In [33]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [34]:
```
import numpy as np
import pandas as pd
```

In [35]:
```
import datetime as dt
```

Reflect Tables into SQLAlchemy ORM

In [36]:
```
# Python SQL toolkit and Object Relational Mapper
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, func
```

In [37]:
```
engine = create_engine("sqlite:///Resources/hawaii.sqlite")
```

In [38]:
```
# reflect an existing database into a new model
base = automap_base()
# reflect the tables
base.prepare(engine, reflect = True)
```

In [39]:
```
# We can view all of the classes that automap found
base.classes.keys()
```

Out[39]:
```
['measurement', 'station']
```

In [40]:
```
# Save references to each table
m = base.classes.measurement
s = base.classes.station
```

In [41]:
```
# Create our session (link) from Python to the DB
session = Session(engine)
```

In [42]:
```
station_cols = s.__table__.columns
print(station_cols)
```

```
['station.id', 'station.station', 'station.name', 'station.latitude', 'station.longitude', 'station.elevation']
```

In [43]:
```
meas_cols = m.__table__.columns
print(meas_cols)
```

```
['measurement.id', 'measurement.station', 'measurement.date', 'measurement.prcp', 'measurement.tobs']
```

Exploratory Climate Analysis

In [44]:
```
# Design a query to retrieve the last 12 months of precipitation data and plot the results
max_date = session.query(func.max(m.date)).all()
max_date_index = max_date[0][0]
print(max_date)
```

```
[('2017-08-23',)]
```

In [45]:
```
# Calculate the date 1 year ago from the last data point in the database
datetime_object = dt.datetime.strptime(max_date_index, '%Y-%m-%d')
datetime_object_past = datetime_object - dt.timedelta(days = 365)
datetime_object_past
```

Out[45]:
```
datetime.datetime(2016, 8, 23, 0, 0)
```

In [46]:
```
# Perform a query to retrieve the data and precipitation scores
prcp_result = session.query(m.date, m.prcp).filter(m.date > datetime_object_past).all()
```

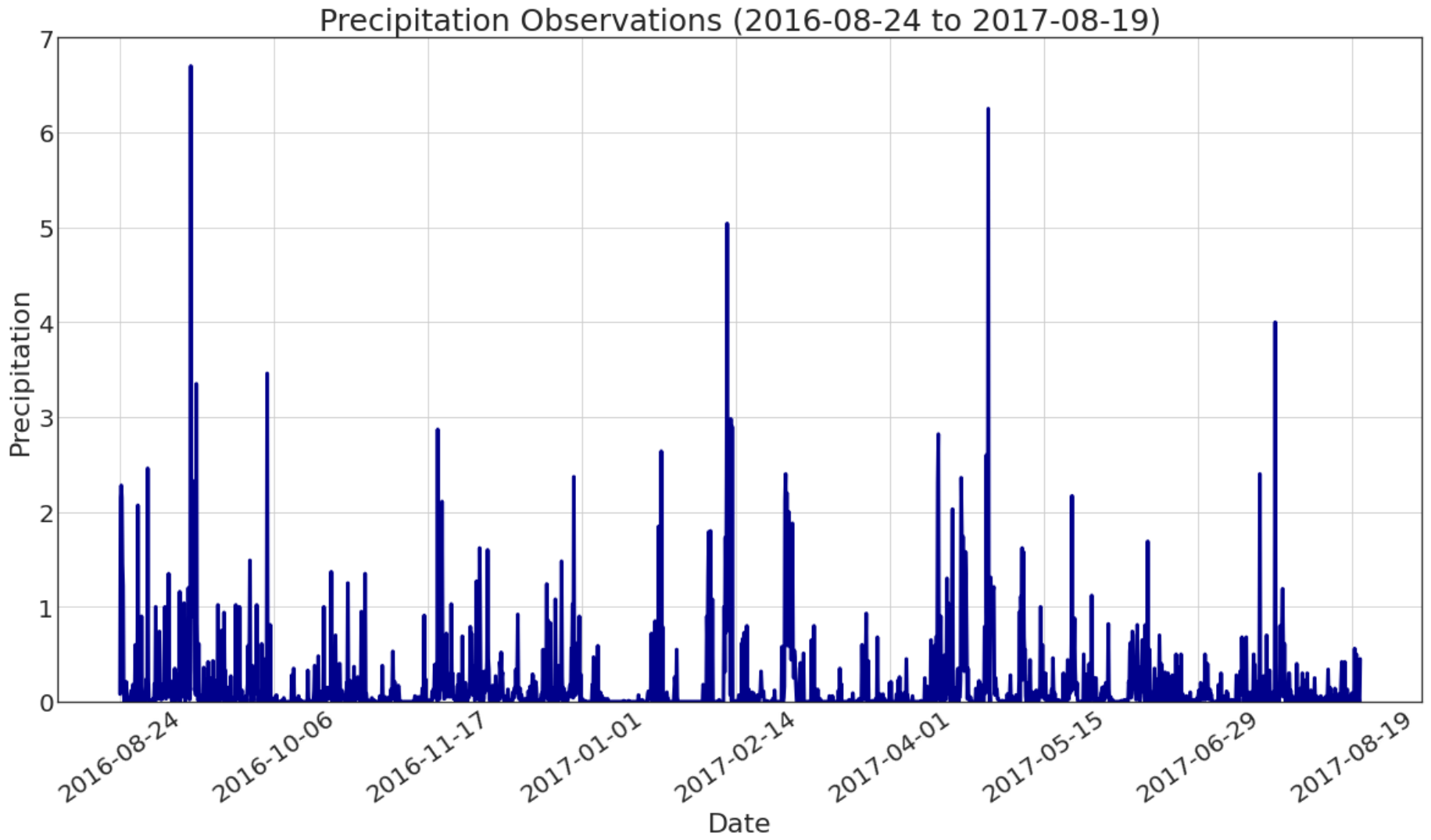
In [47]:
```
# Save the query results as a Pandas DataFrame...
prcp_result_df = pd.DataFrame(prcp_result)

#...and set the index to the date column
# Sort the dataframe by date
prcp_result_df = prcp_result_df.set_index("date").sort_values(by = "date")
prcp_result_df = prcp_result_df.dropna()
prcp_result_df
```

Out[47]:

	prcp
date	
2016-08-24	0.08
2016-08-24	2.15
2016-08-24	2.28
2016-08-24	2.15
2016-08-24	1.45
...	...
2017-08-22	0.00
2017-08-23	0.08
2017-08-23	0.00
2017-08-23	0.00
2017-08-23	0.45

2015 rows × 1 columns

In [48]:
```
with plt.style.context("seaborn-white"):
    plt.rcParams["axes.grid"]=True
    prcp_result_df.plot(rot=35, figsize=(20,10), linewidth=3, color="darkblue", legend=False)
    plt.title("Precipitation Observations (2016-08-24 to 2017-08-19)", fontsize=25)
    plt.xlabel("Date", fontsize=22)
    plt.xticks(fontsize=20)
    plt.ylabel("Precipitation", fontsize=22)
    plt.yticks(fontsize=20)
    plt.ylim(0,7)
plt.savefig("Temperature Observations")
```



In [49]:
```
# Use Pandas to calcualte the summary statistics for the precipitation data
prcp_result_df.describe()
```

Out[49]:

	prcp
count	2015.000000
mean	0.176462
std	0.460288
min	0.000000
25%	0.000000
50%	0.020000
75%	0.130000
max	6.700000

In [50]:
```
# Design a query to show how many stations are available in this dataset?
stations = session.query(func.count(s.station)).all()
stations
```

Out[50]:
```
[(9)]
```

In [51]:
```
#number of records
records = session.query(func.count(m.station)).group_by(m.station).all()
records
```

Out[51]:
```
[(1979), (2709), (2202), (2612), (1372), (511), (2772), (2724), (2669)]
```

In [52]:
```
# What are the most active stations? (i.e. what stations have the most rows)?
session.query(m.station, func.count(m.station)).group_by(m.station).all()
```

Out[52]:
```
[('USC00511918', 1979),
 ('USC00513117', 2709),
 ('USC00514830', 2202),
 ('USC00516128', 2612),
 ('USC00517948', 1372),
 ('USC00518838', 511),
 ('USC00519281', 2772),
 ('USC00519397', 2724),
 ('USC00519523', 2669)]
```

In [53]:
```
# List the stations and the counts in descending order.
station_records = session.query(m.station, func.count(m.station)).group_by(m.station).order_by(func.count(m.station).de
sc()).all()
station_records
```

Out[53]:
```
[('USC00519281', 2772),
 ('USC00519397', 2724),
 ('USC00513117', 2709),
 ('USC00519523', 2669),
 ('USC00516128', 2612),
 ('USC00514830', 2202),
 ('USC00511918', 1979),
 ('USC00517948', 1372),
 ('USC00518838', 511)]
```

In [54]:
```
# Using the station id from the previous query, calculate the lowest temperature recorded,
#highest temperature recorded, and average temperature of the most active station?
session.query(m.station, func.max(m.tobs), func.avg(m.tobs), func.min(m.tobs)).filter(m.station == "USC00519281").all()
```

Out[54]:
```
[('USC00519281', 85.0, 71.66378066378067, 54.0)]
```

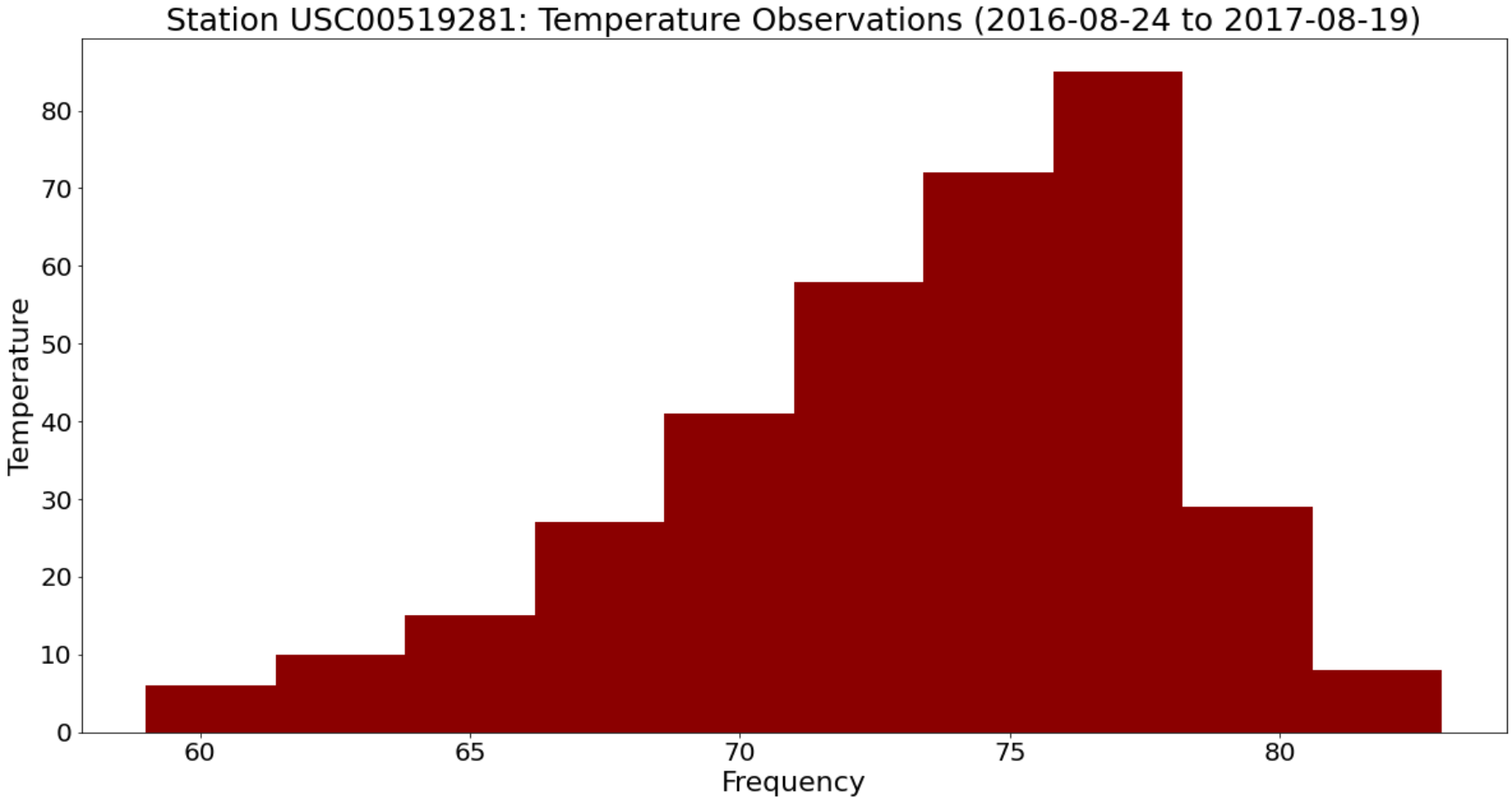
In [55]:
```
# Choose the station with the highest number of temperature observations.
most_tobs = session.query(m.station, func.max(m.tobs))

# Query the last 12 months of temperature observation data for this station...
temp_obs = session.query(m.date, m.tobs).filter(m.date > datetime_object_past).filter(m.station == "USC00519281").all()
temp_obs_df =  pd.DataFrame(temp_obs)
temp_obs_df = temp_obs_df.set_index("date")
temp_obs_df
```

Out[55]:

date	tobs
2016-08-24	77.0
2016-08-25	80.0
2016-08-26	80.0
2016-08-27	75.0
2016-08-28	73.0
...	...
2017-08-14	77.0
2017-08-15	77.0
2017-08-16	76.0
2017-08-17	76.0
2017-08-18	79.0

351 rows × 1 columns

In [56]:
```
#...and plot the results as a histogram
temp_obs_df.plot(kind = "hist", figsize=(20,10), color="darkred", legend=False).set_facecolor("white")
plt.title("Station USC00519281: Temperature Observations (2016-08-24 to 2017-08-19)", fontsize = 25)
plt.xlabel("Frequency", fontsize = 22)
plt.xticks(rotation="horizontal", fontsize = 20)
plt.ylabel("Temperature", fontsize = 22)
plt.yticks(rotation="horizontal", fontsize=20)
plt.show()
plt.savefig("Temperature Observations")
```




```
<Figure size 432x288 with 0 Axes>
```