Climate Analysis

Analysis will be performed on temperature, precipitation, and stations.

Technical Report

- Honolulu normally has more than 6 inches of rain in twelve months period.
- The most active station indicates the lowest temperature (54.0°F), highest temperature (85.0°F), and average (71.7°F) of the last twelve months.
- The chosen trip dates were from 2018-01-01 to 2018-01-05. The temperature range for the trip was (low 62°F and high 77°F) while the average predicted temperatures stay in the 69's.

In [66]:

```
#Dependencies
%matplotlib inline
from matplotlib import style
style.use('fivethirtyeight')
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import sqlalchemy
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine, func
```

Reflect Tables into SQLAlchemy ORM

In [67]:

```
engine = create_engine("sqlite:///Resources/hawaii.sqlite")
```

In [68]:

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

In [69]:

```
# We can view all of the classes that automap found
Base.classes.keys()
```

Out[69]:

```
['measurement', 'station']
```

In [73]:
```
# `engine.execute` to select and show the first 15 rows from the measurement.
result = engine.execute("select * from measurement").fetchall()
print(result[:15])
```

```
[(1, 'USC00519397', '2010-01-01', 0.08, 65.0), (2, 'USC00519397', '2010-01-02',
0.0, 63.0), (3, 'USC00519397', '2010-01-03', 0.0, 74.0), (4, 'USC00519397', '20
10-01-04', 0.0, 76.0), (5, 'USC00519397', '2010-01-06', None, 73.0), (6, 'USC00
519397', '2010-01-07', 0.06, 70.0), (7, 'USC00519397', '2010-01-08', 0.0, 64.
0), (8, 'USC00519397', '2010-01-09', 0.0, 68.0), (9, 'USC00519397', '2010-01-1
0', 0.0, 73.0), (10, 'USC00519397', '2010-01-11', 0.01, 64.0), (11, 'USC0051939
7', '2010-01-12', 0.0, 61.0), (12, 'USC00519397', '2010-01-14', 0.0, 66.0), (1
3, 'USC00519397', '2010-01-15', 0.0, 65.0), (14, 'USC00519397', '2010-01-16',
0.0, 68.0), (15, 'USC00519397', '2010-01-17', 0.0, 64.0)]
```

In [74]:
```
# `engine.execute` to select and show the first 15 rows from the measurement.
result = engine.execute("select *  from station").fetchall()
print(result[:15])
```

```
[(1, 'USC00519397', 'WAIKIKI 717.2, HI US', 21.2716, -157.8168, 3.0), (2, 'USC0
0513117', 'KANEOHE 838.1, HI US', 21.4234, -157.8015, 14.6), (3, 'USC00514830',
'KUALOA RANCH HEADQUARTERS 886.9, HI US', 21.5213, -157.8374, 7.0), (4, 'USC005
17948', 'PEARL CITY, HI US', 21.3934, -157.9751, 11.9), (5, 'USC00518838', 'UPP
ER WAHIAWA 874.3, HI US', 21.4992, -158.0111, 306.6), (6, 'USC00519523', 'WAIMA
NALO EXPERIMENTAL FARM, HI US', 21.33556, -157.71139, 19.5), (7, 'USC00519281',
'WAIHEE 837.5, HI US', 21.45167, -157.84888999999998, 32.9), (8, 'USC00511918',
'HONOLULU OBSERVATORY 702.2, HI US', 21.3152, -157.9992, 0.9), (9, 'USC0051612
8', 'MANOA LYON ARBO 785.2, HI US', 21.3331, -157.8025, 152.4)]
```

In [75]:
```
# Reflect Database into ORM class
Base = automap_base()
Base.prepare(engine, reflect=True)
Measurement = Base.classes.measurement
Station = Base.classes.station
```

In [80]:
```
# Create our session (connection) to the DB
session = Session(engine)
```

In [85]:
```
first_row = session.query(Measurement).first()
first_row.__dict__
```

