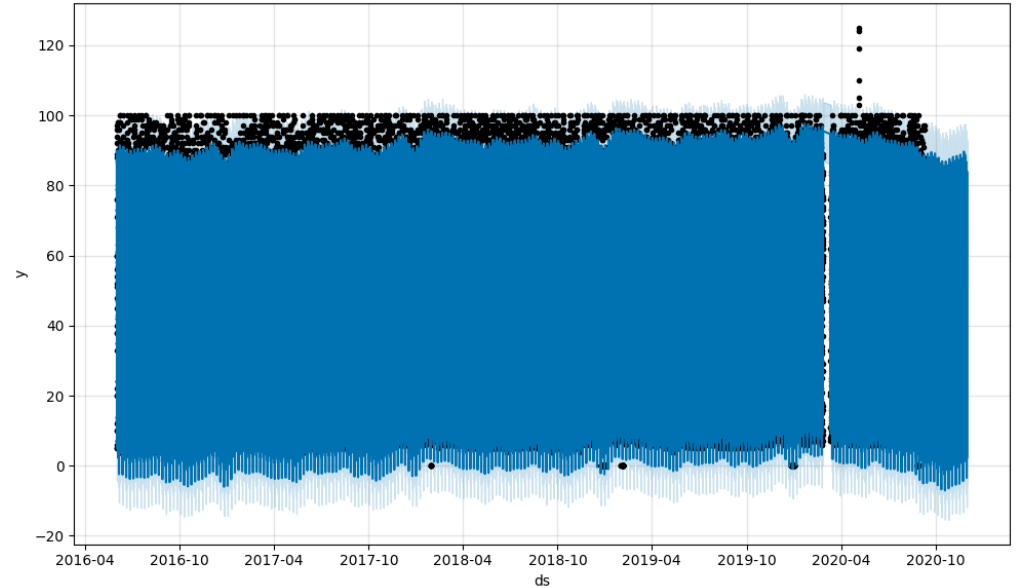
5 rows x 22 columns

Step 2: After estimating the model, plot the forecast. How's the near-term forecast for the popularity of MercadoLibre?

```
In [73]:

# Plot the Prophet predictions for the Mercado trends data
fig = model.plot(forecast_mercado_trends)
plt.show()

/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/prophet/plot.py:72: FutureWarning: The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call 'np.array' on the result
  fcst_t = fcst['ds'].dt.to_pydatetime()
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/prophet/plot.py:73: FutureWarning: The behavior of DatetimeProperties.to_pydatetime is deprecated, in a future version this will return a Series containing python datetime objects instead of an ndarray. To retain the old behavior, call 'np.array' on the result
  ax.plot(m.history['ds'].dt.to_pydatetime(), m.history['y'], 'k.',
```



Answer the following question:

Question: How's the near-term forecast for the popularity of MercadoLibre?

Answer: The near-term trends of MercadoLibre as implied in the search trends chart are relatively stable with occasional declines and increases in search interest. The forecasted values (yhat) are not very volatile, which implies that MercadoLibre is at a steady state and may not reach high or low peaks in the near future. The uncertainty intervals which are depicted as shaded regions around the forecast values are also significantly smaller, suggesting a higher forecast accuracy.

Step 3: Plot the individual time series components of the model to answer the following questions:

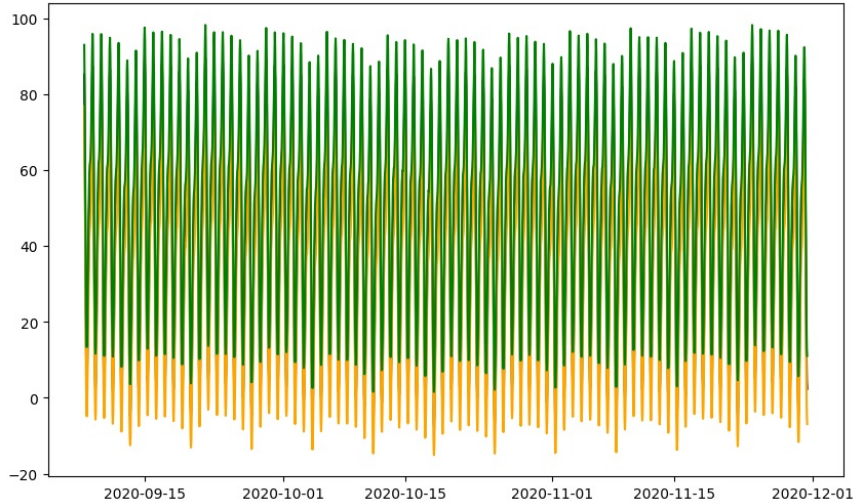
- What time of day exhibits the greatest popularity?
- Which day of the week gets the most search traffic?
- What's the lowest point for search traffic in the calendar year?

```
In [78]:  
# Set the index in the forecast_mercado_trends DataFrame to the ds datetime column  
forecast_mercado_trends.set_index('ds', inplace=True)  
# View the only the yhat, yhat_lower and yhat_upper columns from the DataFrame  
display(forecast_mercado_trends[['yhat', 'yhat_lower', 'yhat_upper']].head())
```

