```
In [ ]: #!pip install folium
```

```
In [ ]: import folium
        import pandas as pd
```

```
In [ ]: utah_df=pd.read_csv('1871-utah-postmaster-salaries.csv')
        print(utah_df.sample(5))
        utah_df.dtypes
```

```
In [ ]: utah_map_empty = folium.Map(location=[40, -111], zoom_start=6)
        utah_map_empty
```

```
In [ ]: # Create a duplicate of our starting point map to start adding markers to
        utah_map = utah_map_empty

        folium.Marker(location=[38.41, -112.339], popup="Adamsville Post Office").add_to(utah_map)
        utah_map
```

```
In [ ]: # We're going to define a function that creates an empty map that we will use to add our markers to. Each time we want to add markers, we can call this function to creat
        def create_empty_map():
            return folium.Map(location=[40, -111], zoom_start=6)

        utah_map = create_empty_map()
        utah_map
```

```
In [ ]: # Melanie Walsh function we will edit:
        def create_map_markers(row, map_name):
            folium.Marker(location=[row['lat'], row['lon']], popup=row['place']).add_to(map_name)
```

```
In [ ]: def create_map_markers(row, map_name):
            folium.Marker(location=[row['Latitude'], row['Longitude']], popup=row['PO_Name']).add_to(map_name)
```

```
In [ ]: # Check for columns with missing values
        missing_values = utah_df.isna().sum()
        print(missing_values)
```

```
In [ ]: # Filter out post offices that are missing a latitude value
        utah_df_locations = utah_df[utah_df['Latitude'].notna()]
        print(len(utah_df))
        print(len(utah_df_locations))
```

```
In [ ]: # Method 1: Using a for loop to iterate through our dataframe and add markers sequentially

        # initialize an empty map
        utah_map = create_empty_map()

        # iterrows() allows you to loop through a dataframe row by row and return the index position + the row
        for index, row in utah_df_locations.iterrows():
            print(f"Name of post office:", row[0])

        #now let's iterate through and call our function for each row
        for index, row in utah_df_locations.iterrows():
            create_map_markers(row, utah_map)

        utah_map
```

```
In [ ]: # Method 2: Using .apply() to add markers with our function for all rows

        # initialize an empty map
        utah_map = utah_map_empty

        # Now apply this function to each row in our filtered DataFrame
        # For each row, we'll pass:
        #   1. The row itself (handled automatically by .apply())
        #   2. Our map object (we need to specify this explicitly)
        #   3. The "axis" value for .apply() to indicate we want to process row by row

        # .apply() allows you to apply a function to each row in the dataframe
        utah_df_locations.apply(
            create_map_markers, # The function to apply
            map_name=utah_map,  # Additional argument to pass to the function
            axis='columns' # Process row by row instead of column by column
        )

        utah_map
```

```
In [ ]: # Melanie Walsh function we will edit:
        def create_ICE_map_markers(row, map_name):
            folium.CircleMarker(location=[row['lat'], row['lon']], raidus=100, fill=True,
                        popup=folium.Popup(f"{row['Name'].title()} <br> {row['City'].title()}, {row['State']}", max_width=200),
                          tooltip=f"{row['Name'].title()} <br> {row['City'].title()}, {row['State']}"
                         ).add_to(map_name)
```

```
In [ ]: def create_circle_markers(row, map_name):
            folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                                radius=40,
                                fill=True,
                                popup=folium.Popup(f"{row['PO_Name'].title()}", max_width=200),
                                tooltip=f"{row['PO_Name'].title()}"
                         ).add_to(map_name)
```

```
In [ ]: # initialize an empty map
        utah_map = create_empty_map()

        # call our function for each row
        utah_df_locations.apply(create_circle_markers, map_name=utah_map, axis="columns")

        utah_map
```

```
In [ ]: # alter map appearance
        def create_circle_markers(row, map_name):
            folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                                radius=8,
                                color = 'green',
                                fill=True,
                                fill_color='green',
                                fill_opacity=0.6,
                                popup=folium.Popup(f"Post Office: {row['PO_Name'].title()}", max_width=200),
                                tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                         ).add_to(map_name)
```

In [ ]:
```
# initialize an empty map
utah_map = create_empty_map()

# call our function for each row
utah_df_locations.apply(
    create_circle_markers, # The function to apply
    map_name=utah_map,  # Additional argument to pass to the function
    axis='columns' # Process row by row instead of column by column
)

utah_map
```

In [ ]:
```
# make new function to create circle markers sized by postmaster salary
def create_sized_circle_markers(row, map_name):
    folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                        radius=row['PM_Salary'],
                        fill=True,
                        popup=folium.Popup(f"Post Office: {row['PO_Name'].title()}", max_width=200),
                        tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                 ).add_to(map_name)
```

In [ ]:
```
# initialize an empty map
utah_map = create_empty_map()

# call our function for each row
utah_df_locations.apply(
    create_sized_circle_markers, # The function to apply
    map_name=utah_map,  # Additional argument to pass to the function
    axis='columns' # Process row by row instead of column by column
)

utah_map
```

In [ ]:
```
# make new function to create circle markers sized by postmaster salary - this time adjusting the radius size in pixels to make it more legible
def create_sized_circle_markers(row, map_name):
    folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                        radius=row['PM_Salary']/100,
                        fill=True,
                        popup=folium.Popup(f"Post Office: {row['PO_Name'].title()}", max_width=200),
                        tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                 ).add_to(map_name)
```

In [ ]:
```
# initialize an empty map
utah_map = create_empty_map()

# call our function for each row
utah_df_locations.apply(
    create_sized_circle_markers, # The function to apply
    map_name=utah_map,  # Additional argument to pass to the function
    axis='columns' # Process row by row instead of column by column
)

utah_map
```

In [ ]:
```
utah_df_locations.describe()
```

In [ ]:
```
def add_salary_buckets(salary):
    # Create a new column for the salary bucket
    if salary < 50:
        bucket = 'Low Salary'
    elif salary >= 50 and salary < 250:
        bucket = 'Medium Salary'
    elif salary >= 250 and salary < 1000:
        bucket = 'High Salary'
    else:
        bucket = 'Very High Salary'
    return bucket

#test out the function
add_salary_buckets(2000)
```

In [ ]:
```
utah_df_locations['Salary_Bucket']=utah_df_locations['PM_Salary'].apply(add_salary_buckets)
utah_df_locations.head()
```

In [ ]:
```
# create a function to add marker sizes based on the salary bucket
def add_marker_sizes(category):
    if category == 'Low Salary':
        return 4
    elif category == 'Medium Salary':
        return 8
    elif category == 'High Salary':
        return 12
    else:
        return 16

#test out the function
add_marker_sizes('High Salary')
```

In [ ]:
```
utah_df_locations['Marker_Size']=utah_df_locations['Salary_Bucket'].apply(add_marker_sizes)
utah_df_locations.head(10)
```

In [ ]:
```
# make new function to create circle markers sized by salary category
def create_sized_circle_markers(row, map_name):
    folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                        radius=row['Marker_Size'],
                        fill=True,
                        opacity=0.6,
                        popup=folium.Popup(f"Post Office: {row['PO_Name'].title()}", max_width=200),
                        tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                 ).add_to(map_name)
```

In [ ]:
```
# initialize an empty map
utah_map = create_empty_map()

# call our function for each row
utah_df_locations.apply(
    create_sized_circle_markers, # The function to apply
    map_name=utah_map,  # Additional argument to pass to the function
```