WeatherPy

Observations:

- Temperature has clear corelation with lattitude.
- The weather becomes slightly warmer as one approches equator. The sourthern hemisphere tends to be more warmer than northan with max temp lying between 40-100
- Cloudiness and lattitude does not corelate. However it is interesting to see strong bond of cities near 0, 78 & 100% cloudiness
- Wind speed tends to generally be betweeen 0 and 15 mph regardless of latitude
- There is no strong relationship between humidity and lattitude. However, there is slightly large cluster of cities with high humidity in northern hemisphere(above 60% humidity)

Note

- Instructions have been included for each segment. You do not have to follow them exactly, but they are included to help you think through the steps.

In [11]:
```
# Importing required modules

import json
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from datetime import date
from scipy.stats import linregress

# Imporing API key
import sys
sys.path.append('..')
from api_keys import weather_api_key

# Incorporated citipy to determine city based on latitude and longitude
from citipy import citipy
```

Generate Cities List

In [12]:

```
# Created two empty lists to store lat-langs & cities list
lat_langs = []
cities = []

# Create a set of random lat and lang combinations
lats = np.random.uniform(low=-90.000, high=90.000, size=1600)
langs = np.random.uniform(low=-180.000, high=180.000, size=1600)

# Zip lat & lang
lat_langs = zip(lats, langs)

# Iterate through each lat-lang combination
for lat_lang in lat_langs:

    #Get the city name by using citipy module
    city = citipy.nearest_city(lat_lang[0],lat_lang[1]).city_name

    # Adding a city to cities list, only if its not already there
    if city not in cities:
        cities.append(city)

# Print the number of cities in cities list
len(cities)
```

Out[12]: 657

Perform API Calls

- Perform a weather check on each city using a series of successive API calls.
- Include a print log of each city as it'sbeing processed (with the city number and city name).

In [13]:

```
# Create a dataframe to store values.
# City column is populated with cities list
city_weather = pd.DataFrame({"City" : cities, "Lat":"", "Lng" : "", "Max Temp" : "", "Humidity" : "",
                             "Cloudiness" : "", "Wind Speed" : "", "Country" : "",  "Date" : ""})

# Display preview of DataFrame
city_weather.head()
```

Out[13]:

	City	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Country	Date
0	nizhniy odes								
1	vaini								
2	clyde river								
3	rikitea								
4	khatanga								

In [14]:

```
# Define base url with api key added & units declared as imperial to get the temp values in Farenheit
url = "http://api.openweathermap.org/data/2.5/weather?" + "&appid=" + weather_api_key + "&units=imperial"

# Initialised variables to track record set & number
recordset = 1
recordnum = 1

print("Beginning Data Retrieval\n-----------------------------")

# Iterate through each row of DataFrame
for index, row in city_weather.iterrows():
    # Store the city from DataFrame row to variable
    city = row['City']

    # Create a query url by appending city to base url
    query_url = url + "&q=" + city

    # Send get request to url & store the data in jason format
    response = requests.get(query_url).json()

    # Execute try block to store obtained values to DataFrame
    try:
        print(f"Processing Record {recordnum} of Set {recordset} | {response['name']}")
        city_weather.loc[index,'Lat'] = response['coord']['lat']
        city_weather.loc[index,'Lng'] = response['coord']['lon']
        city_weather.loc[index,'Max Temp'] = response['main']['temp_max']
        city_weather.loc[index,'Humidity'] = response['main']['humidity']
        city_weather.loc[index,'Cloudiness'] = response['clouds']['all']
        city_weather.loc[index,'Wind Speed'] = response['wind']['speed']
        city_weather.loc[index,'Country'] = response['sys']['country']
        city_weather.loc[index,'Date'] = response['dt']
        recordnum+=1
    # Execute the except block if city is not obtained from weather API
    except(KeyError, IndexError):
        print("City not found. Skipping...")

    # Once the record number completes 50, starts new record set
    if recordnum == 51:
        recordset+=1
        recordnum = 1
print("-----------------------------\nData Retrieval Complete\n-----------------------------")
```

```
Beginning Data Retrieval
-----------------------------
```

```
Processing Record 1 of Set 1 | Nizhniy Odes
Processing Record 2 of Set 1 | Vaini
Processing Record 3 of Set 1 | Clyde River
Processing Record 4 of Set 1 | Rikitea
Processing Record 5 of Set 1 | Khatanga
Processing Record 6 of Set 1 | Painesville
Processing Record 7 of Set 1 | Cloquet
Processing Record 8 of Set 1 | Butaritari
Processing Record 9 of Set 1 | Mabaruma
Processing Record 10 of Set 1 | Avarua
Processing Record 11 of Set 1 | Sioux Lookout
Processing Record 12 of Set 1 | Yumen
Processing Record 13 of Set 1 | Mataura
Processing Record 14 of Set 1 | Bredasdorp
Processing Record 15 of Set 1 | Ambilobe
Processing Record 16 of Set 1 | New Norfolk
Processing Record 17 of Set 1 | Severo-Kuril'sk
```

Convert Raw Data to DataFrame

- Export the city data into a .csv.
- Display the DataFrame

In [15]:
```
# Replace blank values in DataFrame with NaN
city_weather = city_weather.replace("", np.nan, regex=True)

# Checking missing values in DataFrame
city_weather.count()
```

Out[15]:
```
City          657
Lat           604
Lng           604
Max Temp      604
Humidity      604
Cloudiness    604
Wind Speed    604
Country       604
Date          604
dtype: int64
```

In [16]:
```
# Drop rows with blank(NaN) values
city_weather = city_weather.dropna(how='any')

# Checking if there are any missing values.
# All column displaying same values iondicate there are no missing values
city_weather.count()
```

Out[16]:
```
City          604
Lat           604
Lng           604
Max Temp      604
Humidity      604
Cloudiness    604
Wind Speed    604
Country       604
Date          604
dtype: int64
```

In [17]:
```
# Save a clean data to csv
city_weather.to_csv("city_weather_data.csv", encoding="utf-8", index=False)
```

Inspect the data and remove the cities where the humidity > 100%.

Skip this step if there are no cities that have humidity > 100%.

In [18]:
```
# Display statistical summary of DataFrame
city_weather.describe()
```

Out[18]:

	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Date
count	604.000000	604.000000	604.000000	604.000000	604.000000	604.000000	6.040000e+02
mean	20.083377	17.596722	58.625430	72.465232	51.230132	8.009404	1.603910e+09
std	33.240750	92.231498	23.294709	21.253827	40.868454	5.681043	9.926967e+01
min	-54.800000	-179.170000	-14.550000	6.000000	0.000000	0.340000	1.603910e+09
25%	-7.720000	-65.990000	42.447500	62.000000	1.000000	3.880000	1.603910e+09
50%	21.570000	26.185000	63.310000	78.000000	66.000000	6.375000	1.603910e+09
75%	48.182500	99.622500	78.015000	88.000000	90.000000	11.172500	1.603910e+09
max	78.220000	179.320000	98.600000	100.000000	100.000000	38.030000	1.603910e+09

Above table displays max Humidity, which is 100 hence skipping step to remove cities with humidity > 100

Plotting the Data

- Use proper labeling of the plots using plot titles (including date of analysis) and axes labels.
- Save the plotted figures as .pngs.