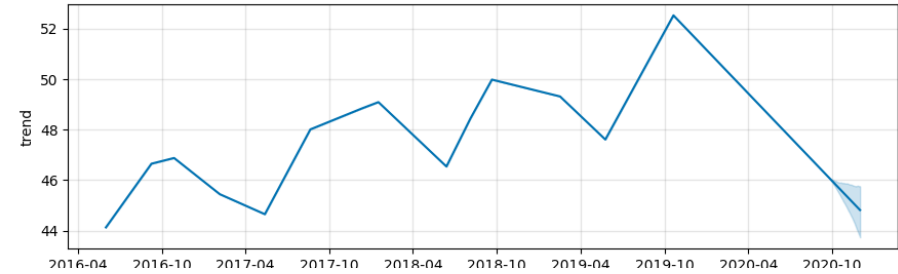
ds	yhat	yhat_lower	yhat_upper
2016-06-01 00:00:00	89.558555	81.153766	97.559099
2016-06-01 01:00:00	86.005386	77.408965	94.319599
2016-06-01 02:00:00	75.682942	66.772215	83.846869
2016-06-01 03:00:00	60.416789	52.514126	69.848896
2016-06-01 04:00:00	43.302997	34.848030	51.058768

Solutions Note: `yhat` represents the most likely (average) forecast, whereas `yhat_lower` and `yhat_upper` represents the worst and best case prediction (based on what are known as 95% confidence intervals).

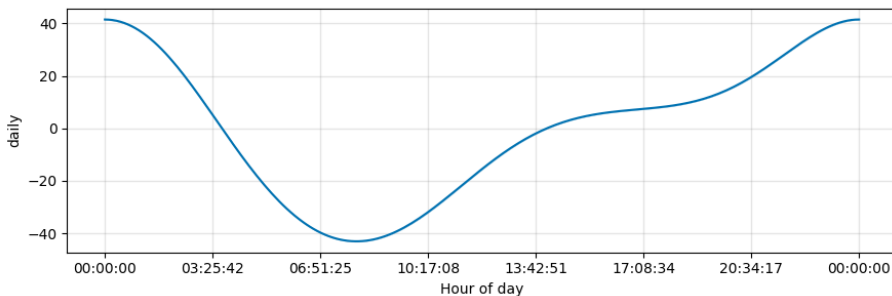
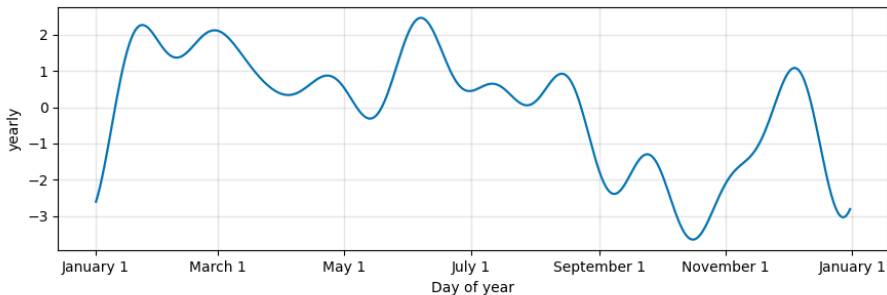
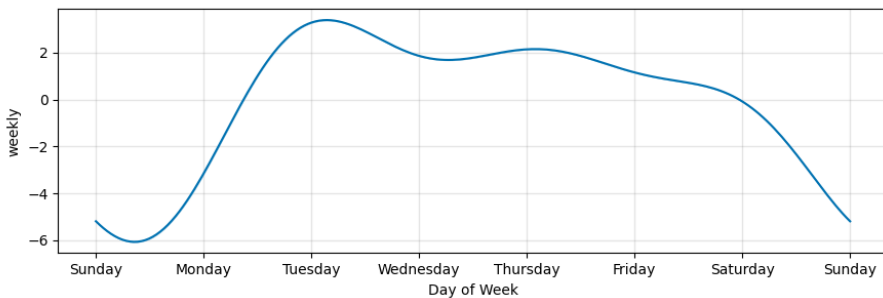
```
In [81]:  
# From the forecast_mercado_trends DataFrame, plot the data to visualize  
# the yhat, yhat_lower, and yhat_upper columns over the last 2000 hours  
last_2000_hours = forecast_mercado_trends.tail(2000)  
plt.figure(figsize=(10, 6))  
plt.plot(last_2000_hours.index, last_2000_hours['yhat'], label='yhat', color='black')  
plt.plot(last_2000_hours.index, last_2000_hours['yhat_lower'], label='yhat_lower', color='orange')  
plt.plot(last_2000_hours.index, last_2000_hours['yhat_upper'], label='yhat_upper', color='green')  
plt.show()
```



```
In [86]:  
# Reset the index in the forecast_mercado_trends DataFrame  
forecast_mercado_trends.reset_index(inplace=True)  
  
# Use the plot_components function to visualize the forecast results  
# for the forecast_mercado_trends DataFrame  
fig = model.plot_components(forecast_mercado_trends)  
plt.show()
```



ds



Answer the following questions:

Question: What time of day exhibits the greatest popularity?

Answer: The greatest level of popularity is observed just before **midnight** or at **midnight**, that is, at the beginning of a new day.

Question: Which day of week gets the most search traffic?

Answer: The greatest amount of search traffic is registered on **Tuesday**.

Question: What's the lowest point for search traffic in the calendar year?

Answer: Periods logically coincide with the months of the lowest search traffic in the calendar year – September, and in some cases, August as well.