Week 6 HW - What's the Weather Like? using APIs and JSON

by: Michael Suomi 6/23/2018

- Observation 1: Temperature does tend to get cooler when you move farther away from the equator - you notice higher relative temperatures in northern latitudes relative to the southern latitudes because it is currently summer in northern hemisphere and winter in summer hemisphere.
- Observation 2: There are not any real discernable trends for Humidity Cloudiness or Wind Speed as you move away from the equator (although there is a slight uptick in wind speed around the -40 to -50 latitudes).
- Observation 3: CiityPy stores many more cities in Europe than any other locations around the world - when I first randomly chose from their dictionary of cities and create a google maps plot they were most dense in Europe.

WeatherPy Instructions:

In this example, you'll be creating a Python script to visualize the weather of 500+ cities across the world of varying distance from the equator. To accomplish this, you'll be utilizing a simple Python library, the OpenWeatherMap API, and a little common sense to create a representative model of weather across world cities.

Your objective is to build a series of scatter plots to showcase the following relationships:

- Temperature (F) vs. Latitude
- Humidity (%) vs. Latitude
- Cloudiness (%) vs. Latitude
- Wind Speed (mph) vs. Latitude

Your final notebook must:

- Randomly select at least 500 unique (non-repeat) cities based on latitude and longitude.
- Perform a weather check on each of the cities using a series of successive API calls.
- Include a print log of each city as it's being processed with the city number and city name.
- Save both a CSV of all data retrieved and png images for each scatter plot.

In [1]:

```
import os
import pandas as pd
import numpy as np
import requests
import json
import matplotlib.pyplot as plt
from citipy import citipy
import random
import gmaps
gmaps.configure(api_key=os.environ.get('googlemaps_api_key'))
openweathermap_api_key = os.environ.get('openweathermap_api_key')
data_output_folder = "data_output\\"
```

Randomly select 500 unique cities based on latitude and longitude.

In [2]:
```
###in order to get better city distribution, make latitude bands of 5 degrees and
###choose 20 random cities from each of those latitude bands
random_lat_longs3 = []

for lat_search in list(range(-90,90,5)):
    lat_range_cities = [x for x in list(citipy.WORLD_CITIES_DICT.keys())
                        if (lat_search+5) > x[0] and lat_search >= x[0]]

    #determine number of random cities to get out of the range:
    #if number of cities is less than 20 then just select all cities
    #otherwise select 20 random cities and add to city list
    if len(lat_range_cities) > 20:
        num_random_cities = 20
        for num_cities in list(range(num_random_cities)):
            random_lat_longs3.append(random.choice(lat_range_cities))
    else:
        for city in lat_range_cities:
            random_lat_longs3.append(city)

random_city_citipy_encodings3 = [citipy.WORLD_CITIES_DICT[x] for x in random_lat_
random_city_names3 = [x.city_name for x in random_city_citipy_encodings3]
random_countries3 = [x.country_code for x in random_city_citipy_encodings3]
```

In [3]:
```
#create dataframe of the random cities
df_random_cities3 = pd.DataFrame({'City': random_city_names3,
                                  'Country': random_countries3,
                                  'Lat, Long': random_lat_longs3,
                                  'Lat': [x[0] for x in random_lat_longs3],
                                  'Lon': [x[1] for x in random_lat_longs3]})
df_random_cities3 = df_random_cities3[['City', 'Country', 'Lat, Long', 'Lat', 'Lo

#due to random chance there may be duplicate cities (especially if the number of
#cities in the lat, long range is short) so drop the duplicate cities
df_random_cities3.drop_duplicates(inplace=True)
#google maps errors out when reference the data frame columns if the index is not
df_random_cities3.reset_index(drop=True, inplace=True)
print(df_random_cities3.shape)
df_random_cities3.head()
```