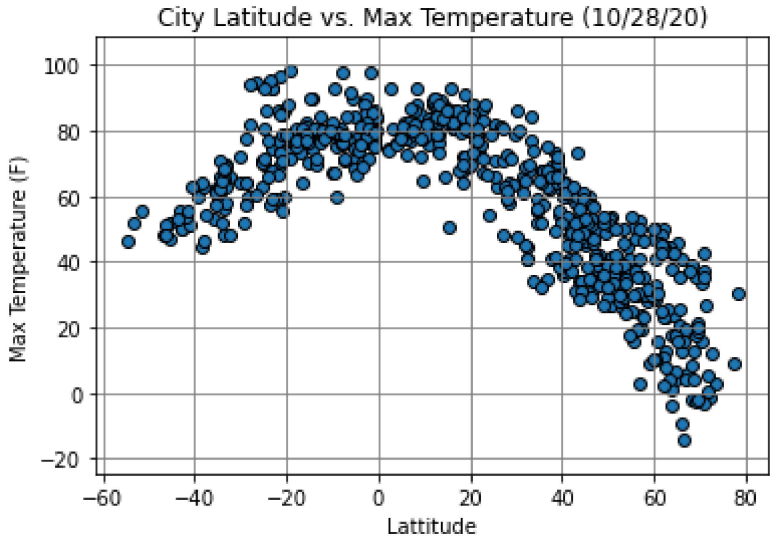
Latitude vs. Temperature Plot

In [19]:

```
# Store today's date to variable
today = date.today()

# Defined a method to create and display scatter plot.
# Which takes arguments as y-axis values, y-axis label, title of plot & Image name to save the plot

def lat_plot(yaxis, ylabel, title, image_name):
    # Plot scatter plot with Lattitude as x-axis and provided Y-axis values
    plt.scatter(city_weather["Lat"], yaxis, edgecolors="black")

    # Set the title, x & y labels, y limits for the plot
    plt.title(title)
    plt.xlabel("Lattitude")
    plt.ylabel(ylabel)
    plt.ylim(yaxis.min()-10, yaxis.max()+10)

    # Display grid with grey color
    plt.grid(color="grey")

    # Save the plot with image_name provided in specified path
    plt.savefig(f"Plot_Images/{image_name}")

    # Display plot
    plt.show()
```

In [20]:

```
# Store the title for Lattitude Vs Temperature plot to variable
title = "City Latitude vs. Max Temperature ({})".format(today.strftime("%m/%d/%y"))

# Call lat_plot method with below parameters:
# city_weather["Max Temp"] as y axis values
# y label as "Max Temperature (F)"
# title as stored above in title variable
# Image name as "1_Lat_Temp_Scatter.png"

lat_plot(city_weather["Max Temp"], "Max Temperature (F)", title, "1_Lat_Temp_Scatter.png")
```

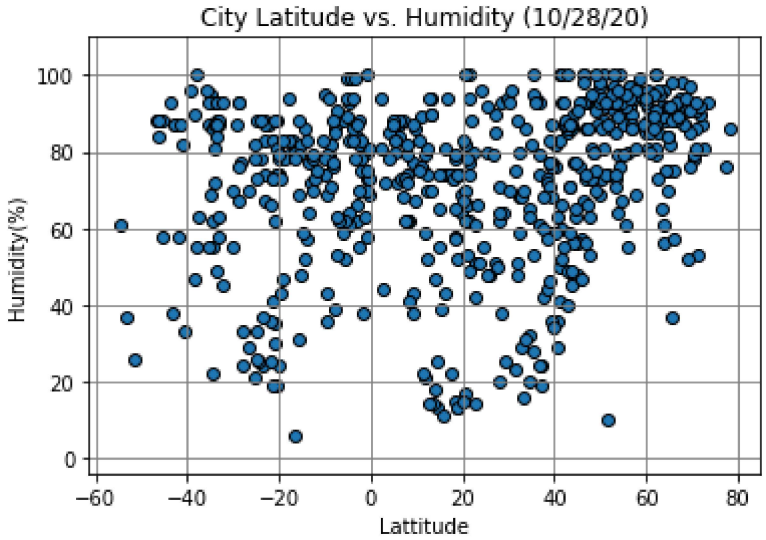


It appears that temperature and lattitude are corelated. As moving towards equator, temp appears to be increasing.

Latitude vs. Humidity Plot

In [21]:

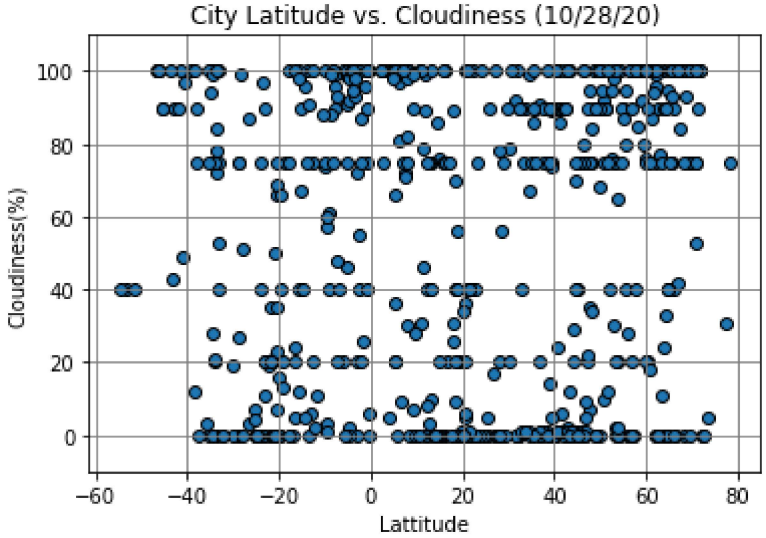
```
# Store the title for Lattitude Vs Humidity plot to variable
title = "City Latitude vs. Humidity ({})".format(today.strftime("%m/%d/%y"))

# Call lat_plot method and with parameters:
# city_weather["Humidity"] as y axis values
# "Humidity(%)" as ylabel
# title variable as plot title
# "2_Lat_Humidity_Scatter.png" as Image name

lat_plot(city_weather["Humidity"], "Humidity(%)", title, "2_Lat_Humidity_Scatter.png")
```



It appears that Lattitude and Humidity are not corelated. There are some data clusters which may indicate that there are more number of cities clusterd in that area

Latitude vs. Cloudiness Plot

In [22]:
```
# Store the title for Lattitude Vs cloudiness plot to variable
title = "City Latitude vs. Cloudiness ({})".format(today.strftime("%m/%d/%y"))

# Call lat_plot method and with parameters:
# city_weather["Cloudiness"] as y axis values
# "Cloudiness(%)" as ylabel
# title variable as plot title
# "3_Lat_Cloudiness_Scatter.png" as Image name

lat_plot(city_weather["Cloudiness"], "Cloudiness(%)", title, "3_Lat_Cloudiness_Scatter.png")
```

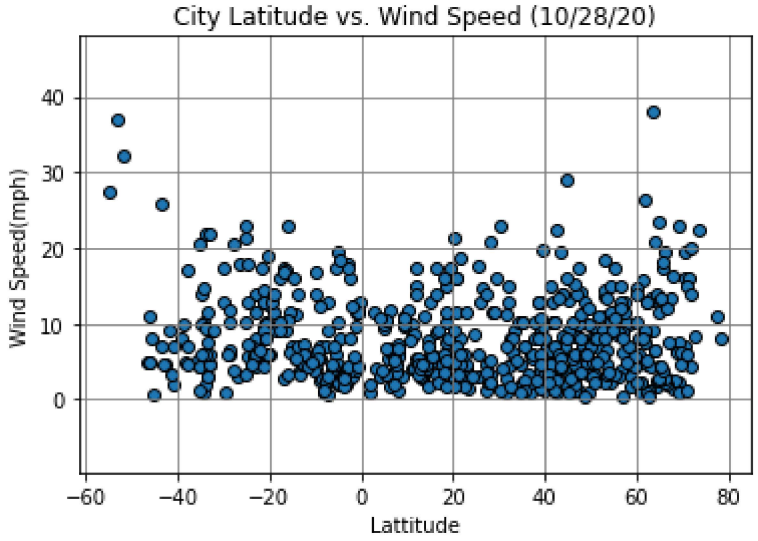


Lattitude and cloudiness does not seem to corelate at all. However there are major data clusters at 0,78,90,100

Latitude vs. Wind Speed Plot

In [23]:

```
# Store the title for Lattitude Vs Wind Speed plot to variable
title = "City Latitude vs. Wind Speed ({})".format(today.strftime("%m/%d/%y"))

# Call lat_plot method and with parameters:
# city_weather["Wind Speed"] as y axis values
# "Wind Speed(mph)" as ylabel
# title variable as plot title
# "4_Lat_WindSpeed_Scatter.png" as Image name

lat_plot(city_weather["Wind Speed"], "Wind Speed(mph)", title, "4_Lat_WindSpeed_Scatter.png")
```



Lattitude and wind speed does not seem to corelate at all. However across the lattitude between -40 to 70, major data clusters are between wind speed of 0-10

Linear Regression

In [24]:

```
# Store cities with lattitude >= 0 to north_cities df & display count
north_cities = city_weather.loc[city_weather["Lat"] >= 0 , :]
print(north_cities.count())

# Store cities with lattitude < 0 to south_cities df & display count
south_cities = city_weather.loc[city_weather["Lat"] < 0 , :]
print(south_cities.count())
```

```
City          407
Lat           407
Lng           407
Max Temp      407
Humidity      407
Cloudiness    407
Wind Speed    407
Country       407
Date          407
dtype: int64
City          197
Lat           197
Lng           197
Max Temp      197
Humidity      197
Cloudiness    197
Wind Speed    197
Country       197
Date          197
dtype: int64
```

In [25]:

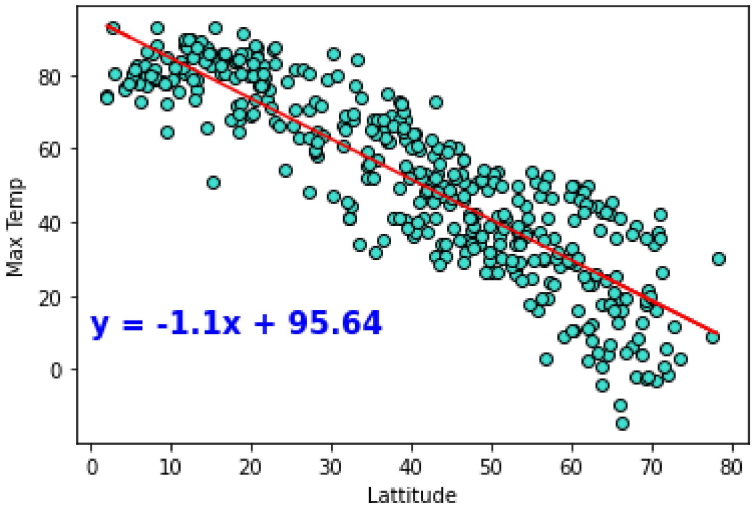
```
# Defined method to calculate linear regression, parameters as below:
# y_values = series/column values to plot on y-axis
# ylabel = string to display as y-label
# image_name = .png file name to save plot as image
# hemisphere = Values as "North" or "South" to get the Lattitude as x-values from respective dataframe
# eqplace = location to place equation on plot

def lat_linear(y_values, ylabel, image_name, hemisphere, eqplace):
    # Get the x-values according to which hemisphere has passed when calling function/methid
    if hemisphere == "North":
        x_values = north_cities["Lat"]
    elif hemisphere == "South":
        x_values = south_cities["Lat"]

    # Get the linear equation
    (slope, intercept, rvalue, pvalue, stderr) = linregress(x_values, y_values)
    regress_values = x_values * slope + intercept
    line_eq = "y = " + str(round(slope,2)) + "x + " + str(round(intercept,2))

    # Plot the scatter plot
    plt.scatter(x_values,y_values, edgecolors="black", facecolors="turquoise")
    plt.plot(x_values,regress_values,"r-")

    # Annotate the line_equation on graph
    plt.annotate(line_eq,eqplace,fontsize=15,color="blue",fontweight="bold")

    # Set label for x & y axis
    plt.xlabel("Lattitude")
    plt.ylabel(ylabel)

    # Print r-value
    print(f"The r-value is: {rvalue}")

    # Save plot image
    plt.savefig(f"Plot_Images/{image_name}")

    # Display Plot
    plt.show()
```

Northern Hemisphere - Max Temp vs. Latitude Linear Regression

In [26]:

```
# Call lat_linear method defined above to calculate regression for temp vs lattitude for Northern Hemisphere
# Parameters as below:
# y_values = north_cities["Max Temp"]
# ylabel = "Max Temp"
# image_name = "5_north_Lat_temp_regression.png"
# hemisphere = "North"
# eqplace = (0,10)

lat_linear(north_cities["Max Temp"], "Max Temp", "5_north_Lat_temp_regression.png", "North", (0,10))
```

```
The r-value is: -0.8791477262618272
```

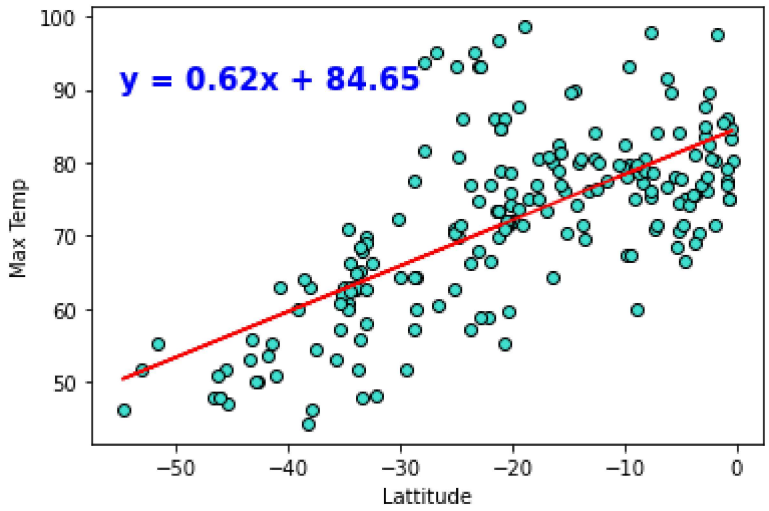


Southern Hemisphere - Max Temp vs. Latitude Linear Regression

In [27]:

```
# Call lat_linear method defined above to calculate regression for temp vs lattitude for Sourthern Hemisphere
# Parameters as below:
# y_values = south_cities["Max Temp"]
# ylabel = "Max Temp"
# image_name = "6_south_Lat_temp_regression.png"
# hemisphere = "South"
# eqplace = (-55,90)

lat_linear(south_cities["Max Temp"], "Max Temp", "6_south_Lat_temp_regression.png", "South", (-55,90))
```

```
The r-value is: 0.6860489114403746
```



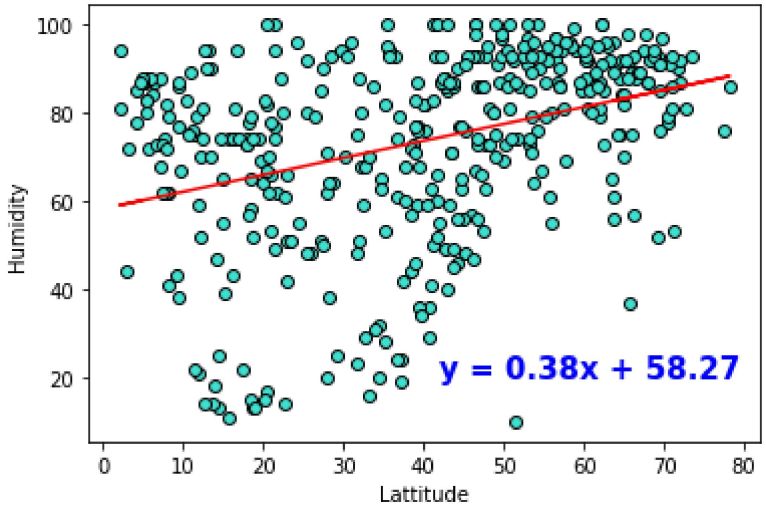
The r value shows strong between temp & lattitude i.e temp increases as we get closer to equator. However, the corelation is not as strong in Sourthern Hemisphere.

Northern Hemisphere - Humidity (%) vs. Latitude Linear Regression

In [28]:

```
# Call lat_linear method defined above to calculate regression for Humidity vs lattitude for Northern Hemisphere
# Parameters as below:
# y_values = north_cities["Humidity"]
# ylabel = "Humidity"
# image_name = "7_north_Lat_humidity_regression.png"
# hemisphere = "North"
# eqplace = (42,20)

lat_linear(north_cities["Humidity"], "Humidity", "7_north_Lat_humidity_regression.png", "North", (42,20))
```

```
The r-value is: 0.35252187096036847
```

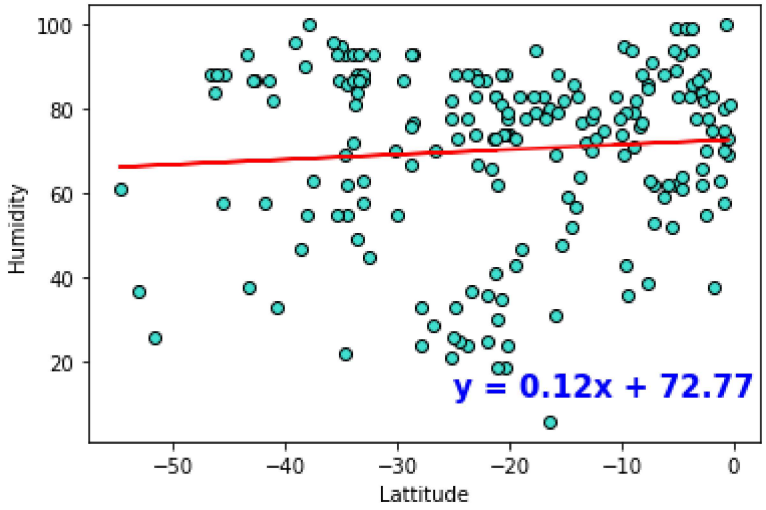


Southern Hemisphere - Humidity (%) vs. Latitude Linear Regression

In [29]:

```
# Call lat_linear method defined above to calculate regression for Humidity vs lattitude for Southern Hemisphere
# Parameters as below:
# y_values = south_cities["Humidity"]
# ylabel = "Humidity"
# image_name = "8_south_Lat_humidity_regression.png"
# hemisphere = "South"
# eqplace = (-25,12)

lat_linear(south_cities["Humidity"], "Humidity", "8_south_Lat_humidity_regression.png", "South", (-25,12))
```

```
The r-value is: 0.0756647948837791
```

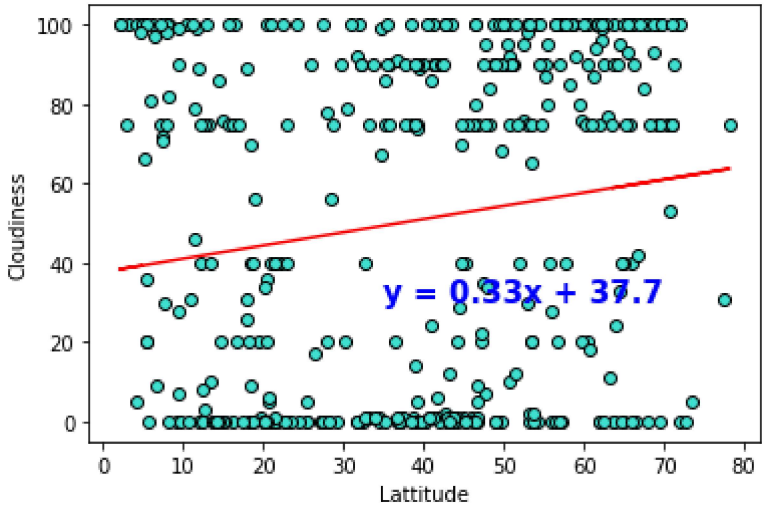


There is no strong corelation between Humidity & latitude. In northern hemisphere it's weekly corelated , indicating it gets drier as we approach equator but not as strong to prove it. In sourthern hemispehere Humidity & latitude does not seem to be corelated at all.

Northern Hemisphere - Cloudiness (%) vs. Latitude Linear Regression

In [30]:
```
# Call lat_linear method defined above to calculate regression for Cloudiness vs lattitude for Northern Hemisphere
# Parameters as below:
# y_values = north_cities["Cloudiness"]
# ylabel = "Cloudiness"
# image_name = "9_north_Lat_cloudiness_regression.png"
# hemisphere = "North"
# eqplace = (35,30)

lat_linear(north_cities["Cloudiness"], "Cloudiness", "9_north_Lat_cloudiness_regression.png", "North", (35,30))
```

```
The r-value is: 0.15790622274768779
```

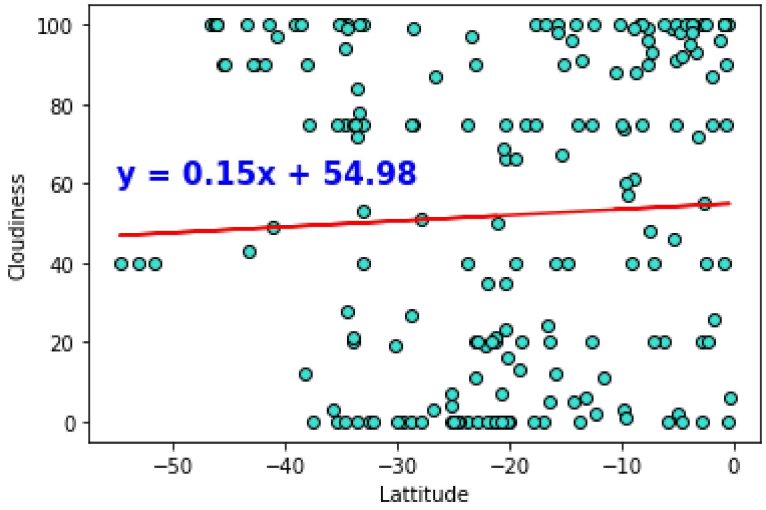


Southern Hemisphere - Cloudiness (%) vs. Latitude Linear Regression

In [31]:

```
# Call lat_linear method defined above to calculate regression for Cloudiness vs lattitude for Southern Hemisphere
# Parameters as below:
# y_values = south_cities["Cloudiness"]
# ylabel = "Cloudiness"
# image_name = "10_south_Lat_cloudiness_regression.png"
# hemisphere = "South"
# eqplace = (-55,60)

lat_linear(south_cities["Cloudiness"], "Cloudiness", "10_south_Lat_cloudiness_regression.png", "South", (-55,60))
```

```
The r-value is: 0.04947293007228338
```



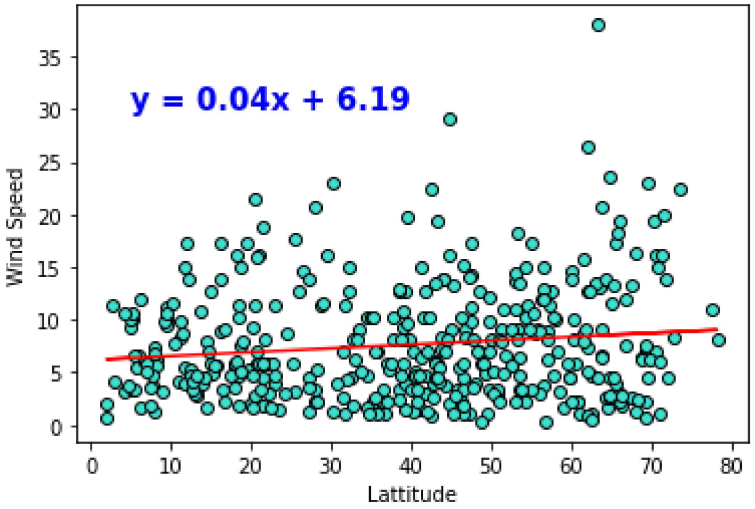
Cloudiness & latitude does not seem to have any corelation based off r value for both hemisphere.

Northern Hemisphere - Wind Speed (mph) vs. Latitude Linear Regression

In [32]:

```
# Call lat_linear method defined above to calculate regression for Wind Speed vs lattitude for Northern Hemisphere
# Parameters as below:
# y_values = north_cities["Wind Speed"]
# ylabel = "Wind Speed"
# image_name = "11_north_Lat_WindSpeed_regression.png"
# hemisphere = "North"
# eqplace = (5,30)

lat_linear(north_cities["Wind Speed"], "Wind Speed", "11_north_Lat_WindSpeed_regression.png", "North", (5,30))
```

```
The r-value is: 0.13217919504949044
```

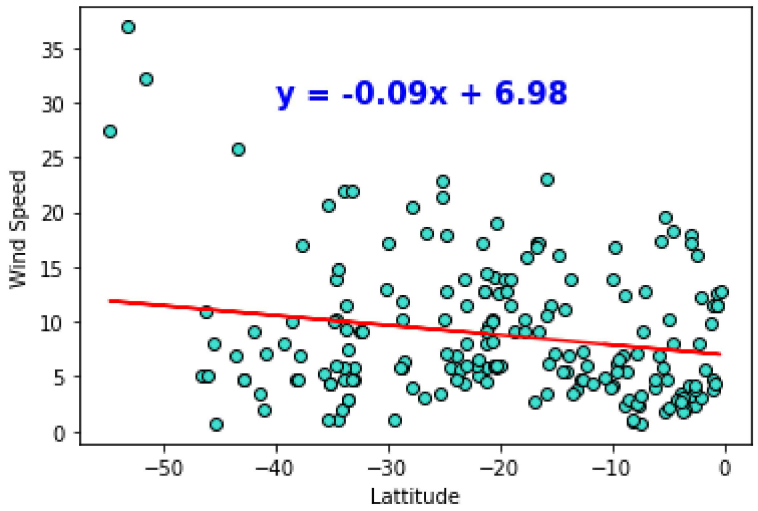


Southern Hemisphere - Wind Speed (mph) vs. Latitude Linear Regression

In [34]:

```
# Call lat_linear method defined above to calculate regression for Wind Speed vs lattitude for Southern Hemisphere
# Parameters as below:
# y_values = south_cities["Wind Speed"]
# ylabel = "Wind Speed"
# image_name = "12_south_Lat_WindSpeed_regression.png"
# hemisphere = "South"
# eqplace = (-40,30)

lat_linear(south_cities["Wind Speed"], "Wind Speed", "12_south_Lat_WindSpeed_regression.png", "South", (-40,30))
```

```
The r-value is: -0.19464129352478912
```



Wind Speed & latitude does not seem to have any corelation based off r value for both hemisphere.