In [1]:

```python
import pandas as pd
CA = pd.read_csv('mapdataall.csv')
```

In [2]:

```python
CA.head()
```

Out[2]:

	incident_name	incident_is_final	incident_date_last_update	incident_date_created	incident_administrative_unit	incident_administrativ
0	Bridge Fire	True	2018-01-09 13:46:00	2017-10-31 11:22:00	Shasta-Trinity National Forest	
1	Pala Fire	True	2020-09-16 14:07:35	2009-05-24 14:56:00	CAL FIRE San Diego Unit	
2	River Fire	True	2013-02-28 20:00:00	2013-02-24 08:16:00	CAL FIRE San Bernardino Unit	
3	Fawnskin Fire	True	2013-04-22 09:00:00	2013-04-20 17:30:00	San Bernardino National Forest	
4	Gold Fire	True	2013-05-01 07:00:00	2013-04-30 12:59:00	CAL FIRE Madera-Mariposa-Merced Unit	

5 rows × 23 columns

In [3]:

```python
len(CA)
```

Out[3]:

```
1719
```

In [4]:

```python
CA.columns
```

Out[4]:

```
Index(['incident_name', 'incident_is_final', 'incident_date_last_update',
       'incident_date_created', 'incident_administrative_unit',
       'incident_administrative_unit_url', 'incident_county',
       'incident_location', 'incident_acres_burned', 'incident_containment',
       'incident_control', 'incident_cooperating_agencies',
       'incident_longitude', 'incident_latitude', 'incident_type',
       'incident_id', 'incident_url', 'incident_date_extinguished',
       'incident_dateonly_extinguished', 'incident_dateonly_created',
       'is_active', 'calfire_incident', 'notification_desired'],
      dtype='object')
```

In [ ]:

```python

```

In [5]:

```python
CA.columns
```

Out[5]:

```
Index(['incident_name', 'incident_is_final', 'incident_date_last_update',
       'incident_date_created', 'incident_administrative_unit',
       'incident_administrative_unit_url', 'incident_county',
       'incident_location', 'incident_acres_burned', 'incident_containment',
       'incident_control', 'incident_cooperating_agencies',
       'incident_longitude', 'incident_latitude', 'incident_type',
       'incident_id', 'incident_url', 'incident_date_extinguished',
       'incident_dateonly_extinguished', 'incident_dateonly_created',
       'is_active', 'calfire_incident', 'notification_desired'],
      dtype='object')
```

In [6]:

```
new = CA[["incident_latitude", "incident_longitude","incident_containment", "incident_county",
'incident_acres_burned', 'is_active', 'incident_date_last_update']].copy()
```

In [7]:

```
new = new.dropna()
```

In [8]:

```
new.isnull().sum()
```

Out[8]:

```
incident_latitude            0
incident_longitude           0
incident_containment         0
incident_county              0
incident_acres_burned        0
is_active                    0
incident_date_last_update    0
dtype: int64
```

In [9]:

```
active_fires = new[new.is_active == 'Y']
```

In [10]:

```
import plotly.express as px

fig = px.scatter(active_fires, x = "incident_longitude", y = "incident_latitude", color = 'is_activ
e', color_discrete_sequence=["red"], size = "incident_containment", hover_data =
["incident_county"])
fig.show()
```

In [11]:

```
dormant_fires = new[new.is_active == 'N']
```

In [12]:

```
fig = px.scatter(dormant_fires, x = "incident_longitude", y = "incident_latitude", color = 'is_acti
ve', color_discrete_sequence=["green"], size = "incident_containment", hover_data = ["incident_coun
ty"])
fig.show()
```

In [13]:

```
import numpy as np
def zoom_center(lons: tuple=None, lats: tuple=None, lonlats: tuple=None, format: str='lonlat', proj
ection: str='mercator',
    width_to_height: float=2.0) -> (float, dict):

    if lons is None and lats is None:
        if isinstance(lonlats, tuple):
            lons, lats = zip(*lonlats)
        else:
            raise ValueError(
                'Must pass lons & lats or lonlats'
            )

    maxlon, minlon = max(lons), min(lons)
    maxlat, minlat = max(lats), min(lats)
    center = {
        'lon': round((maxlon + minlon) / 2, 6),
        'lat': round((maxlat + minlat) / 2, 6)
    }
    lon_zoom_range = np.array([
        0.0007, 0.0014, 0.003, 0.006, 0.012, 0.024, 0.048, 0.096,
        0.192, 0.3712, 0.768, 1.536, 3.072, 6.144, 11.8784, 23.7568,
        47.5136, 98.304, 190.0544, 360.0
    ])

    if projection == 'mercator':
        margin = 1.2
        height = (maxlat - minlat) * margin * width_to_height
        width = (maxlon - minlon) * margin
        lon_zoom = np.interp(width , lon_zoom_range, range(20, 0, -1))
        lat_zoom = np.interp(height, lon_zoom_range, range(20, 0, -1))
        zoom = round(min(lon_zoom, lat_zoom), 2)
    else:
        raise NotImplementedError(
            f'{projection} projection is not implemented'
        )

    return zoom, center
```

In [14]:

```
dormant_fires['incident_date_last_updatev2'] =
pd.to_datetime(dormant_fires.incident_date_last_update)
```

```
C:\Users\Acer\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [15]:

```
top_10_dates = dormant_fires.nlargest(50, 'incident_date_last_updatev2')
```

In [16]:

```
top_10_largest = dormant_fires.nlargest(10, 'incident_acres_burned')
```

In [17]:

```
zoom, center = zoom_center(lons = list(top_10_largest.incident_longitude), lats = list(top_10_large
st.incident_latitude))
```

In [18]:

```
fig = px.scatter_mapbox(top_10_largest,
                        lat= "incident_latitude", lon="incident_longitude", color='is_active', hove
_data = ["incident_county"], opacity = 0.6,
                        size = "incident_acres_burned", color_discrete_sequence=["green"],
mapbox_style="open-street-map", zoom = zoom - 0.75, center = center)


fig.show()
```

In [19]:

```
zoom, center = zoom_center(lons = list(top_10_dates.incident_longitude), lats = list(top_10_dates.i
ncident_latitude))
```

In [28]:

```
fig = px.scatter_mapbox(top_10_dates,
                        lat= "incident_latitude", lon="incident_longitude", color='is_active', hove
_data = ["incident_county"], opacity = 0.6,
                        size = "incident_acres_burned", color_discrete_sequence=["green"],
mapbox_style="open-street-map", zoom = zoom, center = center)


fig.show()
```

In [21]:

```
zoom, center = zoom_center(lons = list(active_fires.incident_longitude), lats = list(active_fires.i
ncident_latitude))
```

In [22]:

```
fig = px.scatter_mapbox(active_fires,
                        lat= "incident_latitude", lon="incident_longitude", color='is_active', hove
_data = ["incident_county"], opacity = 0.6,
                        size = "incident_containment", color_discrete_sequence=["red"],
mapbox_style="open-street-map", zoom = zoom - 0.75, center = center)


fig.show()
```

In [23]:

```
largest_burned = new.nlargest(15, 'incident_acres_burned')
```

In [24]:

```
combined_df = pd.concat([active_fires, top_10_largest])
```

In [25]:

```
zoom, center = zoom_center(lons = list(combined_df.incident_longitude), lats = list(combined_df.inc
ident_latitude))
```

In [26]:

```
fig = px.scatter_mapbox(combined_df,
                        lat= "incident_latitude", lon="incident_longitude", color='is_active', hove
_data = ["incident_county"], opacity = 0.6,
                        size = "incident_containment", color_discrete_sequence=["red", "green"],
mapbox_style="open-street-map", zoom = zoom - 0.75, center = center)


fig.show()
```

In [27]:

```
import plotly.express as px

fig = px.bar(largest_burned, y="incident_county", x="incident_acres_burned",
color="incident_county", orientation="h", hover_name="incident_county",
             color_discrete_sequence=px.colors.qualitative.G10
            )

fig.show()
```