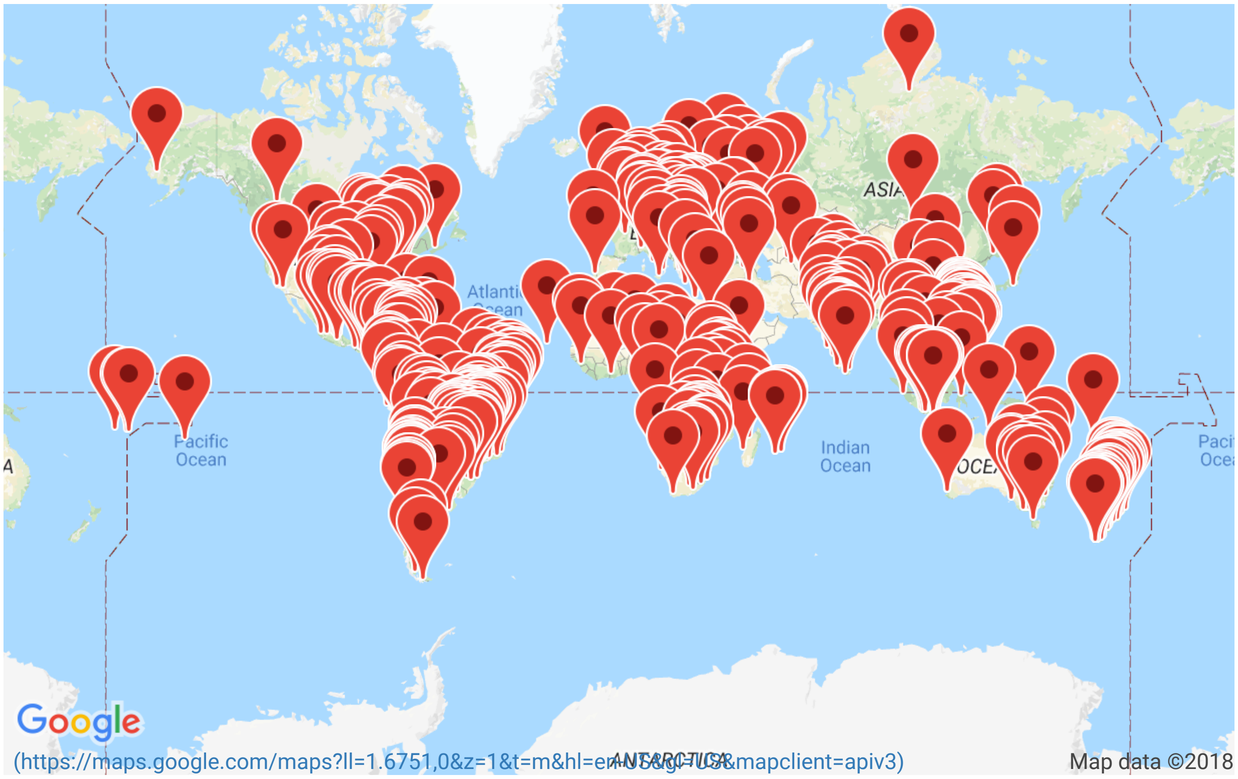
```
(519, 5)
```

Out[3]:

	City	Country	Lat, Long	Lat	Lon
0	rio gallegos	ar	(-51.622613, -69.218127)	-51.622613	-69.218127
1	ushuaia	ar	(-54.8, -68.3)	-54.800000	-68.300000
2	punta arenas	cl	(-53.15, -70.916667)	-53.150000	-70.916667
3	invercargill	nz	(-46.4, 168.35)	-46.400000	168.350000
4	waitati	nz	(-45.75, 170.566667)	-45.750000	170.566667

In [10]:

```
#create a google map of the locations to see how spread out the random cities rea
#make sure it really covers various lat,longs
fig3 = gmaps.figure()

markers = gmaps.marker_layer(locations=df_random_cities3['Lat, Long'],
                             info_box_content=df_random_cities3['Country']) #clic

fig3.add_layer(markers)
fig3
```



Calling 5-Day Forecast Weather API

In [5]:

```
#create a new dataframe and then overwrite the values
df_random_cities_forecast3 = df_random_cities3
# #if need to overwrite dataframe values when re-run df
# df_random_cities_forecast3[['OWM City #', 'OWM City Name', 'Temp (F)', 'Humidit
#                             'Cloudiness (%)', 'Wind Speed (mph)']] = ''
```

In [6]:

```
## in order to get better picture of the weather data than just the exact moment
## use the 5 day forecast API (we can't use the historical data without paying fo
print('Beginning Data Retrieval \n-------------------------------')

for index, city_row in df_random_cities_forecast3.iterrows():
    print(f"Processing Record {index+1} of {len(df_random_cities_forecast3)}")
    #api.openweathermap.org/data/2.5/forecast?lat=35&lon=139
    #http://samples.openweathermap.org/data/2.5/forecast?lat=35&lon=139&appid=b69

    base_url = 'http://api.openweathermap.org/data/2.5/forecast?'
    parameters = {'lat': city_row['Lat'],
                  'lon': city_row['Lon'],
                  'appid': openweathermap_api_key,
                  'units': 'imperial'}

    #print url looking at, but hide the api key by replacing with #'s
    print((requests.get(base_url, parameters).url).replace(openweathermap_api_key
    #get json
    city_weather_data = requests.get(base_url, parameters).json()
    #print(city_weather_data) #temporary check

    ###for some reason city_row['column'] = did not work because its a copy of sl
    ### so use .loc on dataframe instead
    ###get the city id and city name for the processing print records and save in
    df_random_cities_forecast3.loc[index, 'OWM City #'] = city_weather_data.get('
    df_random_cities_forecast3.loc[index, 'OWM City Name'] = city_weather_data.ge
    print(f"City ID = {city_weather_data.get('city').get('id')}, City Name = {cit

    ###for all the forecast data, create a list that collects the weather values
    ###then takes max temp of the forecasts, and takes the mean of the other fore
    df_random_cities_forecast3.loc[index, 'Temp (F)'] = max(
                [x.get('main').get('temp') for x in city_weather_data.get('list')
    df_random_cities_forecast3.loc[index, 'Humidity (%)'] = np.mean(
                [x.get('main').get('humidity') for x in city_weather_data.get('li
    df_random_cities_forecast3.loc[index, 'Cloudiness (%)'] = np.mean(
                [x.get('clouds').get('all') for x in city_weather_data.get('list'
    df_random_cities_forecast3.loc[index, 'Wind Speed (mph)'] = np.mean(
                [x.get('wind').get('speed') for x in city_weather_data.get('list'

print('-------------------------------\n Data Retrieval Complete \n--------------
```

```
City ID = 1785964, City Name = Yudong
Processing Record 342 of 519
http://api.openweathermap.org/data/2.5/forecast?lat=-29.328164&lon=31.289537&
appid=################################&units=imperial (http://api.openweather
map.org/data/2.5/forecast?lat=-29.328164&lon=31.289537&appid=################
################&units=imperial)
City ID = 952734, City Name = Stanger
Processing Record 343 of 519
http://api.openweathermap.org/data/2.5/forecast?lat=1.723624&lon=-76.134028&a
ppid=################################&units=imperial (http://api.openweatherm
ap.org/data/2.5/forecast?lat=1.723624&lon=-76.134028&appid=##################
##############&units=imperial)
City ID = 3673275, City Name = Palestina
Processing Record 344 of 519
http://api.openweathermap.org/data/2.5/forecast?lat=10.643455&lon=123.083603&
appid=################################&units=imperial (http://api.openweather
map.org/data/2.5/forecast?lat=10.643455&lon=123.083603&appid=################
map.org/data/2.5/forecast?lat=10.643455&lon=123.083603&appid=#################
################&units=imperial)
City ID = 1731564, City Name = Alegria
Processing Record 345 of 519
```

In [7]:

```
df_random_cities_forecast3.to_csv(f"{data_output_folder}Random City Weather Data.
df_random_cities_forecast3.head()
```

Out[7]:

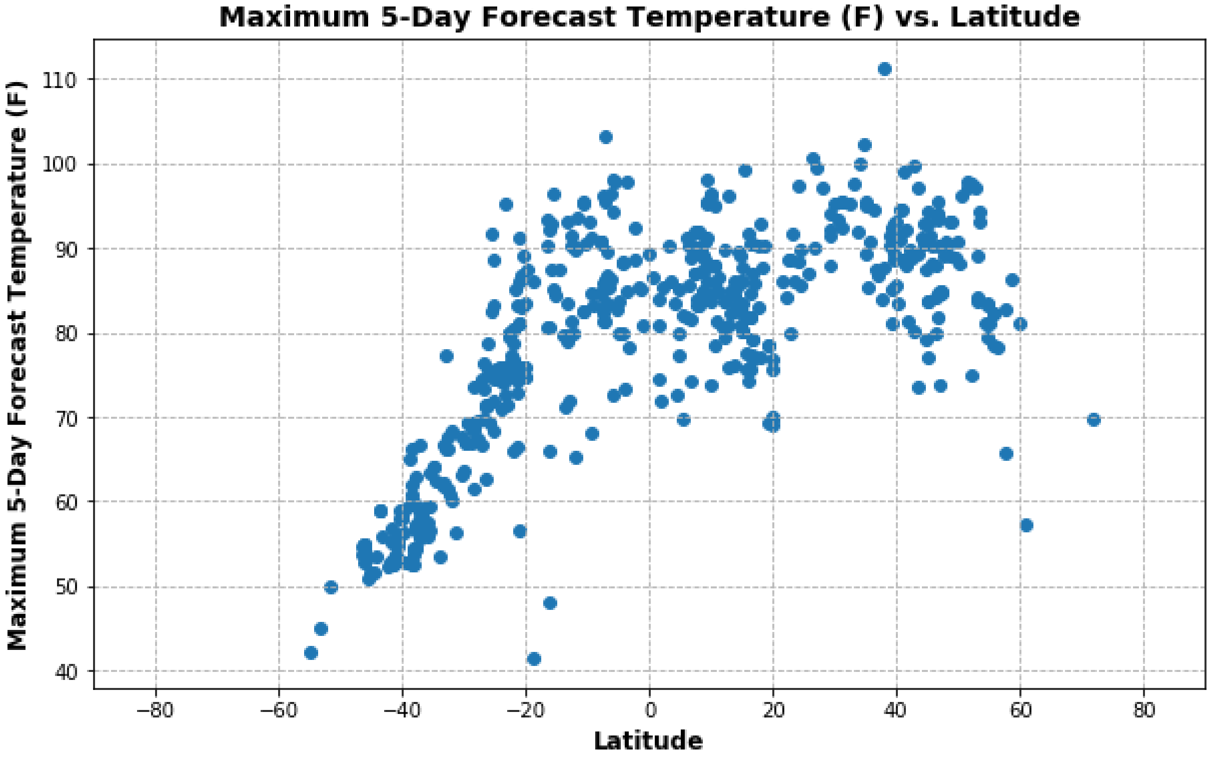
	City	Country	Lat, Long	Lat	Lon	OWM City #	OWM City Name	Temp (F)	Humidity (%)
0	rio gallegos	ar	(-51.622613, -69.218127)	-51.622613	-69.218127	3838859.0	Rio Gallegos	49.91	88.875
1	ushuaia	ar	(-54.8, -68.3)	-54.800000	-68.300000	3833367.0	Ushuaia	42.13	98.225
2	punta arenas	cl	(-53.15, -70.916667)	-53.150000	-70.916667	3874787.0	Punta Arenas	45.09	97.475
3	invercargill	nz	(-46.4, 168.35)	-46.400000	168.350000	2189529.0	Invercargill	54.70	82.975
4	waitati	nz	(-45.75, 170.566667)	-45.750000	170.566667	2179825.0	Waitati	53.30	82.600

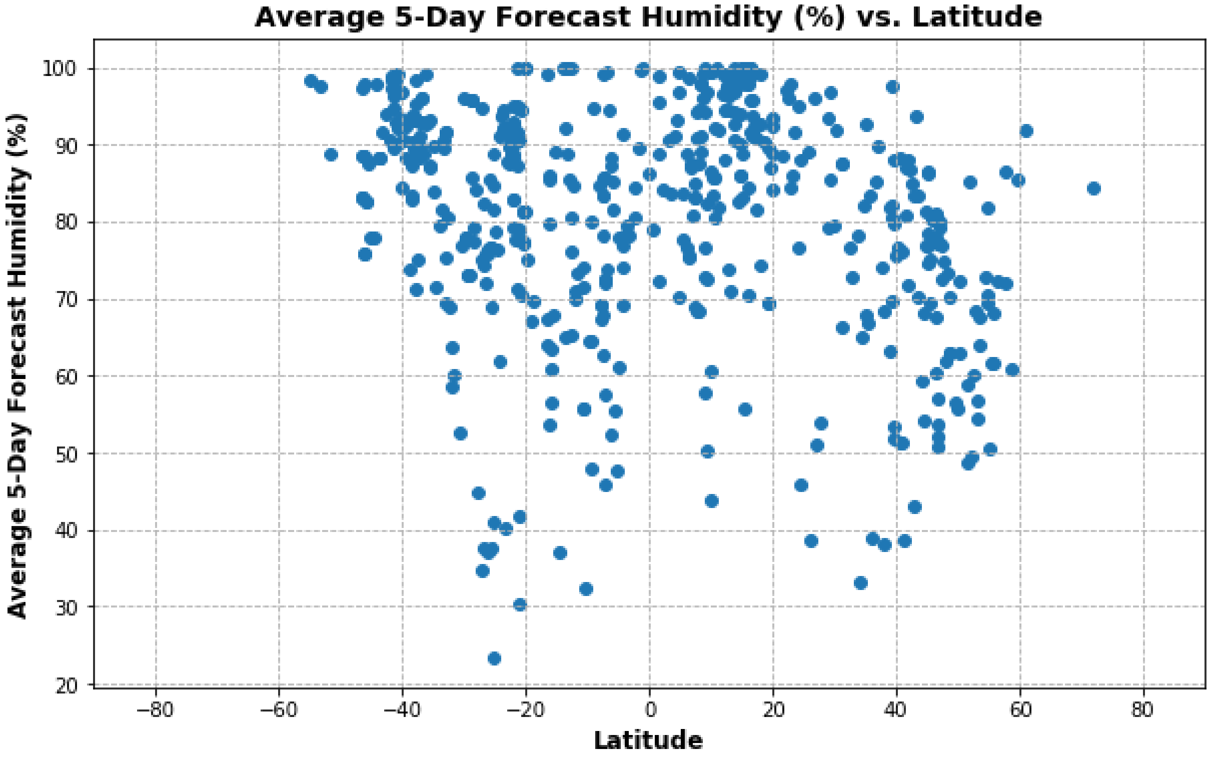
Plotting Weather Results

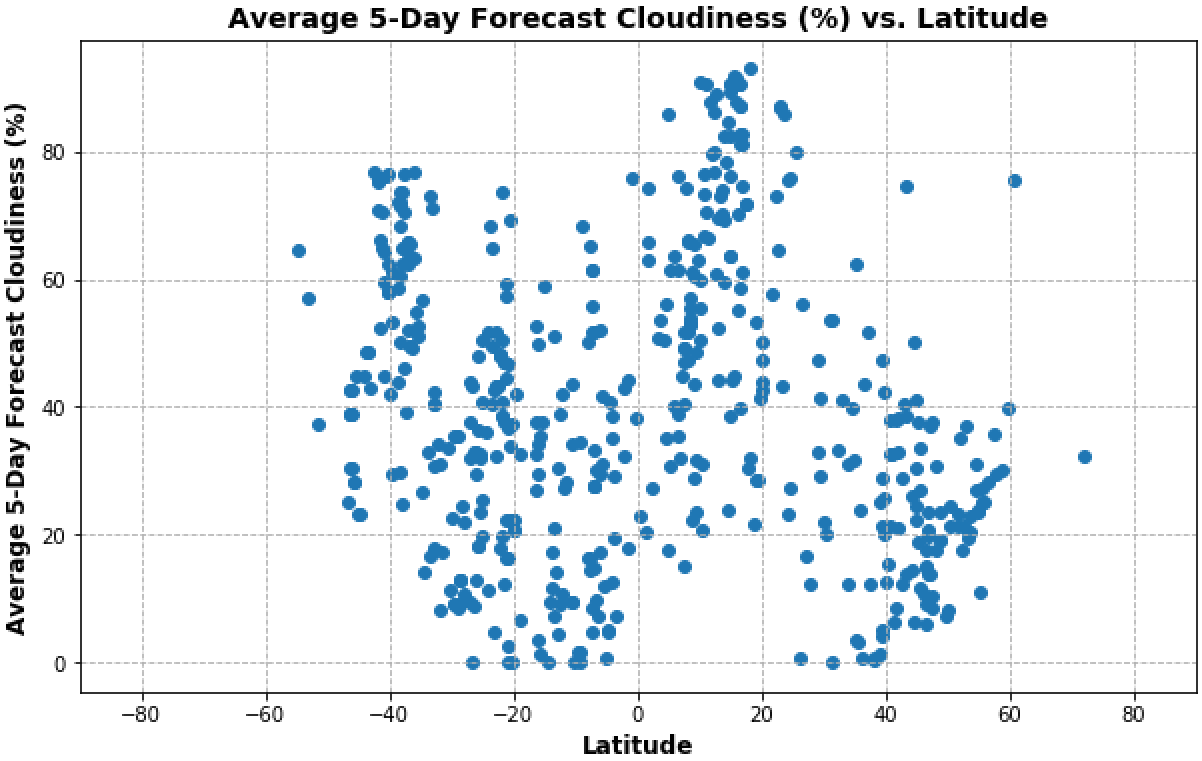
In [8]:

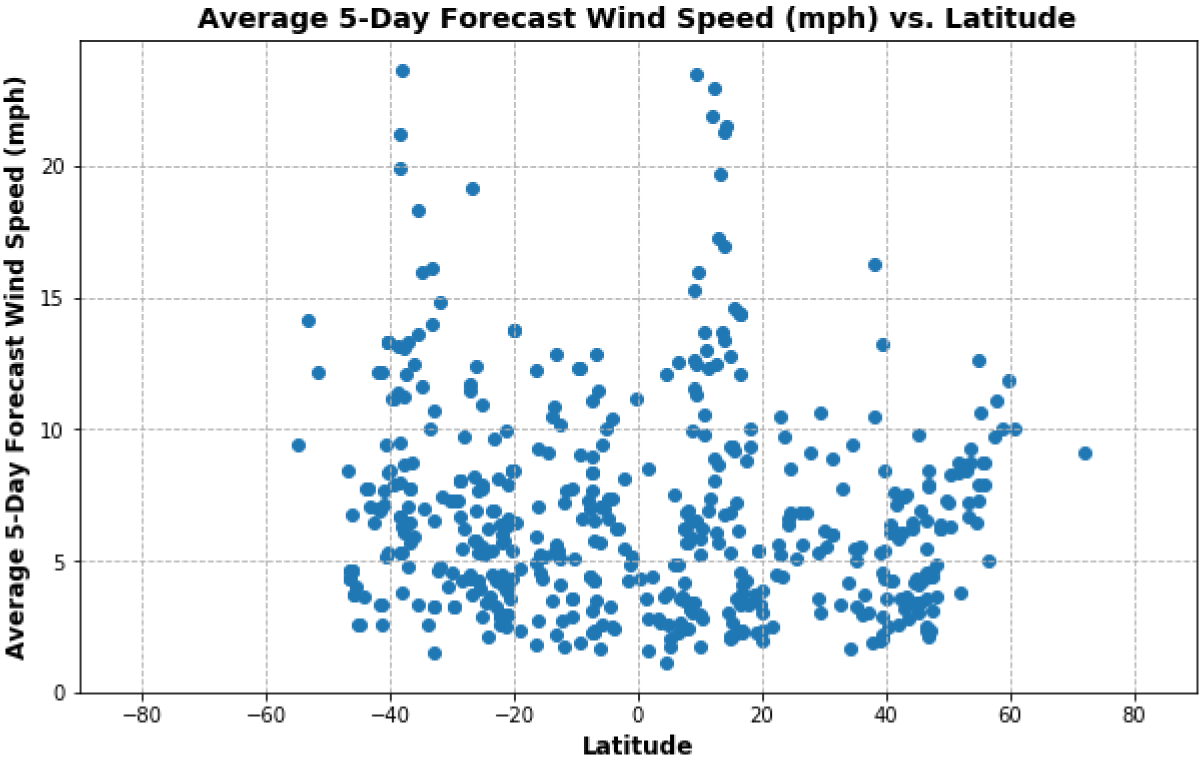
```
#y_data is the df_random_cities['column'] and x_data is df_random_cities['column'
#y_label is the name you want for y and x_label for x - labels default to name of
def scatter_plot(x_data, y_data, x_label=None, y_label=None):
    if x_label==None:
        x_label=x_data.name
    if y_label==None:
        y_label=y_data.name

    plt.figure(figsize=(10,6), )

    plt.scatter(x_data, y_data)

    plt.xlim(-90,90)
#   plt.ylim(0)

    plt.xlabel(x_label, size=12, fontweight='semibold')
    plt.ylabel(y_label, size=12, fontweight='semibold')
    plt.grid(linestyle='--')

    plt.title(f'{y_label} vs. {x_label}', size=14, fontweight='bold')
    plt.savefig(f"{data_output_folder}{y_label} vs. {x_label}.png")
    plt.show()
```

In [9]:
```
scatter_plot(df_random_cities_forecast3['Lat'], df_random_cities_forecast3['Temp
             'Latitude', 'Maximum 5-Day Forecast Temperature (F)')
scatter_plot(df_random_cities_forecast3['Lat'], df_random_cities_forecast3['Humid
             'Latitude', 'Average 5-Day Forecast Humidity (%)')
scatter_plot(df_random_cities_forecast3['Lat'], df_random_cities_forecast3['Cloud
             'Latitude', 'Average 5-Day Forecast Cloudiness (%)')
scatter_plot(df_random_cities_forecast3['Lat'], df_random_cities_forecast3['Wind
             'Latitude', 'Average 5-Day Forecast Wind Speed (mph)')
```


Maximum 5-Day Forecast Temperature (F) vs. Latitude


Average 5-Day Forecast Humidity (%) vs. Latitude


Average 5-Day Forecast Cloudiness (%) vs. Latitude
Average 5-Day Forecast Cloudiness (%)
0
20
40
60
80
−80
−60
−40
−20
0
20
40
60
80
Latitude


Average 5-Day Forecast Wind Speed (mph) vs. Latitude
Average 5-Day Forecast Wind Speed (mph)
0
5
10
15
20
−80
−60
−40
−20
0
20
40
60
80
Latitude