In [1]:
```
%matplotlib inline
from matplotlib import style
style.use('fivethirtyeight')
import matplotlib.pyplot as plt
```

In [2]:
```
import numpy as np
import pandas as pd
import datetime as dt
```

Reflect Tables into SQLAlchemy ORM

In [3]:
```
# 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 [4]:
```
# create engine to hawaii.sqlite
engine = create_engine("sqlite:///Resources/hawaii.sqlite")
```

In [5]:
```
# reflect an existing database into a new model
Base = automap_base()
# reflect the tables
Base.prepare(engine, reflect=True)
```

In [6]:
```
# View all of the classes that automap found
Base.classes.keys()
```

Out[6]:
```
['measurement', 'station']
```

In [7]:
```
# Save references to each table
Measurement = Base.classes.measurement
Station = Base.classes.station
```

In [8]:
```
# Create our session (link) from Python to the DB
session = Session(engine)
```

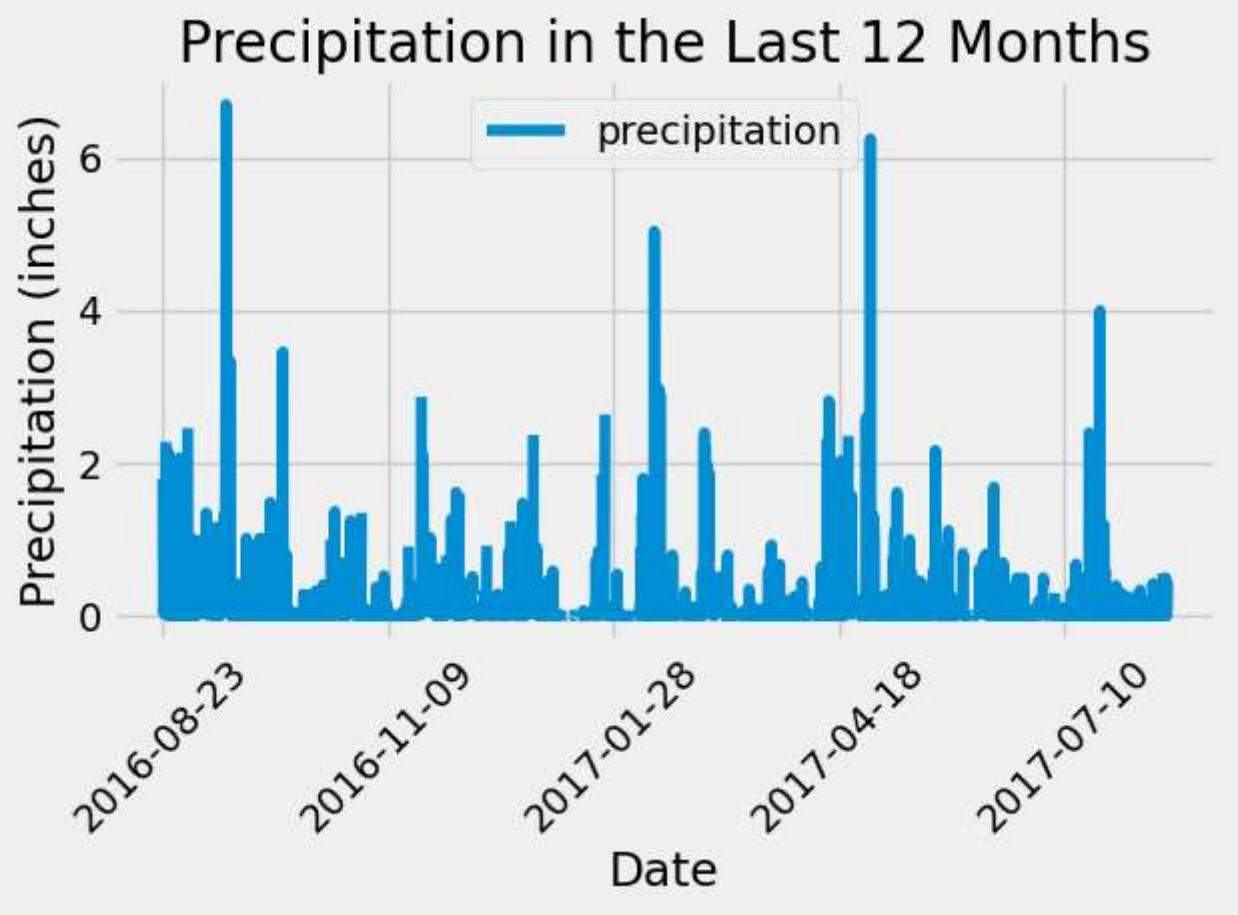
Exploratory Precipitation Analysis

In [9]:
```
# Find the most recent date in the data set.
most_recent_date = session.query(func.max(Measurement.date)).scalar()
most_recent_date
```

Out[9]:
```
'2017-08-23'
```

In [10]:
```
# Design a query to retrieve the last 12 months of precipitation data and plot the res
# Starting from the most recent data point in the database.

# Calculate the date one year from the last date in data set.
prev_year = dt.date(2017, 8, 23) - dt.timedelta(days=365)
```

```
# Perform a query to retrieve the data and precipitation scores
results = session.query(Measurement.date, Measurement.prcp).filter(Measurement.date >=

# Save the query results as a Pandas DataFrame. Explicitly set the column names
df = pd.DataFrame(results, columns=['date', 'precipitation'])
df.set_index('date', inplace=True)

# Sort the dataframe by date
df.sort_index(inplace=True)

# Use Pandas Plotting with Matplotlib to plot the data
df.plot(rot=45)
plt.xlabel("Date")
plt.ylabel("Precipitation (inches)")
plt.title("Precipitation in the Last 12 Months")
plt.tight_layout()
plt.show()
```



In [11]:
```
# Use Pandas to calculate the summary statistics for the precipitation data
df.describe()
```

Out[11]:

	precipitation
count	2021.000000
mean	0.177279
std	0.461190
min	0.000000
25%	0.000000
50%	0.020000
75%	0.130000
max	6.700000

Exploratory Station Analysis

In [12]:
```
# Design a query to calculate the total number of stations in the dataset
session.query(func.count(Station.station)).all()
```

Out[12]:
```
[(9,)]
```

In [13]:
```
# Design a query to find the most active stations (i.e. which stations have the most r
# List the stations and their counts in descending order.
session.query(Measurement.station,func.count(Measurement.station)).\
group_by(Measurement.station).order_by(func.count(Measurement.station).desc()).all()
```

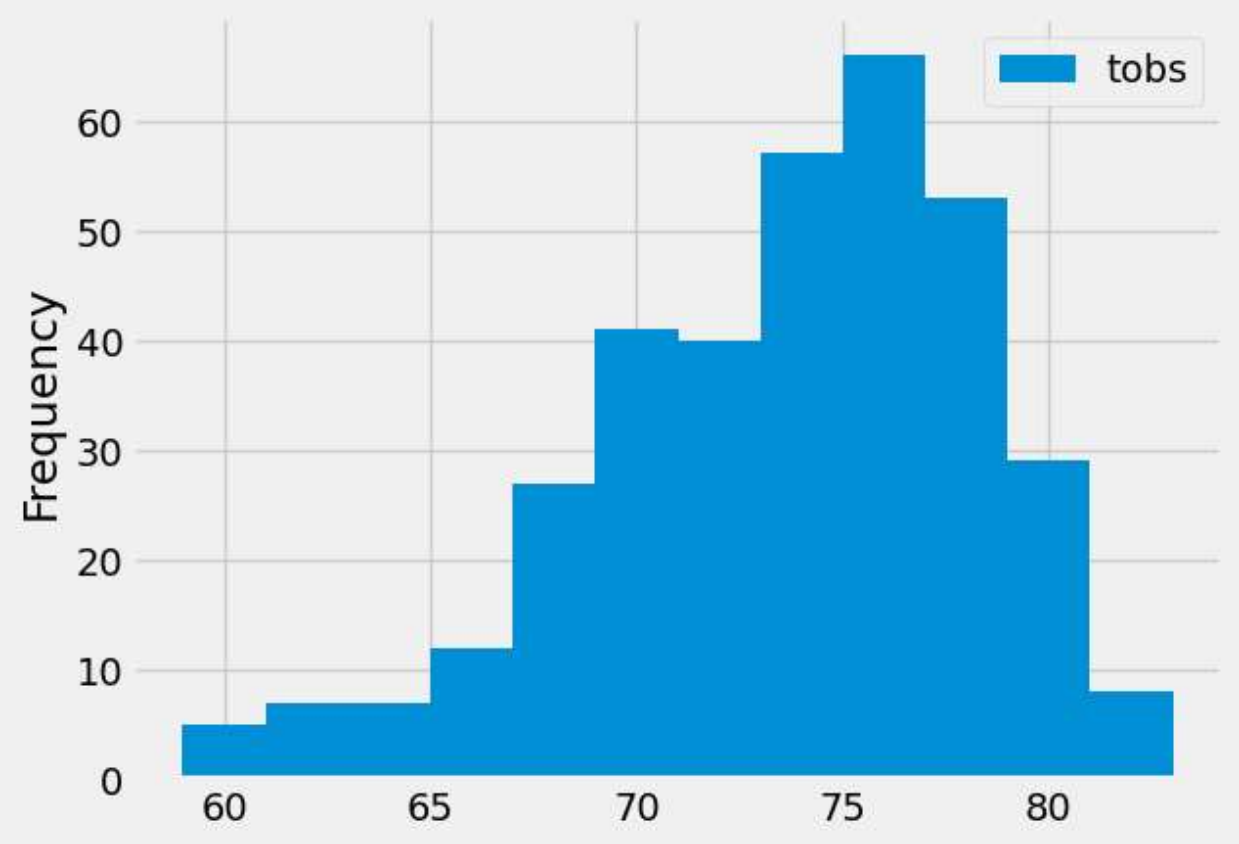
Out[13]:
```
[('USC00519281', 2772),
 ('USC00519397', 2724),
 ('USC00513117', 2709),
 ('USC00519523', 2669),
 ('USC00516128', 2612),
 ('USC00514830', 2202),
 ('USC00511918', 1979),
 ('USC00517948', 1372),
 ('USC00518838', 511)]
```

In [14]:
```
# Using the most active station id from the previous query, calculate the lowest, high
session.query(func.min(Measurement.tobs), func.max(Measurement.tobs), func.avg(Measure
filter(Measurement.station == 'USC00519281').all()
```

Out[14]:
```
[(54.0, 85.0, 71.66378066378067)]
```

In [15]:
```
# Using the most active station id
# Query the last 12 months of temperature observation data for this station and plot t
results = session.query(Measurement.tobs).\
filter(Measurement.station == 'USC00519281').\
filter(Measurement.date >= prev_year).all()
df = pd.DataFrame(results,columns = ['tobs'])
df.plot.hist(bins=12)
```

Out[15]:
```
<Axes: ylabel='Frequency'>
```



Close Session

In [16]:
```
# Close Session
session.close()
```