Out[85]:
```
{'_sa_instance_state': <sqlalchemy.orm.state.InstanceState at 0x1869427fb00>,
 'date': '2010-01-01',
 'id': 1,
 'prcp': 0.08,
 'station': 'USC00519397',
 'tobs': 65.0}
```

In [86]:
```
# Find the # of Measurement from the USC
usc = session.query(Measurement).filter(Measurement.station == 'USC00519397').cou
print("There are {} station from the USC00519397".format(usc))
```

```
There are 2724 station from the USC00519397
```

In [88]:

```
# Query Measurement for id`, `station`, date, prcp, tobs and `data` and save
id=[]
station=[]
date=[]
prcp=[]
tobs=[]
data=[]
for row in session.query(Measurement.id, Measurement.station, Measurement.date, Me
    id.append(row[0])
    station.append(row[1])
    date.append(row[2])
    prcp.append(row[3])
    tobs.append(row[4])
```

In [89]:

```
engine.execute('SELECT * FROM measurement LIMIT 5').fetchall()
```

Out[89]:

```
[(1, 'USC00519397', '2010-01-01', 0.08, 65.0),
 (2, 'USC00519397', '2010-01-02', 0.0, 63.0),
 (3, 'USC00519397', '2010-01-03', 0.0, 74.0),
 (4, 'USC00519397', '2010-01-04', 0.0, 76.0),
 (5, 'USC00519397', '2010-01-06', None, 73.0)]
```

In [90]:

```
# Save references to each table
Measurement = Base.classes.measurement
Station = Base.classes.station
```

In [91]:

```
# Create our session (link) from Python to the DB
session = Session(engine)
```

Exploratory Climate Analysis

In [92]:

```
# Earliest Date
session.query(Measurement.date).order_by(Measurement.date).first().date
date_start = session.query(Measurement.date).order_by(Measurement.date).first().d
date_start
```

Out[92]:

```
'2010-01-01'
```

In [93]:

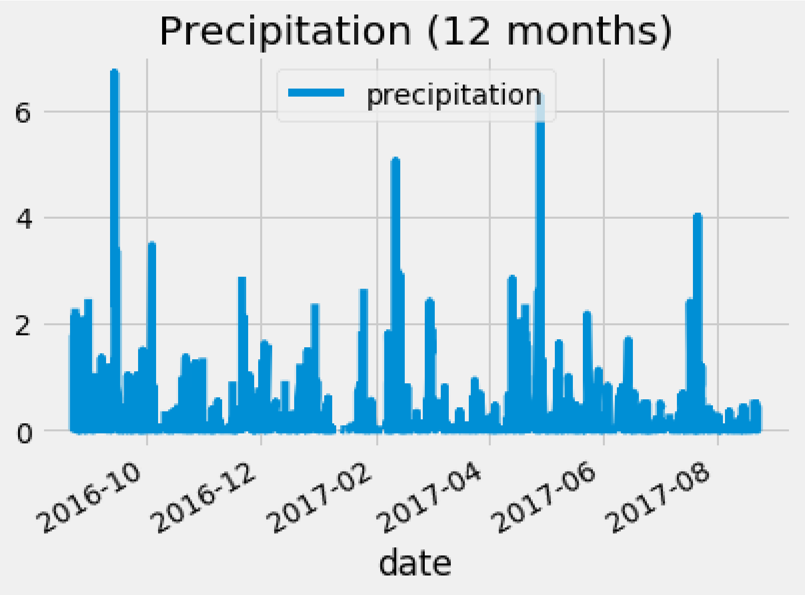
```
# Latest Date
latest_date = session.query(Measurement.date).order_by(Measurement.date.desc()).f
date_end = latest_date = session.query(Measurement.date).order_by(Measurement.dat
latest_date
```

