Monterey Weather: Exploratory Data Visualization

In [459]:
```
from datetime import date

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [301]:
```
df = pd.read_csv('cleaned_df.csv', parse_dates=['datetime'], index_col=['da
df.head()
```

Out[301]:

	DATE	HourlyPresentWeatherType	HourlySkyConditions	HourlyVisibility	Hourlyl
datetime					
2009-04-01 00:08:00	2009-04-01T00:08:00	NaN	[{'BKN': SkyCondition(obscuration=7, vertical_...	10.0	
2009-04-01 00:50:00	2009-04-01T00:50:00	NaN	[{'SCT': SkyCondition(obscuration=4, vertical_...	9.0	
2009-04-01 00:54:00	2009-04-01T00:54:00	NaN	[{'SCT': SkyCondition(obscuration=4, vertical_...	9.0	
2009-04-01 01:54:00	2009-04-01T01:54:00	NaN	[]	9.0	
2009-04-01 02:54:00	2009-04-01T02:54:00	NaN	[]	9.0	

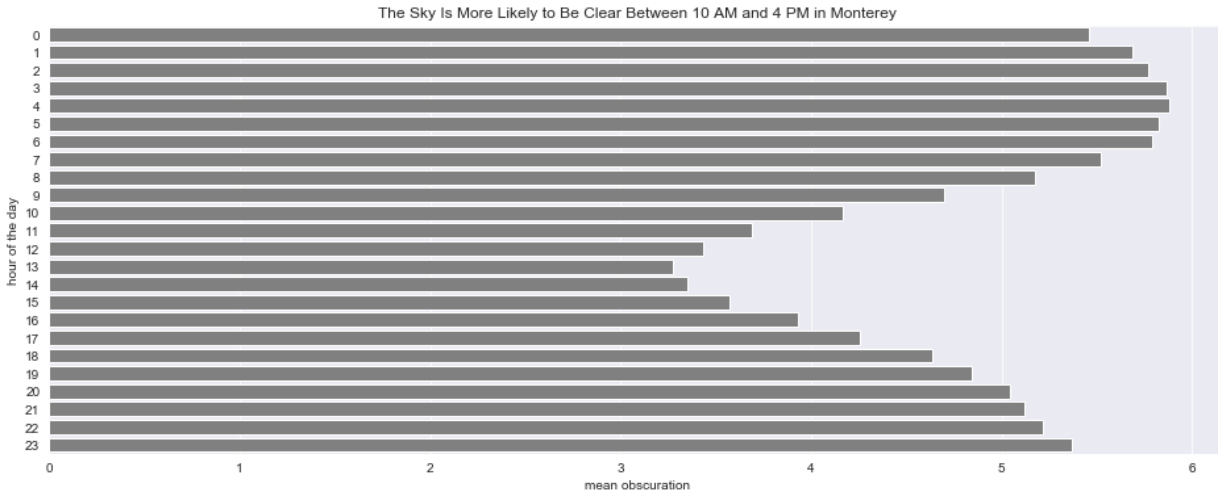
Introduction

Chaos determines weather, and [the most powerful weather prediction systems (https://xkcd.com/1885/)](https://xkcd.com/1885/) depend on models of chaos. With this in mind, when asked to make a prediction for hypothetical event planners based on a decade of hourly weather data using a laptop, it's useful to frame the task of exploratory visualization with the question: from what perspectives is this data least chaotic?

Event planners are often most concerned with the presence or absence of sun, which appears in the dataset as the average sky obscuration value. So a good first question might be,

1. Averaged across the whole decade, which hours of the day tend to be more or less sunny (obscured)?

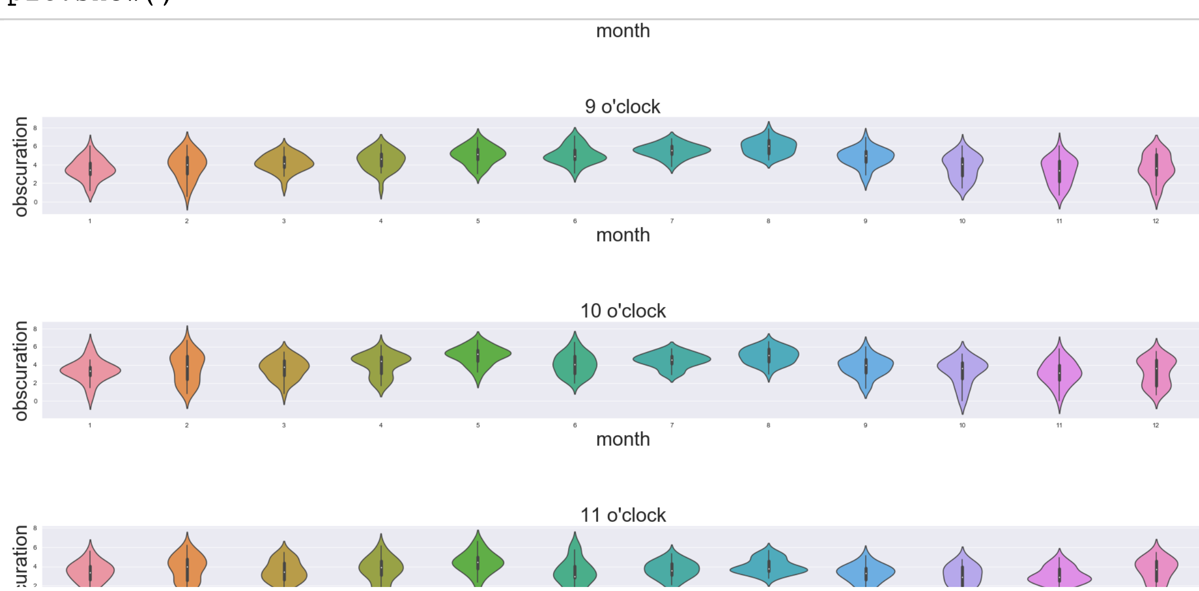
In [302]:
```
hourly_obscuration = pd.DataFrame(df.groupby(df.index.hour).averageObscurat
hourly_obscuration = hourly_obscuration.reset_index()
hourly_obscuration.columns = ['hour of the day', 'mean obscuration']
plt.figure(figsize=(16, 6))
sns.barplot(y='hour of the day', x='mean obscuration', color='grey', orient
plt.show()
```

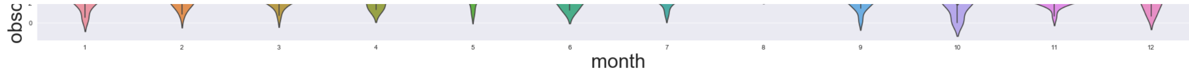



```
It looks like the hours between 10 AM and 4 PM are the least obscured.
But this is averaging out any changes over the course of the usual year.
How might the result take into account this sort of variation?:

*2. Averaged across the decade, for each hour, what's the typical range
obscuration values in a given month?*
```

In [303]:

```
f, axes = plt.subplots(24,1)
f.set_size_inches(32,128)
axes = axes.flatten()

def add_subplot(master_frame, index):
    hour_frame = hours.loc[index]
    hour_frame = hour_frame.reset_index()
    v = sns.violinplot(y='averageObscuration', x='month', data=hour_frame,
    v.set_xlabel("month", fontsize=30)
    v.set_ylabel("obscuration", fontsize=30)
    title_string = str(index) + " o'clock"
    v.set_title(title_string, fontsize=30)

hours = df
hours = pd.DataFrame(df.groupby([df.index.hour, df.index.day, df.index.mont
hours.index = hours.index.set_names(['hour', 'day', 'month'])
for x in range(24):
    add_subplot(hours, x)



left   =  0.125  # the left side of the subplots of the figure
right  =  0.9    # the right side of the subplots of the figure
bottom =  0.1    # the bottom of the subplots of the figure
top    =  0.9    # the top of the subplots of the figure
wspace =  .5     # the amount of width reserved for blank space between sub
hspace =  1.1    # the amount of height reserved for white space between su

# This function actually adjusts the sub plots using the above paramters
plt.subplots_adjust(
    left    =  left,
    bottom  =  bottom,
    right   =  right,
    top     =  top,
    wspace  =  wspace,
    hspace  =  hspace
)
plt.savefig('figures/hourlyAverageObscurationOverYear.png')
plt.show()
```



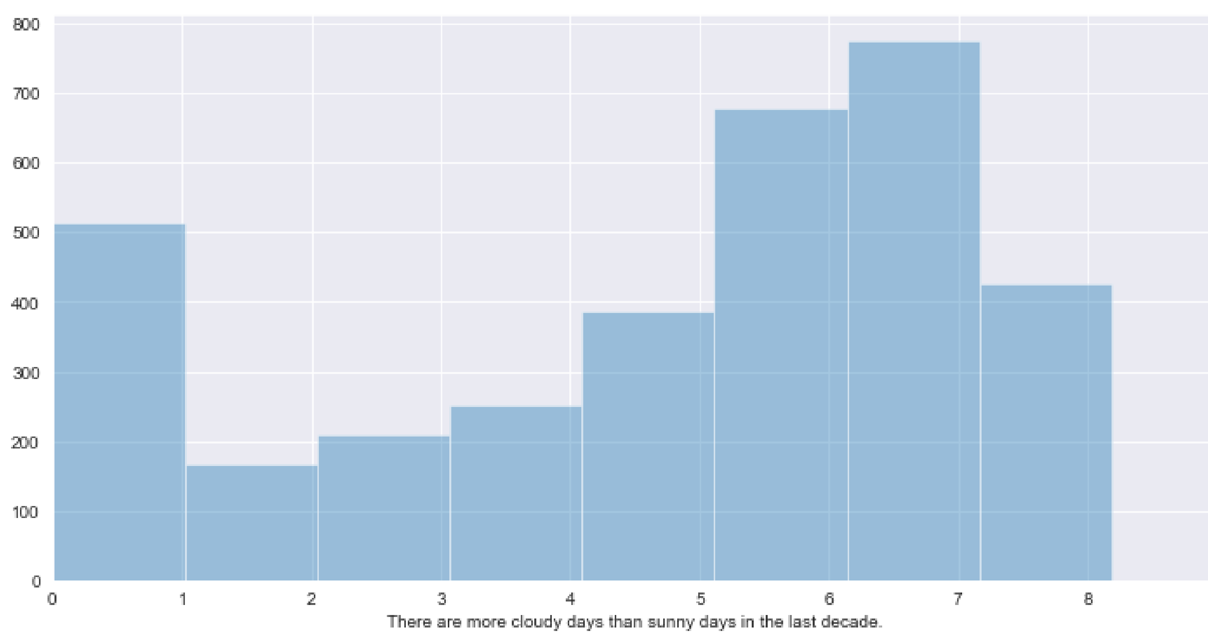


These plots reveal a couple important features of the area's climate. The nighttime and early morning hours become reliably very foggy in the "summer" months, which is the sort of relatively automatic watering that allowed crops like berries to thrive in this area in the early twentieth century. In contrast, late morning to early afternoon hours remain much more varied throughout the year. 1 PM and 2 PM in September and October show the best (most bottom-heavy) visibility distributions.

Before looking into obscuration by day more speciifcally, it's useful to have a look first at the distribution of daily average obscuration values across the ten years of hourly data. (That is, 365 * 10 daily average obscuration values):

3. Given the obscuration values in the data set from 0 to 9, where 0-2 is clear, 2-4 is scattered clouds, and 4-6 is overcast, and 6-9 is fully obscured, how many days in the last ten years fall into each category?

In [304]:
```
plt.figure(figsize=(12, 6))
sns.set_style('darkgrid')

obscuration = pd.DataFrame(df['averageObscuration'].groupby([df.index.date]
obscuration.name = 'There are more cloudy days than sunny days in the last

plt.xlim(0, 9)
sns.distplot(obscuration,bins=8, kde=False, norm_hist=False)
plt.show()
```

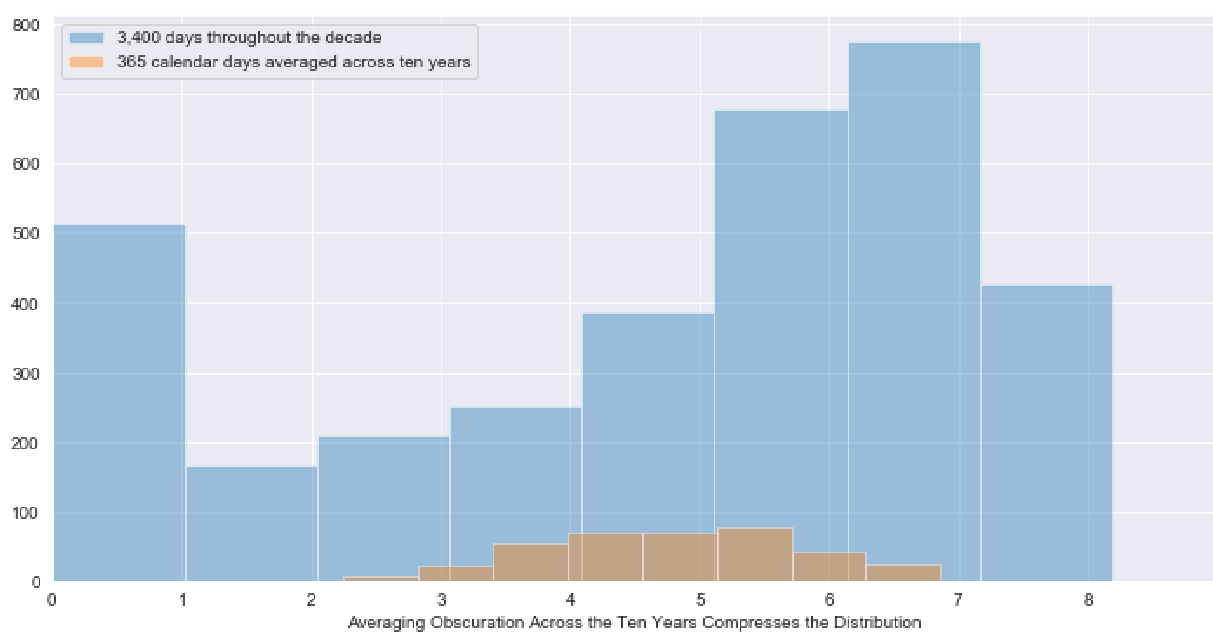


This eight-bin histogram illustrates the national weather service's mapping between sky condition categories and obscuration values on a scale from 0 to 9: 0-2 would get the code corresponding to "clear," 2-4 "scattered clouds," 4-6 "overcast," and 6-9 "fully obscured." (A rating of 9 is rarely used.) This shows us that weather in the area skews a bit toward cloudiness. The recommendation task here takes a binary view of this histogram: we're mainly interested in the days with an obscuration of less than or equal to around 4, the threshold for "overcast."

But this distribution takes each date in the last decade as a single datapoint. This view looks a bit different if we average each calendar day across the ten years of data to get 365 data points instead of 3,645:

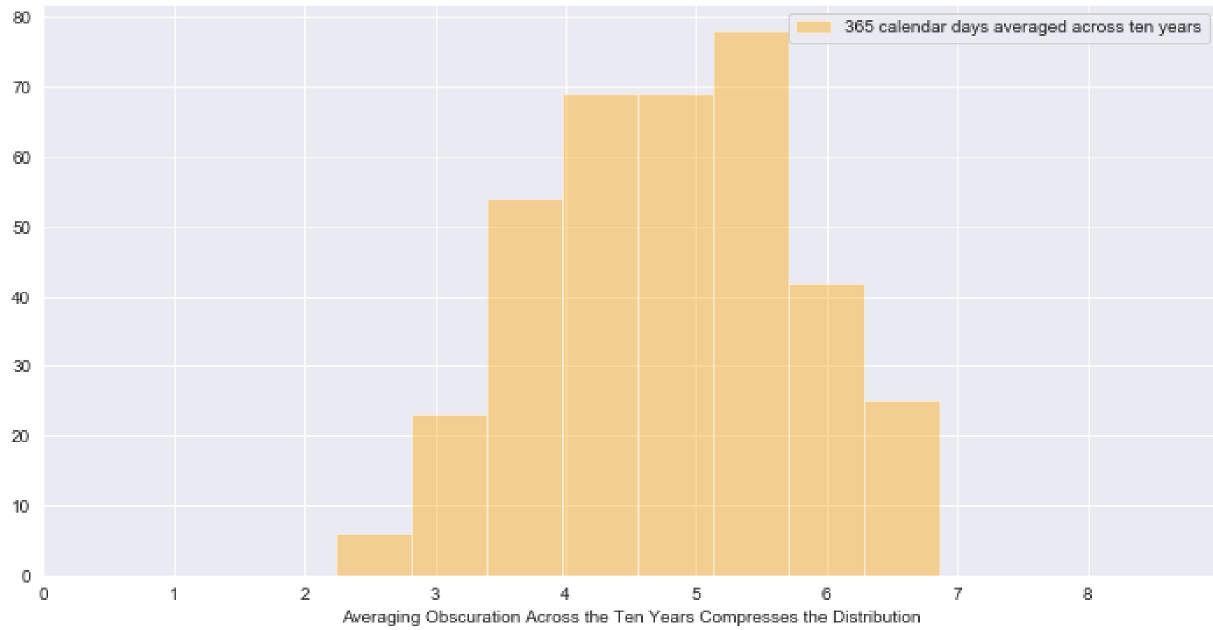
4. Averaging each annual calendar day across the decade, what's the year's distribution of obscuration values?

In [12]:
```
re(figsize=(12, 6))

ion = pd.DataFrame(df['averageObscuration'].groupby([df.index.date]).mean().
ion.name = 'There are more cloudy days than sunny days in the last decade.'
plot(obscuration,bins=8, kde=False, label='3,400 days throughout the decade

ion_year_averaged_across_decade = pd.DataFrame(df['averageObscuration'].grou
ion_year_averaged_across_decade.name = 'Averaging Obscuration Across the Ter
plot(obscuration_year_averaged_across_decade,bins=8, kde=False, label='365 c


im(0, 500)
(0, 9)
nd()
()
vefig('figures/dailyMeanForDatesAcrossDecade.png') # uncomment to write out
```



Let's have a closer look at the distribution of obscuration values for 365 calendar days each averaged over 10 yearly values:

In [305]:
```
plt.figure(figsize=(12, 6))
obscuration_year_averaged_across_decade = pd.DataFrame(df['averageObscurati
obscuration_year_averaged_across_decade.name = 'Averaging Obscuration Acros
sns.distplot(obscuration_year_averaged_across_decade,bins=8, color='orange'


# plt.ylim(0, 500)
plt.xlim(0, 9)
plt.legend()
plt.show()
# plt.savefig('figures/dailyMeanForDatesAcrossDecade.png') # uncomment to w
```



Averaging each day across its ten yearly values gives a distribution that looks more like a normal distribution, as the clearest and least clear outlier dates are averaged toward less extreme values.

After getting a sense of how daily obscuration distributes, we can focus in on the clear days.

5. Averaged over the decade, how many calendar days have been clear or nearly clear between 10 AM and 4 PM over the last decade (defined here as a "clearish day")?

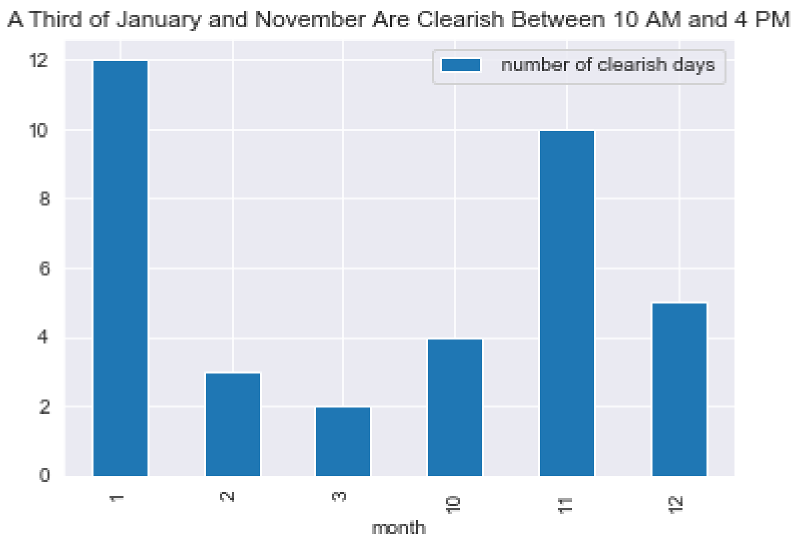
In [310]:
```
by_date = df[(df.index.hour >= 10) & (df.index.hour <= 16)] # get 10 AM to 
by_date = df.groupby([df.index.month, df.index.day]).averageObscuration.mea
by_date = by_date.sort_values()
by_date = by_date[by_date <= 3.5] # 3.5 is a conservative cut-off for a cle
print(str(len(by_date)) + " days have had a decade average obscuration rati
```

```
36 days have had a decade average obscuration rating of under 3.5.
```

A cut-off of 3.0 cut this number down from 36 days to 12 days, which means that scattered cloud days are especially important to this analysis.

6. Which months are these days in?

In [333]:
```
by_date = df[(df.index.hour >= 10) & (df.index.hour <= 16)] # get 10 AM to
by_date = df.groupby([df.index.month, df.index.day]).averageObscuration.mea
by_date = by_date.sort_values()
by_date = by_date[by_date <= 3.5] # 3.5 is a conservative cut-off for a cle
by_date = pd.DataFrame(by_date)
by_date.index = by_date.index.rename(["month", "day"])
# by_date = by_date.unstack(level=0)
by_date
by_date.groupby('month').count().rename(columns={'averageObscuration':' num
plt.show()
```

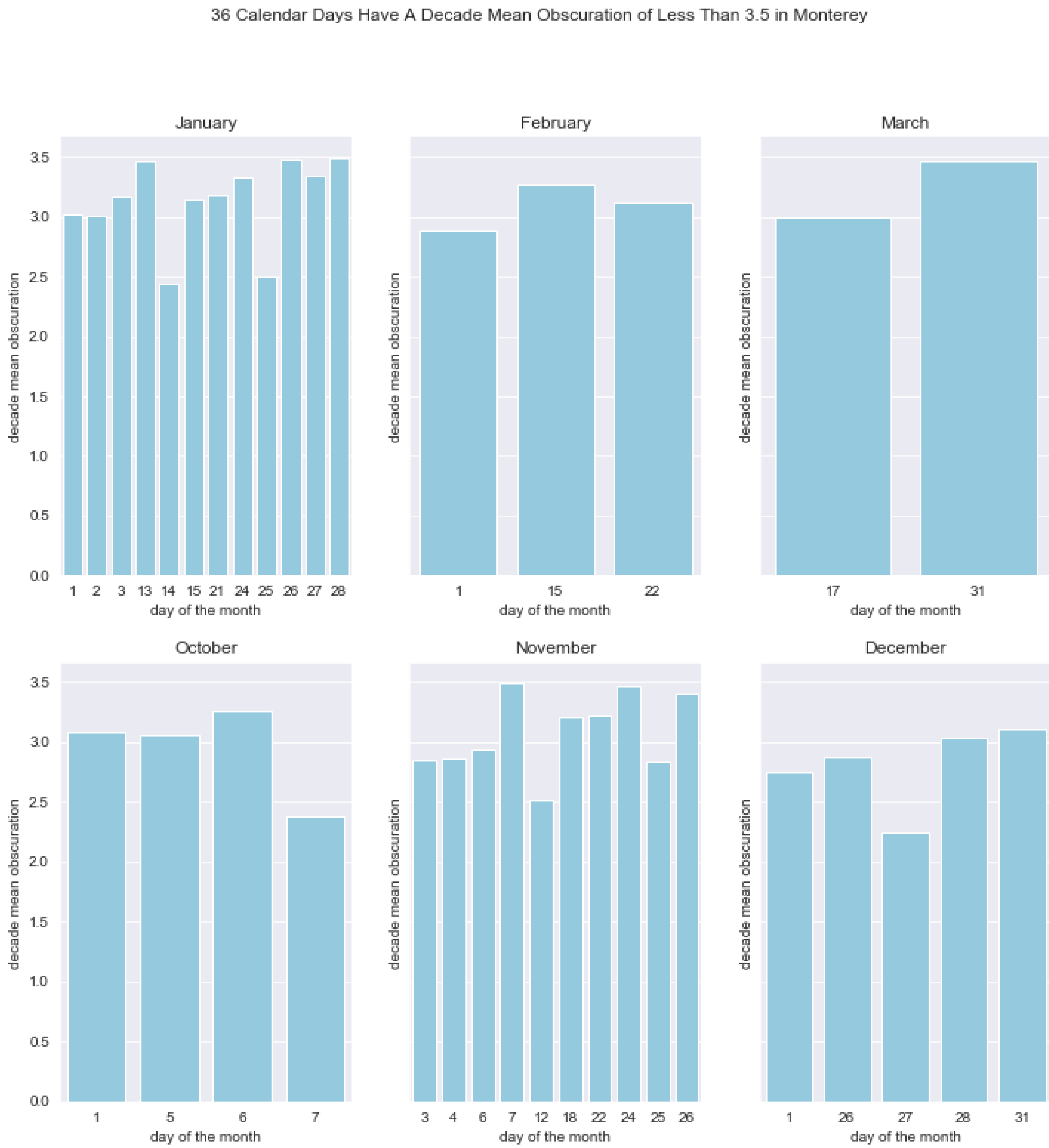


Public consensus in the area is that September and October have the nicest days, but the data reveal that this understanding might be conflating temperature with sky clarity: November and January have the most reliably clearish days. Next, it might be useful to map out all 36 of these days with their obscurations.

7. Which days of the year are the 36 days, and what is each day's daily average obscuration?

In [336]:

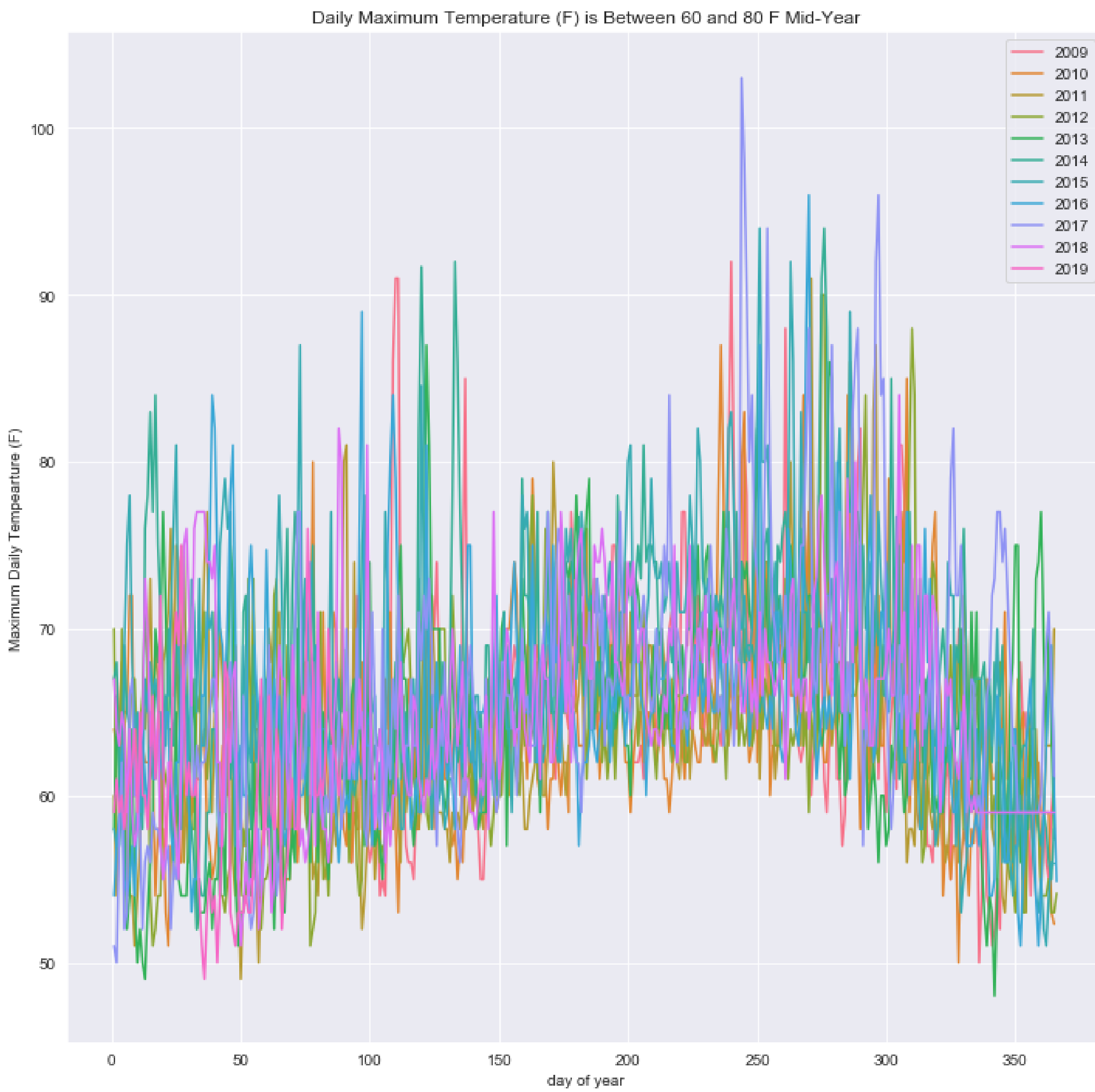
```
by_date = df[(df.index.hour >= 10) & (df.index.hour <= 16)] # get 10 AM to
by_date = df.groupby([df.index.month, df.index.day]).averageObscuration.mea
by_date = by_date.sort_values()
by_date = by_date[by_date <= 3.5] # 3.5 is a conservative cut-off for a cle
by_date = pd.DataFrame(by_date)
by_date.index = by_date.index.rename(["month", "day"])

# set up subplots
f, axes = plt.subplots(2,3, sharey='row')
f.set_size_inches(12,12)
axes = axes.flatten()

# set up title lookup
month_dict = {1: 'January', 2: 'February', 3: 'March', 10: 'October', 11: '

# plot a month
def plot_month(frame, month_index, plot_index):
    """Plots the decade mean obscuration for the index month's clearish day
    frame = frame.reset_index()
    frame = frame[frame['month'] == month_index]
    frame = frame.set_index('day')
    frame = frame.sort_index()
    b = sns.barplot(data=frame, x=frame.index, color='skyblue', y='averageO
    b.set_xlabel('day of the month')
    b.set_ylabel('decade mean obscuration')
    b.set_title(month_dict[month_index])

for plot, month in enumerate(set(by_date.index.get_level_values(0))):
    plot_month(by_date, month, plot)

plt.suptitle("36 Calendar Days Have A Decade Mean Obscuration of Less Than
plt.savefig('figures/clearishDaysByMonth.png')
plt.show()
```



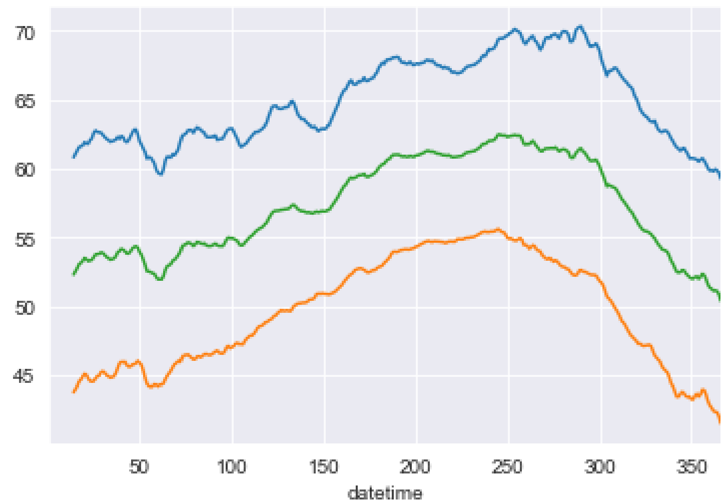
This suggests a correlation between temperature and obscuration, as there are so many clear days in winter months. Let's investigate that by plotting temperature range against obscuration. But first, let's have a look at temperature itself.

8. When is maximum daily temperature most and least chaotic?

In [225]:
```
plt.figure(figsize=(12, 12))

# processing setup
x = df
x = x.reset_index()

# wrangle: year columns, day of year index, daily high temperature values
x = x.set_index('datetime')
x = x.groupby([x.index.year, x.index.dayofyear]).DailyMaximumDryBulbTempera
x.index = x.index.rename(['year', 'day of year'])
x = x.unstack(level=0)

# plot
b = sns.lineplot(data=x, dashes=False)
b.set_title('Daily Maximum Temperature (F) is Between 60 and 80 F Mid-Year'
b.set_ylabel('Maximum Daily Tempearture (F)')
plt.legend()
plt.show()
```



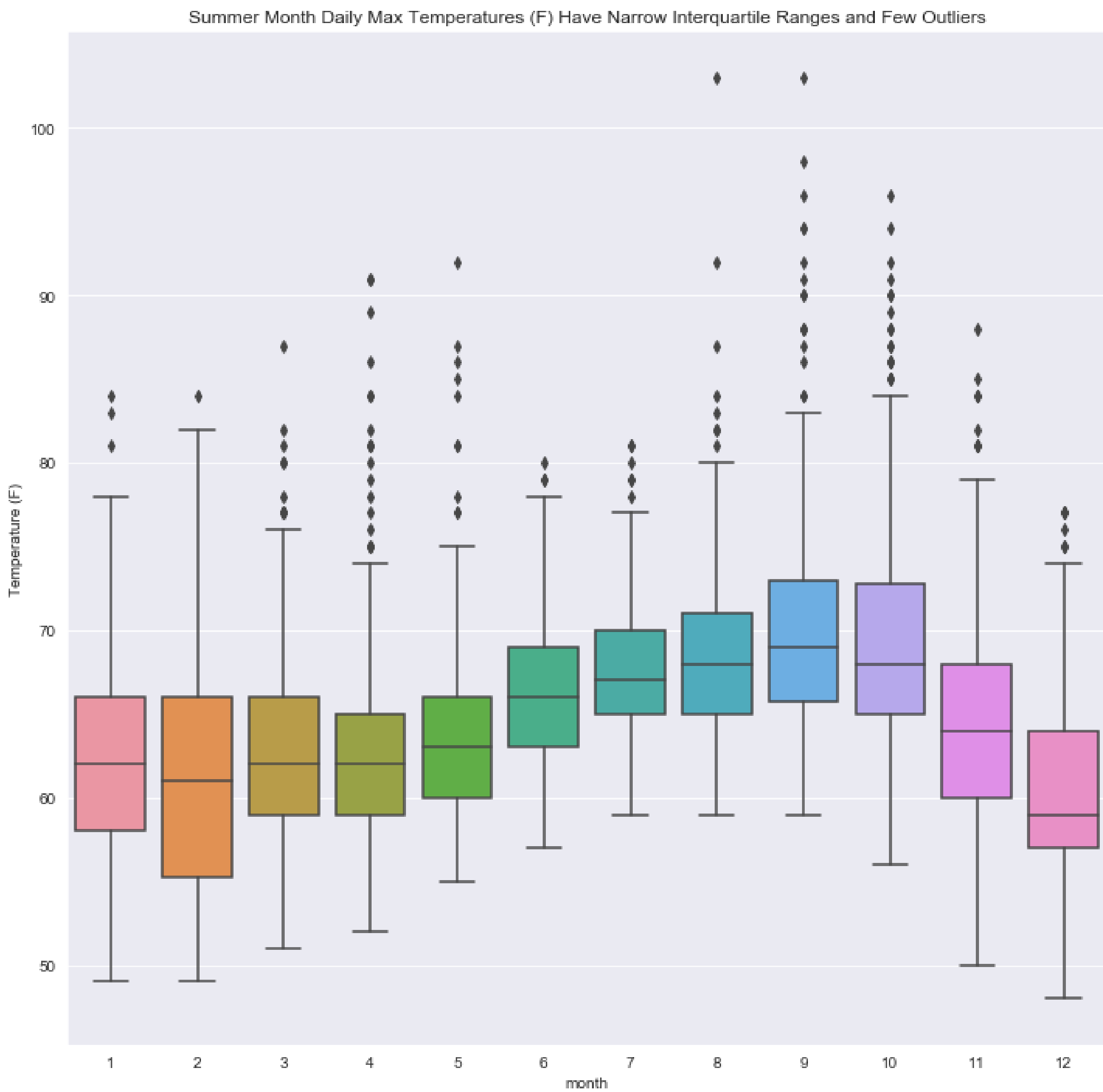
In [515]:
```
x = df
max_temp = x.groupby(df.index.dayofyear)['DailyMaximumDryBulbTemperature'].
min_temp = x.groupby(df.index.dayofyear)['DailyMinimumDryBulbTemperature'].

x['mean_temp'] = (df['DailyMaximumDryBulbTemperature'] + df['DailyMinimumDr
mean_temp = x.groupby(df.index.dayofyear)['mean_temp'].mean().rolling(14).m

# overlay max, min, mean
```



This plot of maximum daily temperature shows that days in the middle of the year have much less choatic variation than the rest of the year. Some boxplots per month should confirm this from the month-to-month view of the year.

9. Which months have the narrowest range of daily maximum temperatures and the fewest number of outliers across the decade?

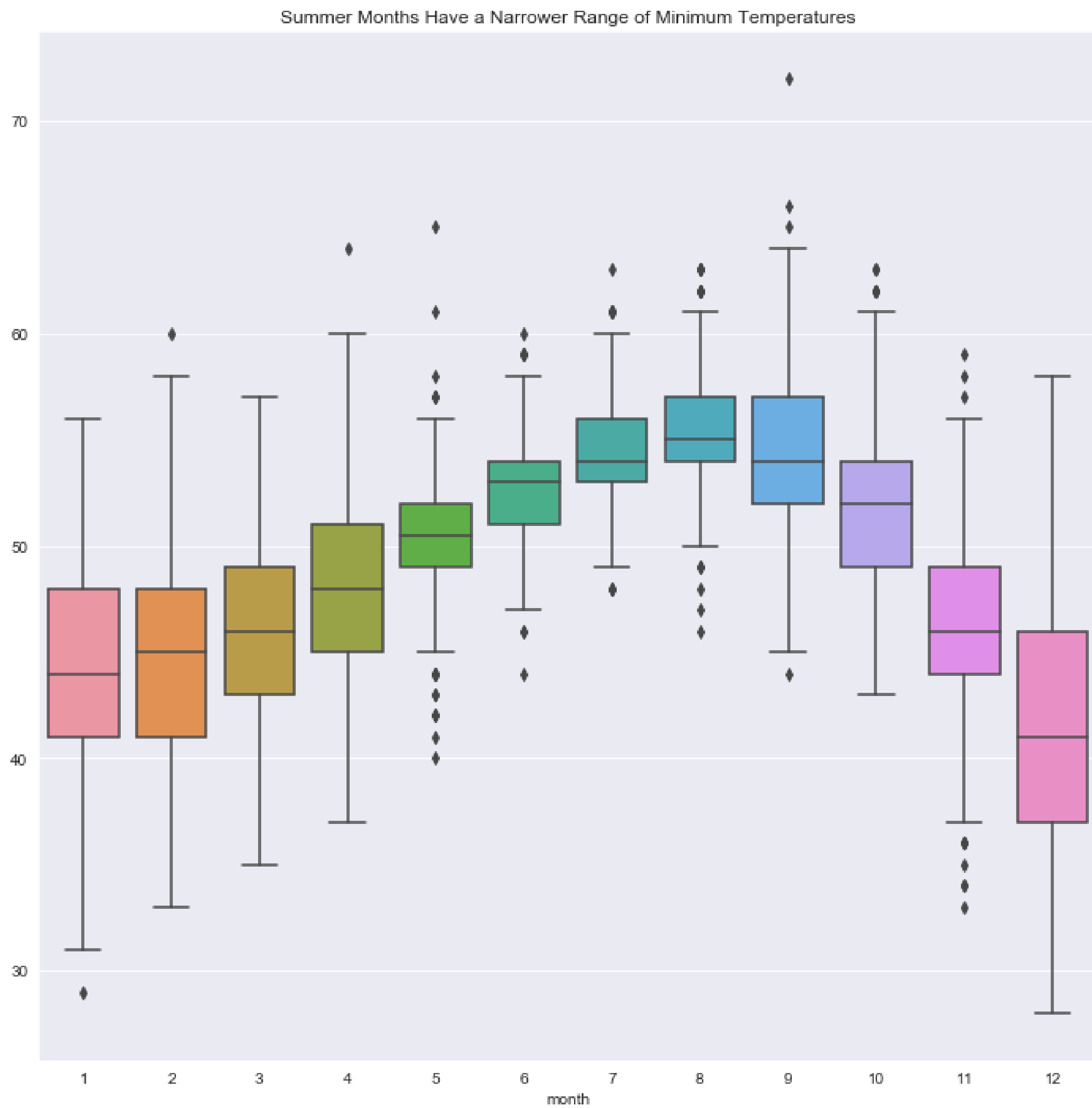
In [228]:
```
plt.figure(figsize=(12, 12))

# processing setup
x = df
x.reset_index()

# wrangle: year columns, day of year index, daily high temperature values
# mean does nothing here: all entries have same max value
x = x.groupby([x.index.month, x.index.date]).DailyMaximumDryBulbTemperature
x.index = x.index.rename(['month', 'date'])
x = x.unstack(level=0)
x.head(100)


# # plot
b = sns.boxplot(data=x)
b.set_title('Summer Month Daily Max Temperatures (F) Have Narrow Interquart
b.set_ylabel('Temperature (F)')
plt.show()
```



```
This plot confirms what we see in the previous line plot: the "summer"
months have a much narrower range of maximum temperatures than the other
months; June and July have the fewest outlier temperatures. February has
an exceptionally wide range of temperatures compared to the other months.
```

10. Which months have the narrowest range of daily minimum temperatures and the fewest number of outliers across the decade?

In [312]:
```
plt.figure(figsize=(12, 12))

# processing setup
x = df
x.reset_index()

# wrangle: year columns, day of year index, daily high temperature values
x = x.groupby([x.index.month, x.index.date]).DailyMinimumDryBulbTemperature
x.index = x.index.rename(['month', 'date'])
x = x.unstack(level=0)
x.head(100)


# # plot
sns.boxplot(data=x).set_title('Summer Months Have a Narrower Range of Minim
plt.show()
```

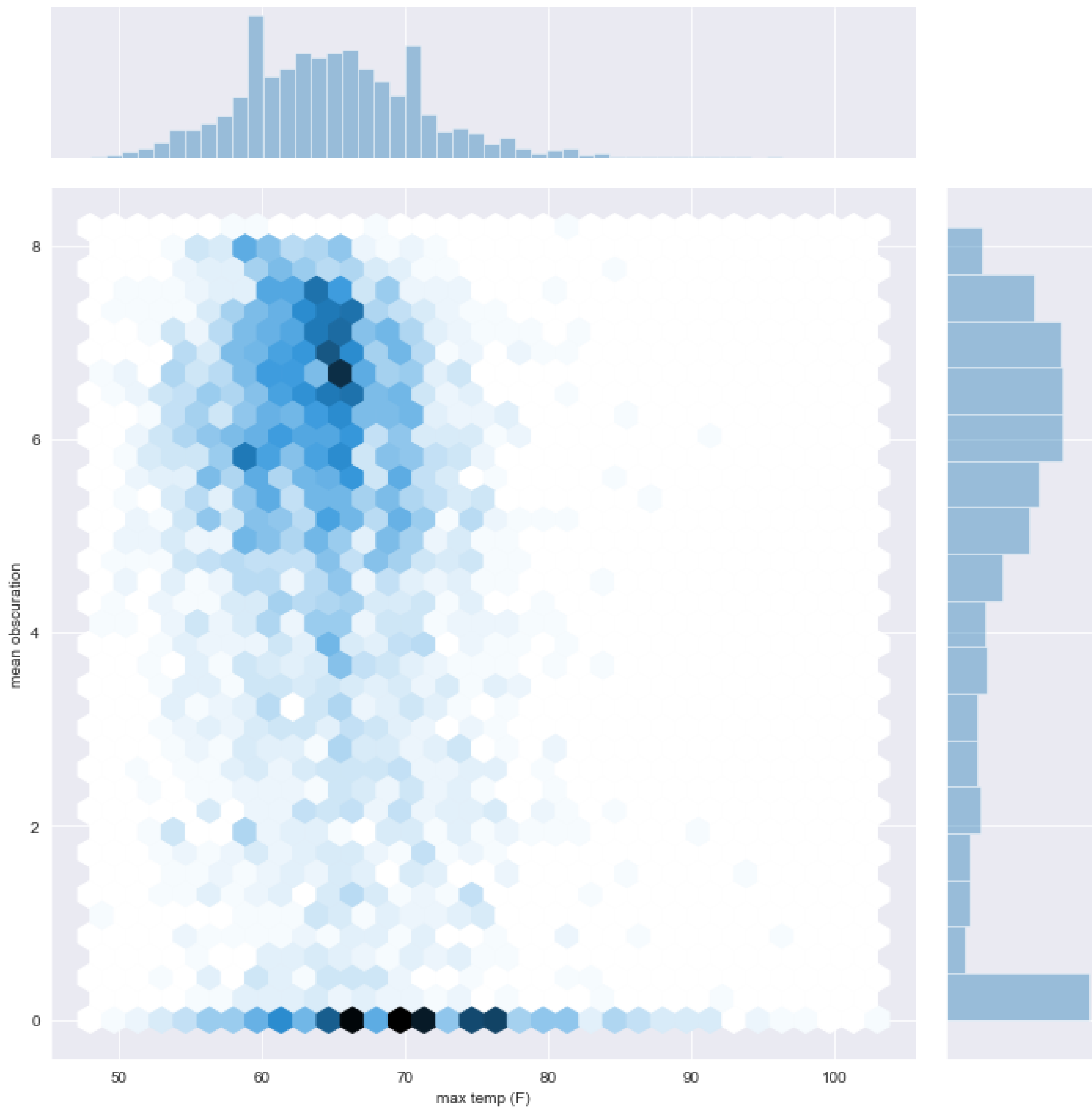


The minimum temperatures per month show a much more pronounced seasonal variation, with the "summer" months showing notably smaller interquartile ranges than the other months.

Now let's have a look at the correlation between maximum daily temperature and daily average sky obscuration:

11. Does sky obscuration correlate with daily maximum temperature?

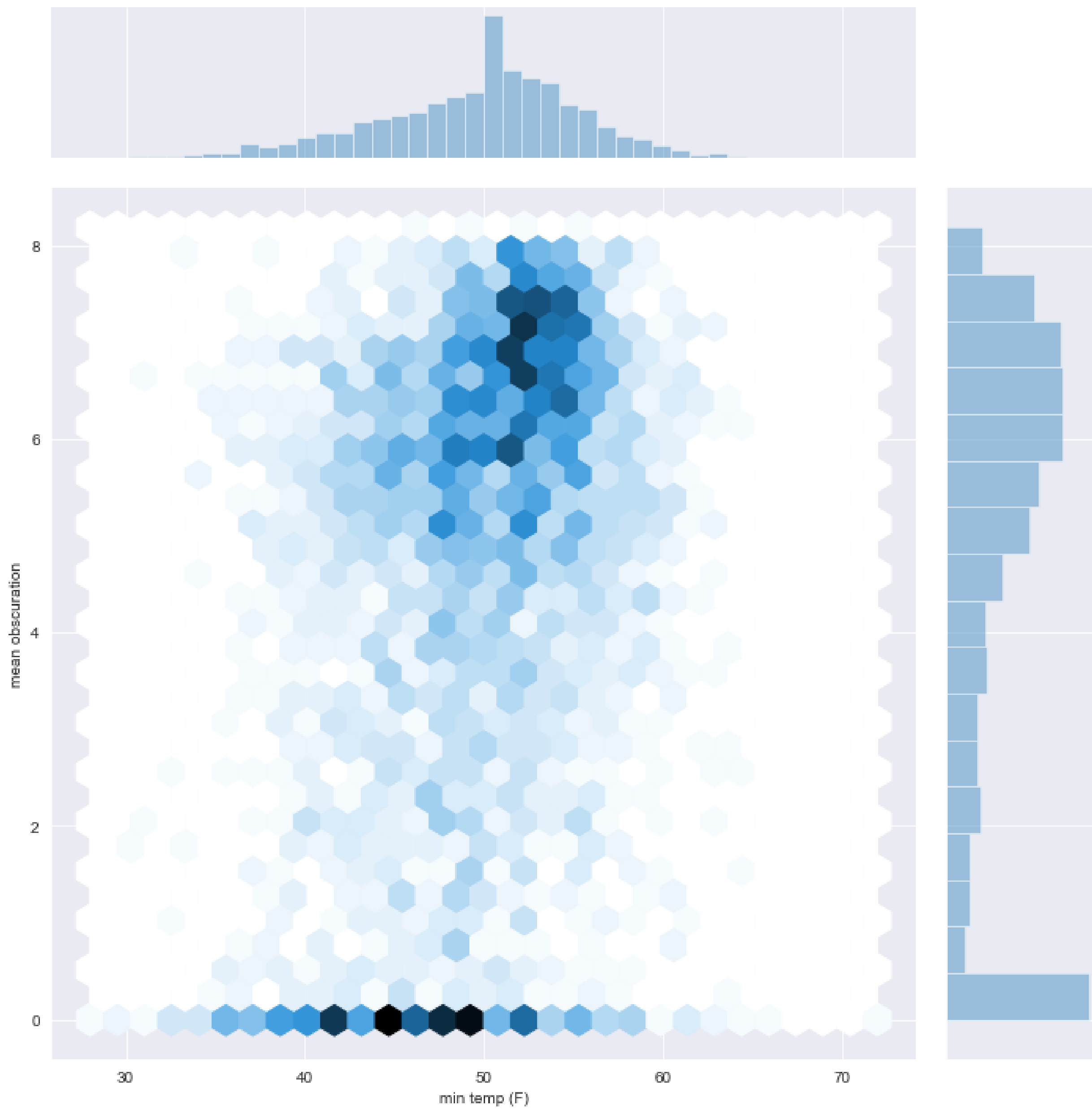
In [390]:
```
# processing setup
x = df
x.reset_index()
x.head()

# wrangle: create average obscuration and max temperature columns
x = pd.DataFrame(x.groupby([x.index.date, x['DailyMaximumDryBulbTemperature
x.index = x.index.rename(['date', 'max temp (F)'])
x = x.reset_index()
x = x.set_index(['date'])
x.columns = ['max temp (F)', 'mean obscuration']
x.head()

# plot
hexplot = sns.jointplot(x='max temp (F)', y='mean obscuration', height=10,
plt.show()
```



This plot and its accompanying histograms show a strong correlation between a range of temperatures and overcast weather: there are lots of overcast days with a maximum temperature between 60 and 70 degrees.

11. How does obscuration correlate with minimum daily temperature?

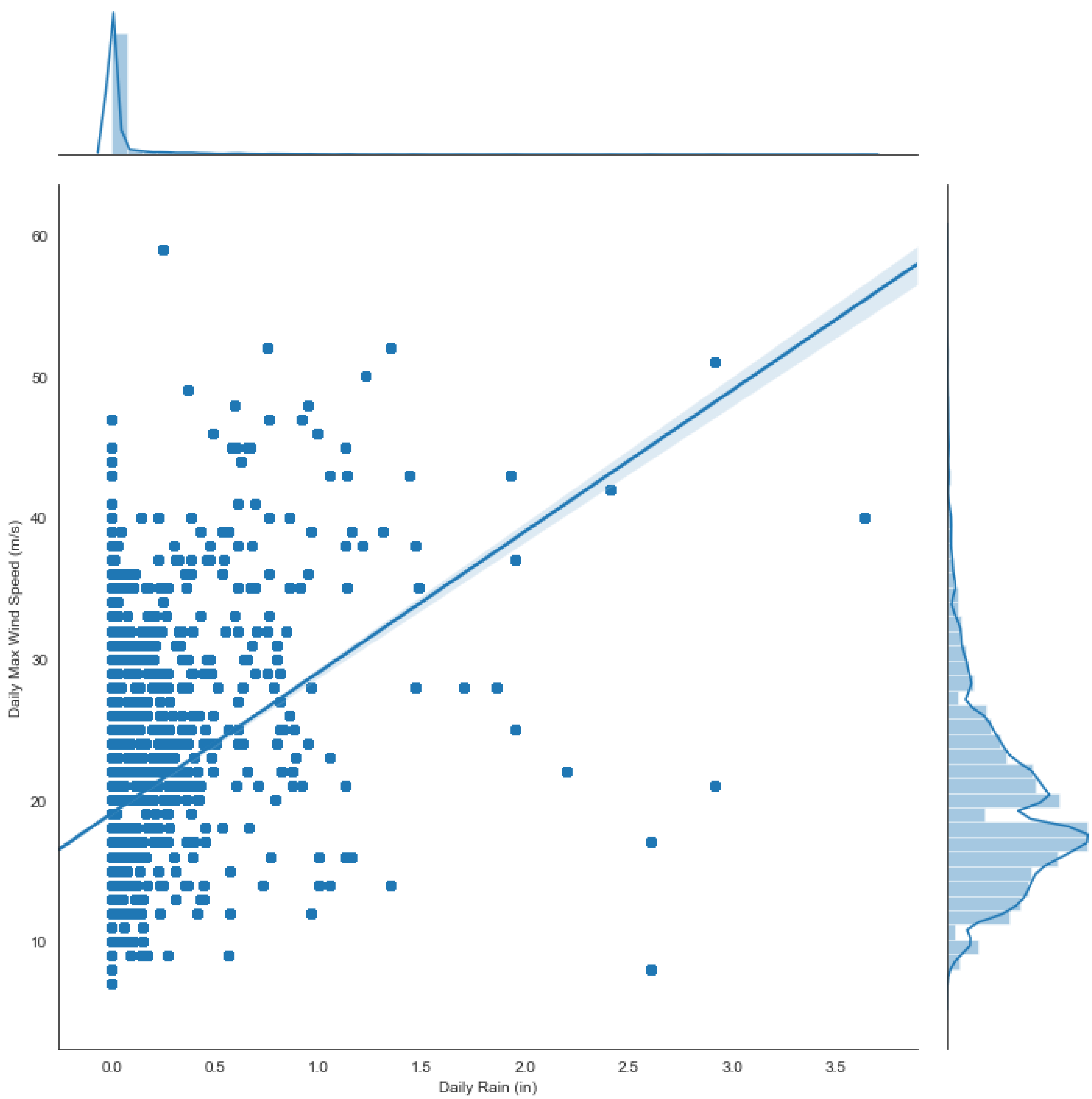
In [391]:
```
# processing setup
x = df
x.reset_index()
x.head()

# wrangle: create average obscuration and min temperature columns
x = pd.DataFrame(x.groupby([x.index.date, x['DailyMinimumDryBulbTemperature
x.index = x.index.rename(['date', 'max temp (F)'])
x = x.reset_index()
x = x.set_index(['date'])
x.columns = ['min temp (F)', 'mean obscuration']
x.head()

# plot
hexplot = sns.jointplot(x='min temp (F)', y='mean obscuration', height=10,
plt.show()
```



Lastly, we'll have a look at rainfall and wind patterns.

12. Is there a correlation between daily peak wind speed and daily precipitation?

In [457]:
```
# processing setup
x = df
x.reset_index()
x.head()

# wrangle: rename wind speed and precipitation columns
x = x.rename(columns={'DailyPrecipitation': "Daily Rain (in)", 'DailyPeakWi

# plot
sns.set_style('white')
j = sns.jointplot(x='Daily Rain (in)', y='Daily Max Wind Speed (m/s)', heig
plt.show()
```



It looks like higher wind speeds correlate with more daily precipitation, but the confidence interval spreads as the amount of rain increases, because we have many fewer data points. It also looks like most days have between 0 and 1 mm of precipitation, which makes sense, as California experienced a historic drought for a large part of the last decade. But in the past few years, California has also seen historic rains, thanks to increasingly common atmospheric river events.

13. How many days had no rain in each year of the last decade?

In [401]:
```
x = df[(df.index.year >= 2010) & (df.index.year < 2019)] # choose 2010 thro
x = x.groupby([x.index.year, x.index.date]).DailyPrecipitation.sum()
x.index = x.index.rename(['year', 'date'])
x = x[x == 0]
x = x.reset_index()
x = x.groupby('year').count().rename(columns={'DailyPrecipitation':'Days Wi
x
```

Out[401]:

	Days Without Rain
year	
2010	265
2011	295
2012	303
2013	341
2014	303
2015	324
2016	295
2017	286
2018	312

Several online sources report that the average annual rainfall in Monterey is around 500 mm. How do these data align with that figure? The drought and the atmospheric river events should both pull the average lower and higher respectively — which trend wins out?

14. What has the average annual of rainfall been for the last decade?

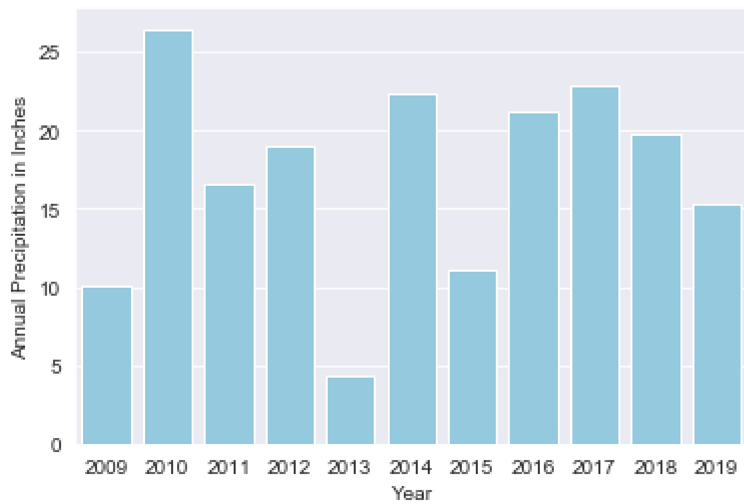
In [402]:
```
x = df
x = x.groupby([x.index.year, x.index.date])['DailyPrecipitation'].first()
x.index = x.index.rename(['year', 'date'])
x = x.groupby(['year']).sum().mean()
x
```

Out[402]:
```
17.19363636363636
```

17.2 inches is about 436 mm, which brings in the average a bit below a widely reported average figure.

15. What's the annual rainfall for each year between 2010 and 2018 inclusive?

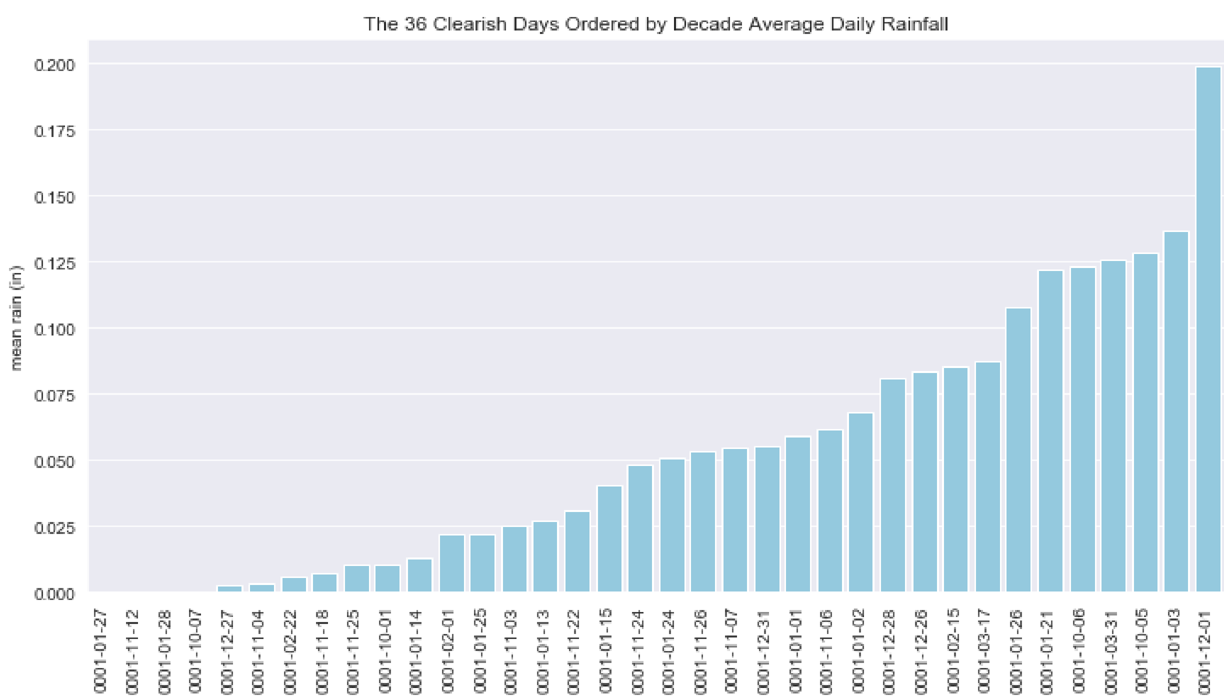
In [458]:
```
x = df
x = x.groupby([x.index.year, x.index.date])['DailyPrecipitation'].first()
x.index = x.index.rename(['Year', 'Date'])
x = pd.DataFrame(x.groupby(['Year']).sum())
x.columns = ['Annual Precipitation in Inches']
x = x.reset_index()

# plot
sns.set_style('darkgrid')
sns.barplot(x='Year', y='Annual Precipitation in Inches', color='skyblue',
plt.show()
```



There's been a wide range of annual rainfalls in the last decade, from 4.33 inches in 2013 to 22.85 inches in 2017.

16. What is the daily precipitation for the 36 clearish calendar days?

In [503]:
```
plt.figure(figsize=(12, 6))

by_date = df[(df.index.hour >= 10) & (df.index.hour <= 16)] # get 10 AM to 

# mean the average daily obscuration and keep the first value for daily pre
by_date = df.groupby([df.index.date]).agg({'DailyPrecipitation': 'first', '

# further average both obscuration and daily rainfall by calendar day
by_date = df.groupby([df.index.month, df.index.day]).agg({'DailyPrecipitati
by_date.index = by_date.index.rename(['month', 'day'])
by_date.columns = ['mean rain (in)', 'mean obscuration']
by_date = by_date.reset_index()

# filter out clearish days
by_date = by_date[by_date['mean obscuration'] <= 3.5] # 3.5 is a conservati

# sort by ascending rainfall
by_date = by_date.sort_values(by='mean rain (in)')

# add a date column to serve as the index
by_date['date'] = by_date.apply(lambda x: date(year=1, month=int(x['month']
by_date


# plot
b = sns.barplot(x='date', y='mean rain (in)', color='skyblue', data=by_date
b.set_title('The 36 Clearish Days Ordered by Decade Average Daily Rainfall'
plt.xticks(rotation=90)
plt.show()
```



In [ ]: