Ready | Run notebook

NASA Wildfire DATA

Introduction

This Data is Near Real Time (NRT) Data extracted from NASA's MODIS Data API . I am looking at wildfires across Canada and thought to ask the following questions of the DATA:

- What is the distribution of fires in Canada, based on latitude and longitude?
- How do the brightness and confidence levels of fires vary across different regions in Canada?
- What is the fire radiative power (FRP) in different areas of the country?
- Can we identify any outliers or extreme values in the data that may indicate unusual fire events?
- Can we identify areas with high or low fire confidence levels?
- Are there any correlations between fire occurrences and other environmental factors?
- Can we detect any clusters or hotspots of fires?

I had many other questions, but for the sake of brevity and simplicity, I trimmed the number of queries to keep this project shorter than it could have been.

```
#Let's start by importing the first library - pandas
import pandas as pd
#I requested a free map key which is only valid for 30 days
MAP_KEY= "705e9a197437bc582c609cb9d9c9e4c6"
#I query the status of my map key - i should be able to run 1000 queries every 10 minutes
url='https://firms.modaps.eosdis.nasa.gov/mapserver/mapkey_status/?MAP_KEY=' + MAP_KEY
try:
    df = pd.read_json(url, typ='series')
    display(df)
except:
    print("There is an issue with the query.\n Try in your browser instead: %s" % url)
```

```
transaction_limit            1000
current_transactions            0
transaction_interval    10 minutes
dtype: object
```

```
#When in doubt, i want to be able to find out how many transactions i have run, so i write this function
def get_transaction_count() :
    count= 0
    try:
        df = pd.read_json(url, typ='series')
        count = df['current_transactions']
    except:
        print("Error in your call!")
    return count
tcount = get_transaction_count()
#At the end of the query, when I print, i will get the running total of transactions run so far
print('Your current transaction count is %i' % tcount)
```

```
Your current transaction count is 0
```

```
#I want to know what Datasets are available in this API
da_url = 'https://firms.modaps.eosdis.nasa.gov/api/data_availability/csv/' + MAP_KEY + '/all'
df = pd.read_csv(da_url)
#Display the output in a table
display(df)
```

Visualize

	data_id object	min_date object	max_date object
0	MODIS_NRT	2023-02-01	2023-11-29
1	MODIS_SP	2000-11-01	2023-01-31
2	VIIRS_NOAA20_N...	2019-12-04	2023-11-29
3	VIIRS_SNPP_NRT	2022-09-01	2023-11-29
4	VIIRS_SNPP_SP	2012-01-20	2022-08-31
5	LANDSAT_NRT	2022-06-20	2023-11-28
6	GOES_NRT	2022-08-09	2023-11-29
7	BA_MODIS	2000-11-01	2023-05-01

8 rows, showing 10 per page | Page 1 of 1

```
#I am querying my transactions again, to make sure I haven't done too many already - good habit I guess
start_count = get_transaction_count()
pd.read_csv(da_url)
end_count= get_transaction_count()
print("You used %i transactions!" % (end_count-start_count))
```

```
You used 5 transactions!
```

```python
#I want to get a "feel" for the data before I start refining my queries - can we see world data?
area_url = 'https://firms.modaps.eosdis.nasa.gov/api/area/csv/' + MAP_KEY + '/MODIS_NRT/world/1'
#We'll get a visual and do a transaction count at the same time
start_count = get_transaction_count()
df_area = pd.read_csv(area_url)
end_count= get_transaction_count()
print('You used %i transactions.' % (end_count-start_count))
df_area
#Now that I have the data table below, I can start inspecting the columns to see what data types they hold
#I expect the same format to be applied to all countries in this dataset
#The other thing I want to see is if there are any missing values.
#On further inspection, I see that there are no missing values, soit's clean
```

```
You used 36 transactions.
```

Visualize

	latitude float64 -39.42406 - 57.44...	longitude float64 -129.70062 - 164.4...	brightness float64 300.04 - 458.41	scan float64 1.0 - 4.79	track float64 1.0 - 2.0	acq_date object 2023-11-29 100%	acq_time int64 15 - 1429	s
0	-17.75074	-42.02003	301.22	4.28	1.91	2023-11-29	15	T
1	-16.31449	-42.57652	342	4.54	1.96	2023-11-29	15	T
2	-16.31315	-42.58186	333.64	4.54	1.96	2023-11-29	15	T
3	-16.31114	-42.53852	323.1	4.5	1.95	2023-11-29	15	T
4	-16.3095	-42.54318	331.06	4.5	1.95	2023-11-29	15	T
5	-16.15796	-40.87572	305.38	3.34	1.72	2023-11-29	15	T
6	-16.14577	-40.88031	316.32	3.34	1.72	2023-11-29	15	T
7	-16.14277	-40.85105	310.67	3.32	1.72	2023-11-29	15	T
8	-15.87849	-40.48719	308.3	3.08	1.67	2023-11-29	15	T
9	-15.87822	-40.48042	318.22	3.08	1.67	2023-11-29	15	T

4697 rows, showing 10 per page Page 1 of 470

```python
#In this query, I am looking at a region, it happens to be Canada - specified by longtitude and latitude
area_url = 'https://firms.modaps.eosdis.nasa.gov/api/area/csv/' + MAP_KEY +'/MODIS_NRT/56,1.5,106,34/3'
df_area = pd.read_csv(area_url)
df_area
```

Visualize

	latitude float64 10.3384 - 33.60555	longitude float64 56.09797 - 105.977...	brightness float64 300.02 - 397.97	scan float64 1.0 - 4.79	track float64 1.0 - 2.0	acq_date object 2023-11-29 40.1% 2023-11-27 35.5% 2023-11-28 24.4%	acq_time int64 257 - 2131	s
0	26.57834	101.66646	310.06	2.4	1.49	2023-11-27	257	T
1	30.84928	104.27655	303.6	1.79	1.31	2023-11-27	257	T
2	16.87041	104.92948	313.02	1.1	1.05	2023-11-27	300	T
3	19.58054	105.3693	307.02	1.12	1.05	2023-11-27	300	T
4	20.47612	105.76268	308.3	1.1	1.04	2023-11-27	300	T
5	21.0175	105.22166	306.37	1.17	1.08	2023-11-27	300	T
6	21.62421	105.35273	303.86	1.17	1.08	2023-11-27	300	T
7	21.85125	105.42683	305.37	1.17	1.08	2023-11-27	300	T
8	21.94267	105.8006	304.73	1.13	1.06	2023-11-27	300	T
9	21.95202	105.80203	311.38	1.13	1.06	2023-11-27	300	T

574 rows, showing 10 per page Page 1 of 58

```python
#If I want to query any country, I need to specify the 3 letter abbreviation, so I extracted this list
countries_url = 'https://firms.modaps.eosdis.nasa.gov/api/countries'
df_countries = pd.read_csv(countries_url, sep=';')
df_countries
```

Visualize

	id int64 1 - 246	abreviation object ABW 0.4% AFG 0.4% 242 others 99.2%	name object Aruba 0.4% Afghanistan 0.4% 242 others 99.2%	extent object BOX(-70.062... 0.4% BOX(60.486... 0.4% 242 others 99.2%
30	31	BOL	Bolivia	BOX(-69.666492...
31	32	BRA	Brazil	BOX(-74.0184746...
32	33	BRB	Barbados	BOX(-59.654204...
33	34	BRN	Brunei Darussalam	BOX(113.998789...
34	35	BTN	Bhutan	BOX(88.7300667...
35	36	BWA	Botswana	BOX(19.9783459...
36	37	CAF	Central African R...	BOX(14.3872660...
37	38	CAN	Canada	BOX(-141.005548...
38	39	CHE	Switzerland	BOX(5.95480920...
39	40	CHL	Chile	BOX(-109.45372...

244 rows, showing 10 per page Page 4 of 25

```python
#Canada is my country of interest, as can be seen by the "CAN" abbreviation in the 'country_id' column
canada_url = 'https://firms.modaps.eosdis.nasa.gov/api/country/csv/' + MAP_KEY + '/MODIS_NRT/CAN/6'
df_canada = pd.read_csv(canada_url)
df_canada
```

Visualize

	country_id object CAN 100%	latitude float64 43.25783 - 59.84792	longitude float64 -129.70062 - -76.1...	brightness float64 300.07 - 400.56	scan float64 1.0 - 4.61	track float64 1.0 - 1.97	acq_date object 2023-11-28 32.1% 2023-11-24 28.5% 4 others 39.4%	a
0	CAN	58.78014	-117.41367	345.18	3.05	1.66	2023-11-24	
1	CAN	49.70291	-120.3214	304.9	1	1	2023-11-24	
2	CAN	49.75622	-120.79758	322.71	1	1	2023-11-24	
3	CAN	49.82349	-120.78467	306.84	1	1	2023-11-24	
4	CAN	49.82541	-120.77117	323.21	1	1	2023-11-24	
5	CAN	49.87561	-120.67506	311.13	1	1	2023-11-24	
6	CAN	49.90153	-120.88832	307.62	1	1	2023-11-24	
7	CAN	50.55973	-123.47666	305.24	1.06	1.03	2023-11-24	

8	CAN	54.65393	-123.2545	301.06	1	1	2023-11-24
9	CAN	54.66262	-123.25829	314.31	1	1	2023-11-24

312 rows, showing 10 per page Page 1 of 32

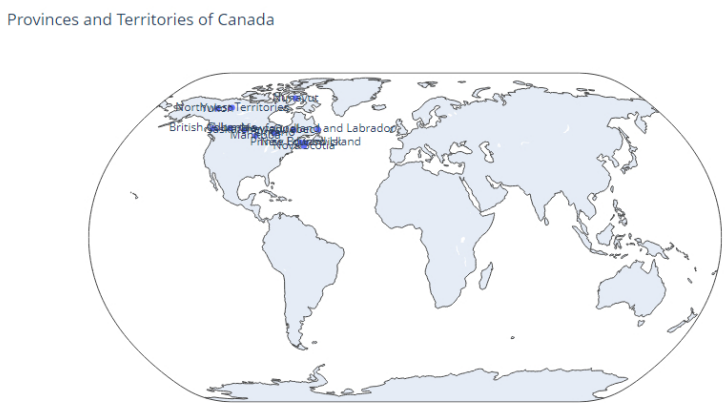
```
#I wanted a list of the columns and extracted this from the dataframe
df_canada.columns.tolist()
```

```
['country_id',
 'latitude',
 'longitude',
 'brightness',
 'scan',
 'track',
 'acq_date',
 'acq_time',
 'satellite',
 'instrument',
 'confidence',
 'version',
 'bright_t31',
 'frp',
 'daynight']
```

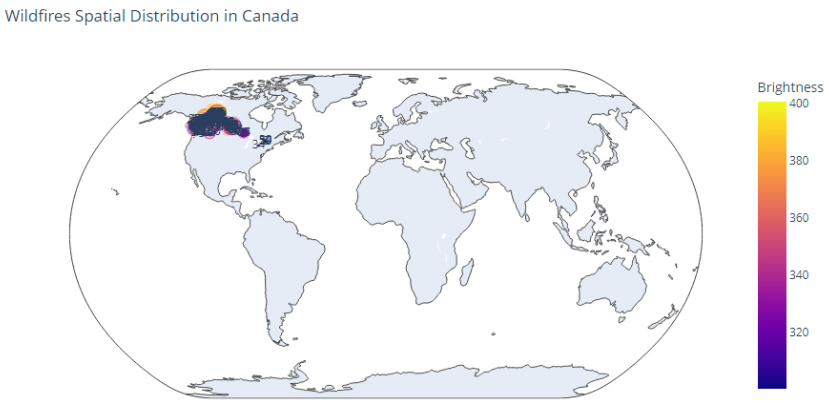
```
#I want a list of the main provinces in Canada, with their corresponding longtitude and latitude
import pandas as pd

# List of provinces and territories in Canada
data = {
    'Name': ['Alberta', 'British Columbia', 'Manitoba', 'New Brunswick', 'Newfoundland and Labrador', 'Nova Scotia', '
    'Latitude': [53.9333, 53.7267, 49.8951, 46.5653, 53.1355, 44.6819, 51.2538, 46.5107, 52.9399, 52.9399, 64.8255, 70
    'Longitude': [-116.5765, -127.6476, -97.1384, -66.4619, -57.6604, -63.7443, -85.3232, -63.4168, -73.5491, -106.450
}

# I create a DataFrame
df = pd.DataFrame(data)

# Show the DataFrame
print(df)
```

```
                        Name  Latitude  Longitude
0                    Alberta   53.9333  -116.5765
1           British Columbia   53.7267  -127.6476
2                   Manitoba   49.8951   -97.1384
3              New Brunswick   46.5653   -66.4619
4   Newfoundland and Labrador   53.1355   -57.6604
5                 Nova Scotia   44.6819   -63.7443
6                     Ontario   51.2538   -85.3232
7       Prince Edward Island   46.5107   -63.4168
8                    Quebec   52.9399   -73.5491
9               Saskatchewan   52.9399  -106.4509
10     Northwest Territories   64.8255  -124.8457
11                   Nunavut   70.2998   -83.0410
12                    Yukon   64.2823  -135.0000
```

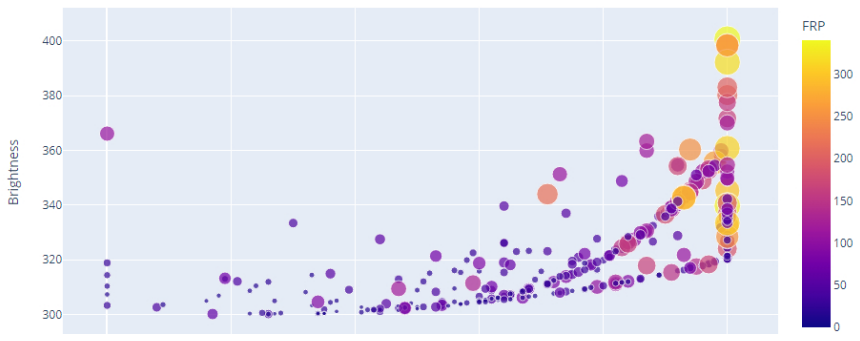
```
#With my list of provinces in hand, I want to see them on a map
import plotly.express as px
#I start by importing plotly to create my map
# I add the Data for all of the provinces and territories in Canada
data = {
    'Name': ['Alberta', 'British Columbia', 'Manitoba', 'New Brunswick', 'Newfoundland and Labrador', 'Nova Scotia', '
    'Latitude': [53.9333, 53.7267, 49.8951, 46.5653, 53.1355, 44.6819, 51.2538, 46.5107, 52.9399, 52.9399, 64.8255, 70
    'Longitude': [-116.5765, -127.6476, -97.1384, -66.4619, -57.6604, -63.7443, -85.3232, -63.4168, -73.5491, -106.450
}

# I create a DataFrame
df = pd.DataFrame(data)

# I create a map using Plotly
fig = px.scatter_geo(df, lat='Latitude', lon='Longitude', text='Name', projection='natural earth', title='Provinces an

# Here I show the map - it's interactive, so you can zoom in and scroll about
fig.show()
```

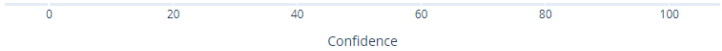


```python
import pandas as pd
import plotly.express as px

# Now that we know where all the Provinces are, lets create a visual to show Fire Distribution

# I create a scatter map using Plotly
fig = px.scatter_geo(
    df_canada,
    lat='latitude',
    lon='longitude',
    text='confidence',
    color='brightness',
    size='frp',
    title='Wildfires Spatial Distribution in Canada',
    projection='natural earth'
)

# I customize the layout
fig.update_layout(
    coloraxis_colorbar=dict(title='Brightness'),
    geo=dict(showland=True),
)

# I show the map, which is interactive, with tooltips to show you the location and other data on hover
fig.show()

```

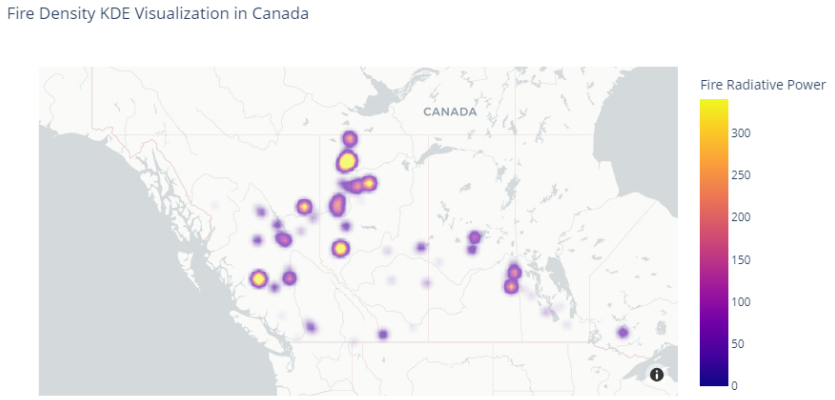



```python
import pandas as pd
import plotly.express as px

# We've seen the distribution of wildfires.  Now, lets compare confidence and brightness

# I create a scatter plot graphic
fig = px.scatter(
    df_canada,
    x='confidence',
    y='brightness',
    title='Confidence vs Brightness Scatterplot for Wildfires in Canada',
    labels={'confidence': 'Confidence', 'brightness': 'Brightness'},
    hover_data=['latitude', 'longitude'],  # Additional information on hover
    color='frp',  # Color by Fire Radiative Power (FRP)
    size='frp',   # Size by Fire Radiative Power (FRP)
    opacity=0.7,   # Set opacity for better visibility of overlapping points
)

# Let's customize the layout
fig.update_layout(
    coloraxis_colorbar=dict(title='FRP'),
)

# Let's see the graphic
fig.show()


```

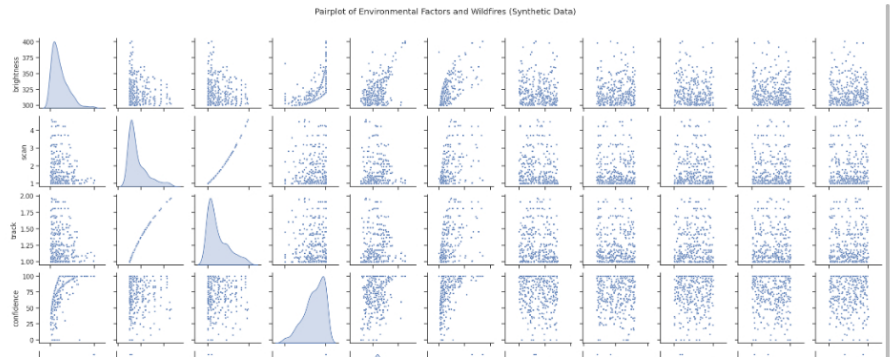


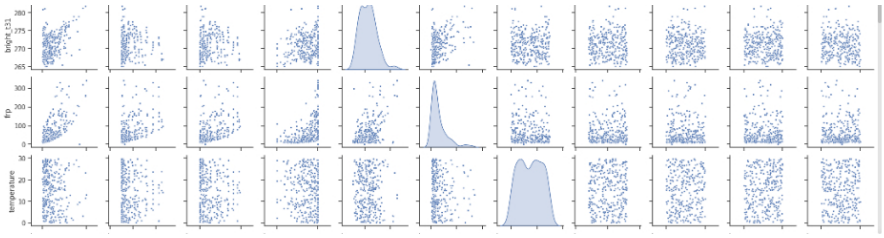



```python
import pandas as pd
import plotly.express as px

# What kind of energy or radiative power can these fires generate across Canada

# Let's create a fire density visualization
fig = px.density_mapbox(
    df_canada,
    lat='latitude',
    lon='longitude',
    z='frp',  # Use FRP as the density measure
    radius=10,  # Adjust the radius of each point for better visualization
    title='Fire Density KDE Visualization in Canada',
    labels={'frp': 'Fire Radiative Power'},
    mapbox_style='carto-positron',
    zoom=3,  # Adjust the initial zoom level
    opacity=0.6,  # Set opacity for better visibility
)

# Show the graphic
fig.show()
```

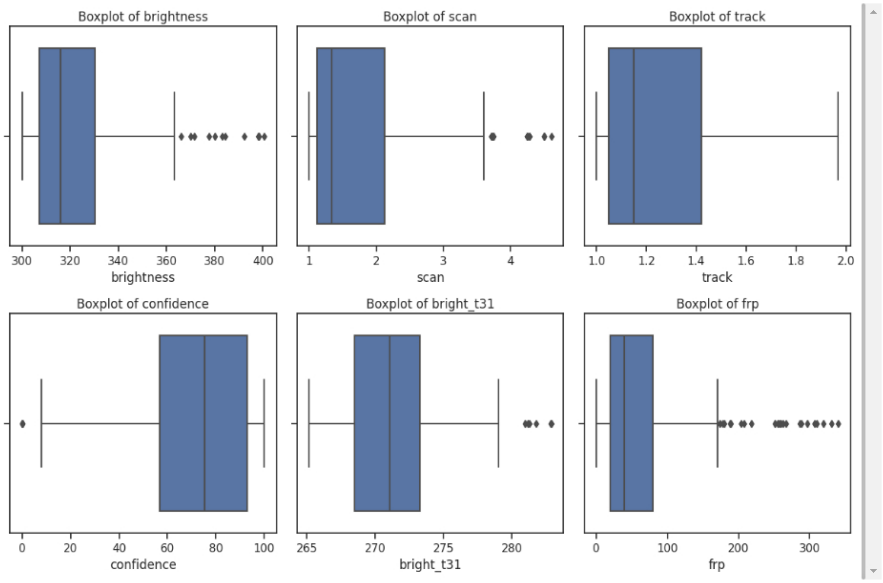



```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Time for some simulations - I've loaded some synthetic (Random) data into the query
#The simulation takes into consideration, various environmental factors

# Here's the simulation query
np.random.seed(42)
df_synthetic = df_canada.copy()
df_synthetic['temperature'] = np.random.uniform(0, 30, len(df_synthetic))
df_synthetic['humidity'] = np.random.uniform(20, 80, len(df_synthetic))
df_synthetic['wind_speed'] = np.random.uniform(0, 20, len(df_synthetic))
df_synthetic['elevation'] = np.random.uniform(0, 5000, len(df_synthetic))
df_synthetic['precipitation'] = np.random.uniform(0, 50, len(df_synthetic))

# We enter the relevant columns for the graphic, including our random factors
columns_of_interest = ['brightness', 'scan', 'track', 'confidence', 'bright_t31', 'frp', 'temperature', 'humidity', 'w

# Here we create a DataFrame with the columns we chose
df_selected = df_synthetic[columns_of_interest]

# Let's show the graphic
sns.set(style="ticks")
sns.pairplot(df_selected, height=2, markers=".", diag_kind="kde")
plt.suptitle("Pairplot of Environmental Factors and Wildfires (Synthetic Data)", y=1.02)
plt.show()
```





```python
import matplotlib.pyplot as plt
import seaborn as sns

# Is there anything that is totally random, far outside of the normal fire patterns that should concern us?
#We use the boxplots to chart fires, to see if there is anything weird or odd

# Let's choose the columns we want to query in order to answer these questions
columns_of_interest = ['brightness', 'scan', 'track', 'confidence', 'bright_t31', 'frp']

# Let's create a DataFrame from our selected columns
df_selected = df_canada[columns_of_interest]

# We create thesize of our graphic
plt.figure(figsize=(12, 8))

# We specify the boxes to be added into our graphic
for i, column in enumerate(columns_of_interest, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(x=df_selected[column])
    plt.title(f'Boxplot of {column}')

#Show us the graphic
plt.tight_layout()
plt.show()
```

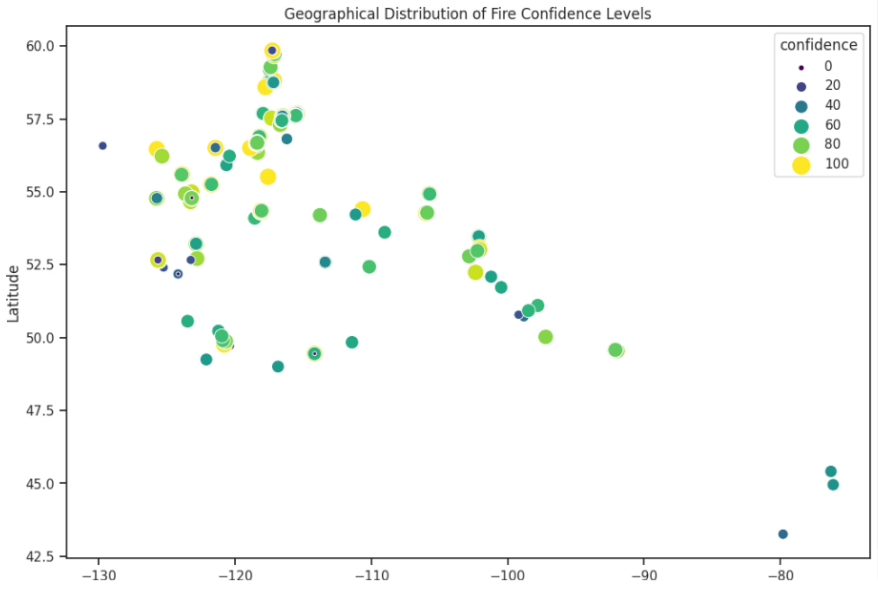



```python
#If you're more of a numbers person, we used the data in the table below to create the above graphic
# We calculated the IQR (Interquartile Range) for each column
Q1 = df_selected.quantile(0.25)
Q3 = df_selected.quantile(0.75)
IQR = Q3 - Q1

# We try to show any weird or odd values
outliers = ((df_selected < (Q1 - 1.5 * IQR)) | (df_selected > (Q3 + 1.5 * IQR))).any(axis=1)

# Show the rows with potentially weird numbers
print(df_selected[outliers])
```

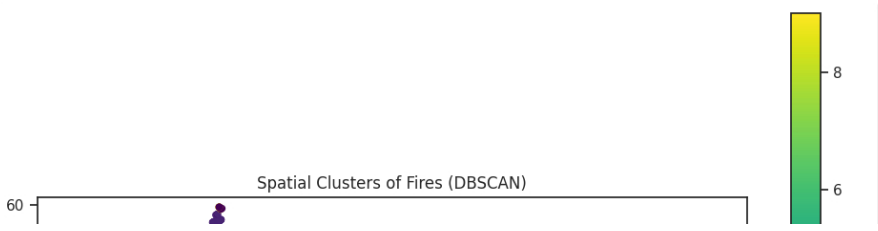
```
      brightness  scan  track  confidence  bright_t31     frp
88        360.24  1.98   1.37          94      277.51  250.95
89        302.30  4.61   1.97          48      267.13   96.10
99        392.17  1.31   1.14         100      281.27  319.34
122       303.26  2.23   1.45           0      265.94   29.85
123       300.59  1.12   1.05          34      280.95    8.43
138       314.32  1.24   1.11           0      270.25   24.28
148       369.98  1.01   1.01         100      277.81  123.77
166       384.34  1.29   1.13         100      269.83    0.00
174       303.29  3.71   1.80          54      269.74   71.74
177       340.72  2.50   1.52         100      269.89  188.92
200       336.53  2.69   1.57          90      273.38  180.32
201       343.93  2.61   1.55          71      274.62  218.00
218       310.31  1.14   1.06           0      265.18   19.55
227       354.21  1.79   1.31          92      276.47  173.98
229       304.55  4.50   1.95          34      268.63   90.42
230       311.41  4.50   1.95          59      267.15  132.15
231       309.37  4.50   1.95          47      266.90  119.99
234       342.85  3.19   1.69          93      275.94  288.39
235       342.58  3.19   1.69          93      276.09  286.31
237       304.83  1.29   1.13          60      281.16   12.99
248       318.81  1.18   1.08           0      269.98   28.40
```

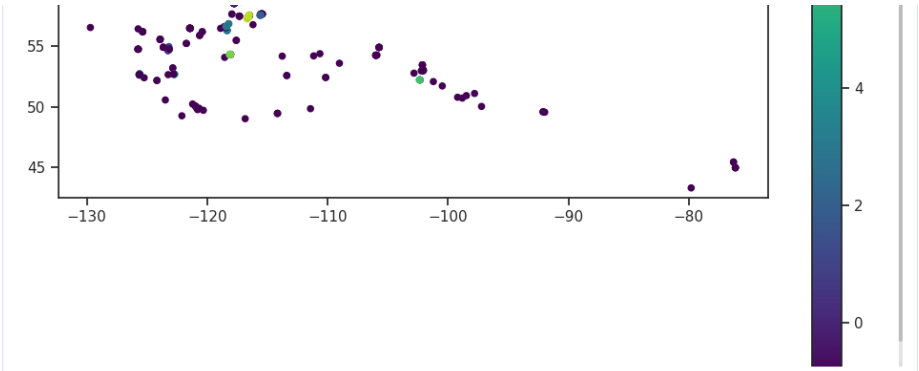
```
251     398.41  1.03   1.01        100     282.85  257.33
272     383.02  1.13   1.06        100     278.95  203.35
284     318.27  4.28   1.91         97     270.90  170.42
285     317.82  4.28   1.91         87     270.88  165.58
286     315.31  4.28   1.91         91     271.01  149.74
287     328.38  4.25   1.91        100     273.02  254.99
288     333.39  4.25   1.91        100     273.25  306.63
289     311.77  4.25   1.91         82     271.66  123.89
290     302.30  4.25   1.91         48     271.40   79.24
```

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# We now want to know what the fire confidence levels are across Canada and what the spread looks like

# Let's choose our columns for analysis
columns_of_interest = ['latitude', 'longitude', 'confidence']

# I create a DataFrame with selected columns
df_selected = df_canada[columns_of_interest]

# I create a scatter plot to display the distribution of fire confidence levels
plt.figure(figsize=(12, 8))
sns.scatterplot(x='longitude', y='latitude', hue='confidence', data=df_selected, palette='viridis', size='confidence',
plt.title('Geographical Distribution of Fire Confidence Levels')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()

```




```python
import geopandas as gpd
from shapely.geometry import Point
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

# Are there any clusters or hotspots that we can pay more attention to?

# Let's choose relevant columns for our analysis
columns_for_spatial_analysis = ['latitude', 'longitude', 'confidence']

# I create a DataFrame
df_spatial = df_canada[columns_for_spatial_analysis]

# First I need to convert the DataFrame to a GeoDataFrame
geometry = [Point(xy) for xy in zip(df_spatial['longitude'], df_spatial['latitude'])]
gdf = gpd.GeoDataFrame(df_spatial, geometry=geometry)

# I need to standardize the confidence values
scaler = StandardScaler()
gdf['confidence_scaled'] = scaler.fit_transform(gdf[['confidence']].values)

# I apply the DBSCAN clustering model
dbscan = DBSCAN(eps=0.5, min_samples=5)
gdf['cluster'] = dbscan.fit_predict(gdf[['longitude', 'latitude', 'confidence_scaled']])

# Let's plot and display the clusters
fig, ax = plt.subplots(figsize=(12, 8))
gdf.plot(column='cluster', cmap='viridis', markersize=20, legend=True, ax=ax)
plt.title('Spatial Clusters of Fires (DBSCAN)')
plt.show()

```






```python
#We are at the end of the Project - Let's see how we did with the transactions
def get_transaction_count() :
    count= 0
    try:
        df = pd.read_json(url, typ='series')
        count = df['current_transactions']
    except:
        print("Error in your call!")
    return count
tcount = get_transaction_count()
#Let's see the running total of transactions so far
print('Your current transaction count is %i' % tcount)
```

```
Your current transaction count is 292
```

Visualize data from Select dataframe...

Start by selecting a dataframe

No data, no party

AI | Code | Text | SQL | Chart | Input