Out[93]:

```
'2017-08-23'
```

In [98]:

```
# Design a query to retrieve the last 12 months of precipitation data and plot th
max_date = session.query(Measurement.date).order_by(Measurement.date.desc()).firs
max_date = max_date[0]
# Calculate the date 1 year ago from the last data point in the database
one_year_ago = dt.datetime.strptime(max_date, "%Y-%m-%d") - dt.timedelta(days=366
# Perform a query to retrieve the data and precipitation scores
query = session.query(Measurement.date, Measurement.prcp).filter(Measurement.date
# Save the query results as a Pandas DataFrame and set the index to the date colu
precipitation_df = pd.DataFrame(query,columns=['date', 'precipitation'])
precipitation_df['date'] = pd.to_datetime(precipitation_df['date'], format='%Y-%m
precipitation_df.set_index('date', inplace=True)
# Sort the dataframe by date
precipitation_df = precipitation_df.sort_values(by='date',ascending=True)
# Use Pandas Plotting with Matplotlib to plot the data
precipitation_df .plot(title="Precipitation (12 months)")
plt.legend(loc='upper center')
plt.show()
```



In [99]:
```
# Use Pandas to calcualte the summary statistics for the precipitation data
precipitation_df.describe()
```

Out[99]:

	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

In [100]:
```
# Design a query to show how many stations are available in this dataset?
available_stations = session.query(Measurement.station).distinct().count()
print(f"Stations Available: {available_stations} ")
```

```
Stations Available: 9
```

In [124]:
```
# What are the most active stations? (i.e. what stations have the most rows)?
# List the stations and the counts in descending order.
active_stations = session.query(Measurement.station,
                                func.count(Measurement.station)).group_by(Measure
print(f"Most active stations")
active_stations
```

```
Most active stations
```

Out[124]:
```
[('USC00519281', 2772),
 ('USC00519397', 2724),
 ('USC00513117', 2709),
 ('USC00519523', 2669),
 ('USC00516128', 2612),
 ('USC00514830', 2202),
 ('USC00511918', 1979),
 ('USC00517948', 1372),
 ('USC00518838', 511)]
```

In [123]:
```
# Finding most active station
most_active_station=active_stations[0][0]
print(f"Most active station: {most_active_station}")
```

```
Most active station: USC00519281
```

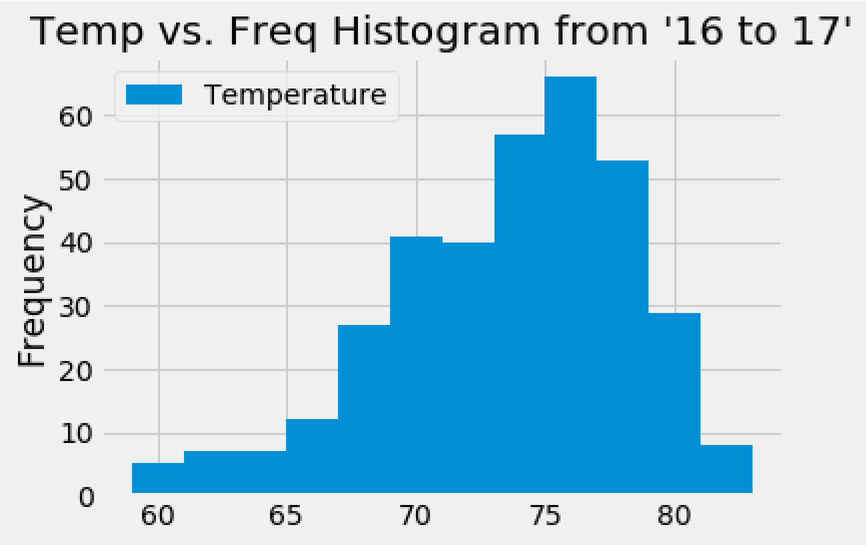
In [122]:

```
# Using the station id from the previous query, calculate the lowest temperature r
# highest temperature recorded, and average temperature most active station?
most_active_temps = session.query(func.min(Measurement.tobs), func.max(Measurement
                                  func.avg(Measurement.tobs)).filter(Measurement.
print(f"Most active station temperatures")
print(f"Low: {most_active_temps[0][0]} High: {most_active_temps[0][1]} Avg: {round
```

```
Most active station temperatures
Low: 54.0 High: 85.0 Avg: 71.7
```

In [120]:

```
# Choose the station with the highest number of temperature observations.
# Query the last 12 months of temperature observation data for this station and pl
most_temps_station = session.query(Measurement.station, func.count(Measurement.tob
most_temps_station= most_temps_station[0]
temperature_observations = session.query( Measurement.tobs).filter(Measurement.dat
temperature_observations = pd.DataFrame(temperature_observations, columns=['Temper
temperature_observations.plot.hist(bins=12, title="Temp vs. Freq Histogram from '1
plt.tight_layout()
plt.show()
```



In [107]:

```
# This function called `calc_temps` will accept start date and end date in the fo
# and return the minimum, average, and maximum temperatures for that range of dat
def calc_temps(start_date, end_date):
    """TMIN, TAVG, and TMAX for a list of dates.

    Args:
        start_date (string): A date string in the format %Y-%m-%d
        end_date (string): A date string in the format %Y-%m-%d

    Returns:
        TMIN, TAVE, and TMAX
    """

    return session.query(func.min(Measurement.tobs), func.avg(Measurement.tobs), 
        filter(Measurement.date >= start_date).filter(Measurement.date <= end_dat

# function usage example
print(calc_temps('2017-01-01', '2018-01-01'))
```

```
[(58.0, 74.14387974230493, 87.0)]
```

Trip Climate Analysis

In [108]:

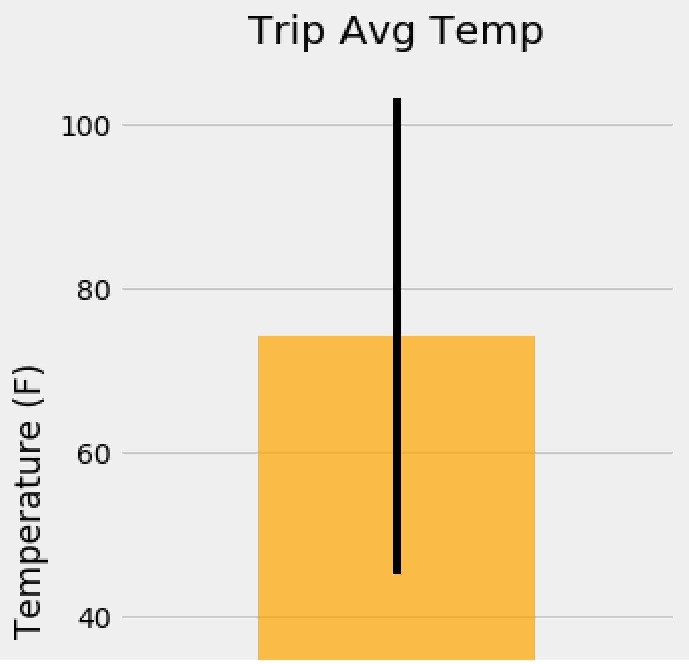
```
# Use your previous function `calc_temps` to calculate the tmin, tavg, and tmax
# for your trip using the previous year's data for those same dates.
trip = calc_temps('2017-01-01', '2018-01-01')
trip
```

Out[108]:

```
[(58.0, 74.14387974230493, 87.0)]
```

In [118]:

```
trip_temp_df = pd.DataFrame(trip, columns=['tmin', 'tavg', 'tmax'])
# Plot the results from your previous query as a bar chart.
# Use "Trip Avg Temp" as your Title
# Use the average temperature for the y value
# Use the peak-to-peak (tmax-tmin) value as the y error bar (yerr)
trip_temp_df.plot.bar(y='tavg', yerr=(trip_temp_df['tmax'] - trip_temp_df['tmin']
plt.xticks(np.arange(1, 1.0))
plt.ylabel("Temperature (F)")
plt.tight_layout()
plt.gca().legend_.remove()
plt.show()
```



In [110]:

```
# Calculate the rainfall per weather station for your trip dates using the previo
# Sort this in descending order by precipitation amount and list the station, nam
def precipitation(start_date, end_date):

        # Docstring for the function `calc_temps`
    """Precipitation information per weather station

    Args:
        start_date (string): A date string in the format %Y-%m-%d
        end_date (string): A date string in the format %Y-%m-%d

    Returns:
        A list of tuples containing precipitation amount, station, name, latitude
    """

    sel = [Measurement.station,
           Station.name,
           Station.latitude,
           Station.longitude,
           Station.elevation,
           Measurement.prcp]

    return session.query(*sel).\
            filter(Measurement.station == Station.station).filter(Measurement.dat

print(precipitation('2017-01-01', '2018-01-01'))
```

```
[('USC00516128', 'MANOA LYON ARBO 785.2, HI US', 21.3331, -157.8025, 152.4, 0.4
5), ('USC00519523', 'WAIMANALO EXPERIMENTAL FARM, HI US', 21.33556, -157.71139,
19.5, 0.08), ('USC00519281', 'WAIHEE 837.5, HI US', 21.45167, -157.848889999999
98, 32.9, 0.06), ('USC00513117', 'KANEOHE 838.1, HI US', 21.4234, -157.8015, 1
4.6, 0.0), ('USC00514830', 'KUALOA RANCH HEADQUARTERS 886.9, HI US', 21.5213, -
157.8374, 7.0, 0.0), ('USC00519397', 'WAIKIKI 717.2, HI US', 21.2716, -157.816
8, 3.0, 0.0), ('USC00517948', 'PEARL CITY, HI US', 21.3934, -157.9751, 11.9, No
ne)]
```

In [111]:

```
#get average rainfall for each weather station for the last year
import datetime
yearly_rainfall = session.query(Station.station, Station.name, Station.latitude, 
                                Station.elevation, func.avg(Measurement.prcp)).\
    filter(Measurement.station == Station.station).\
    filter(func.strftime("%Y-%m-%d", Measurement.date) >= datetime.date(2016, 8, 
    order_by(func.avg(Measurement.prcp).desc()).all()

#load into a dataframe
yearly_rainfall_df = pd.DataFrame(yearly_rainfall, columns = ['Station', 'Name', 
                                                              'Elevation', 'Avg.

yearly_rainfall_df
```

Out[111]:

	Station	Name	Latitude	Longitude	Elevation	Avg. Precipitation (in.)
0	USC00516128	MANOA LYON ARBO 785.2, HI US	21.33310	-157.80250	152.4	0.450640
1	USC00519281	WAIHEE 837.5, HI US	21.45167	-157.84889	32.9	0.198949
2	USC00513117	KANEOHE 838.1, HI US	21.42340	-157.80150	14.6	0.141429
3	USC00514830	KUALOA RANCH HEADQUARTERS 886.9, HI US	21.52130	-157.83740	7.0	0.125434
4	USC00519523	WAIMANALO EXPERIMENTAL FARM, HI US	21.33556	-157.71139	19.5	0.121051
5	USC00517948	PEARL CITY, HI US	21.39340	-157.97510	11.9	0.076500
6	USC00519397	WAIKIKI 717.2, HI US	21.27160	-157.81680	3.0	0.044819

Optional Challenge Assignment

Chosen trip days from 2018-01-01 to 2018-01-05

In [112]:

```
# Create a query that will calculate the daily normals
# (i.e. the averages for tmin, tmax, and tavg for all historic data matching a spe

def daily_normals(date):
    """Daily Normals.

    Args:
        date (str): A date string in the format '%m-%d'

    Returns:
        A list of tuples containing the daily normals, tmin, tavg, and tmax

    """

    sel = [func.min(Measurement.tobs), func.avg(Measurement.tobs), func.max(Measu
    return session.query(*sel).filter(func.strftime("%m-%d", Measurement.date) ==

daily_normals("01-01")
```

Out[112]:

```
[(62.0, 69.15384615384616, 77.0)]
```

In [113]:

```
#set the start and end date for the trip
startDate = "2018-01-01"
endDate = "2018-01-05"

#calculate trip length2018-01-01
startNum = int(startDate[-2:])
endNum = int(endDate[-2:])
tripLength = endNum - startNum + 1

#start date as datetime object
startDate = dt.datetime.strptime(startDate, '%Y-%m-%d')
#list dates (MM-DD) of trip
dateList = [dt.datetime.strftime(startDate + dt.timedelta(days = x), '%m-%d')
            for x in range(0, tripLength)]

#calculate normals for each date
tripNormals = [daily_normals(date) for date in dateList]

tripNormals
```

Out[113]:

```
[[(62.0, 69.15384615384616, 77.0)],
 [(60.0, 69.39622641509433, 77.0)],
 [(62.0, 68.9090909090909, 77.0)],
 [(58.0, 70.0, 76.0)],
 [(56.0, 67.96428571428571, 76.0)]]
```

In [114]:

```
#extract normals into a list of lists
tripNormals = [np.array(normal[0]) for normal in tripNormals]

#convert normals list into a data frame
normalsTable = pd.DataFrame(tripNormals)
#add date column
normalsTable["Date"] = dateList
#set index and rename columns
normalsTable = normalsTable.set_index("Date")
normalsTable = normalsTable.rename(columns={0: "Low Temp", 1: "Avg Temp", 2: "Hig

normalsTable
```

Out[114]:

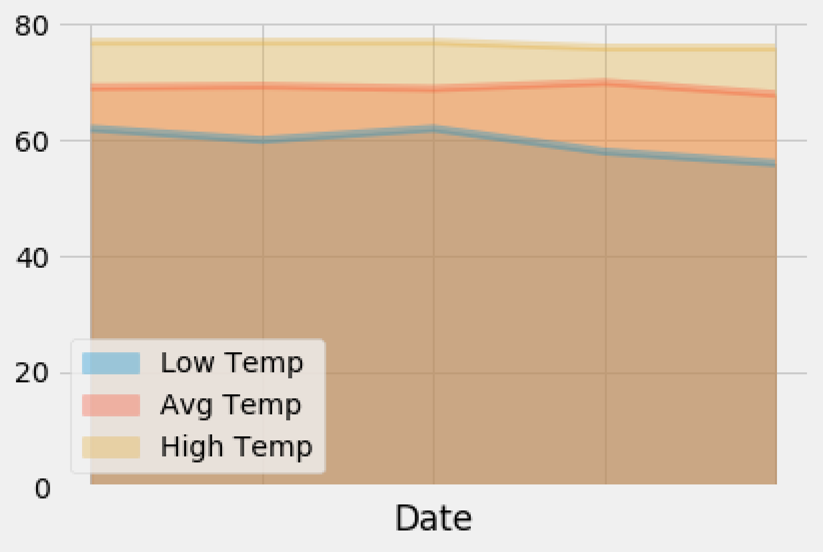
	Low Temp	Avg Temp	High Temp
Date			
01-01	62.0	69.153846	77.0
01-02	60.0	69.396226	77.0
01-03	62.0	68.909091	77.0
01-04	58.0	70.000000	76.0
01-05	56.0	67.964286	76.0

In [115]:

```
#plot with pandas
normalsTable.plot.area(stacked=False, alpha=.333)
```

Out[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18694218208>
```



In [ ]:

In [ ]: