In [1]:
```
!pip install folium
```

```
Requirement already satisfied: folium in /opt/anaconda3/lib/python3.12/site-
packages (0.19.5)
Requirement already satisfied: branca>=0.6.0 in /opt/anaconda3/lib/python3.1
2/site-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in /opt/anaconda3/lib/python3.12/
site-packages (from folium) (3.1.4)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-p
ackages (from folium) (1.26.4)
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.12/sit
e-packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in /opt/anaconda3/lib/python3.12/
site-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/anaconda3/lib/python
3.12/site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/anaconda3/li
b/python3.12/site-packages (from requests->folium) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/lib/python3.1
2/site-packages (from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/anaconda3/lib/pyth
on3.12/site-packages (from requests->folium) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/lib/pyth
on3.12/site-packages (from requests->folium) (2024.12.14)
```

In [2]:
```
import folium
import pandas as pd
```

In [3]:
```
utah_df = pd.read_csv('1871-utah-postmaster-salaries.csv')
print(utah_df.sample(5))
utah_df.dtypes
```

```
         PO_Name      County State  PM_Salary   Latitude   Longitude
132  West Jordan   Salt Lake    UT         10  40.602000 -111.958600
41      Gunnison     Sanpete    UT         51  39.138850 -111.818814
98    Provo City        Utah    UT        300  40.233844 -111.658534
92   Pine Valley  Washington    UT         17  37.391091 -113.514122
68         Manti     Sanpete    UT        100  39.267462 -111.636585
```

Out[3]:
```
PO_Name       object
County        object
State         object
PM_Salary      int64
Latitude     float64
Longitude    float64
dtype: object
```

In [4]:
```
utah_map_empty = folium.Map(location=[40, -111], zoom_start=6)
utah_map_empty
```

Out[4]:

In [5]:
```
def create_empty_map():
    return folium.Map(location=[40, -111], zoom_start=6)

utah_map = create_empty_map()
utah_map
```

Out[5]:

In [6]:
```
# Check for columns with missing values
```

```
missing_values = utah_df.isna().sum()
print(missing_values)
```

```
PO_Name      0
County       0
State        0
PM_Salary    0
Latitude     4
Longitude    4
dtype: int64
```

In [7]:
```
# Filter out post offices that are missing a latitude value (ie. we don't ha
utah_df_locations = utah_df[utah_df['Latitude'].notna()]
print(len(utah_df))
print(len(utah_df_locations))
```

```
136
132
```

In [8]:
```
folium.Marker(location=[38.41, -112.339], popup="Adamsville Post Office").ad
utah_map
```

Out[8]:

In [9]:
```
# Melanie Walsh function we will adadpt to our dataset:
# def create_map_markers(row, map_name):
#     folium.Marker(location=[row['lat'], row['lon']], popup=row['place']).ad

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

In [10]:
```
#create a base empty map
utah_map = create_empty_map()

#generate a random row of data
sample_row = utah_df_locations.sample(1)

#use our function on the random row
create_map_markers(sample_row, utah_map)

#display the map
utah_map
```

```
/opt/anaconda3/lib/python3.12/site-packages/folium/utilities.py:101: FutureW
arning: Calling float on a single element Series is deprecated and will rais
e a TypeError in the future. Use float(ser.iloc[0]) instead
  float(coord)
/opt/anaconda3/lib/python3.12/site-packages/folium/utilities.py:107: FutureW
arning: Calling float on a single element Series is deprecated and will rais
e a TypeError in the future. Use float(ser.iloc[0]) instead
  if math.isnan(float(coord)):
/opt/anaconda3/lib/python3.12/site-packages/folium/utilities.py:109: FutureW
arning: Calling float on a single element Series is deprecated and will rais
e a TypeError in the future. Use float(ser.iloc[0]) instead
  return [float(x) for x in coords]
```

Out[10]:

In [11]:
```
# Method 1: Using a for loop to iterate through our dataframe and add marker
# initialize an empty map
utah_map = create_empty_map()
```

```
# iterrows() allows you to loop through a dataframe row by row and return th
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
```

```
Name of post office: Adamsville
Name of post office: Alma
Name of post office: Alpine City
Name of post office: American Fork
Name of post office: Bellevue
Name of post office: Bingham Canyon
Name of post office: Brigham City
Name of post office: Bullion
Name of post office: Cedar City
Name of post office: Cedar Valley
Name of post office: Centerville
Name of post office: Central City
Name of post office: Chicken Creek
Name of post office: Clarkston
Name of post office: Clifton
Name of post office: Clover Valley
Name of post office: Coalville
Name of post office: Corinne
Name of post office: Cove Creek
Name of post office: Croydon
Name of post office: Deseret
Name of post office: Diamond
Name of post office: Draper
Name of post office: Duncans Retreat
Name of post office: Echo City
Name of post office: Eden
Name of post office: Emmaville
Name of post office: Ephraim
Name of post office: Eureka
Name of post office: Fair View
Name of post office: Fairfield
Name of post office: Farmington
Name of post office: Fillmore City
Name of post office: Forest City
Name of post office: Fort Hamblin
Name of post office: Fountain Green
Name of post office: Franklin
Name of post office: Goshen
Name of post office: Grafton
Name of post office: Grantsville
```

```
Name of post office: Gunnison
Name of post office: Harrisburg
Name of post office: Harrisville
Name of post office: Heber
Name of post office: Hebron
Name of post office: Herriman
Name of post office: Holden
Name of post office: Hooper
Name of post office: Huntsville
Name of post office: Hyde Park
Name of post office: Hyrum
Name of post office: Iron City
Name of post office: Johnson
Name of post office: Kamas
Name of post office: Kanab
Name of post office: Kanarraville
Name of post office: Kanosh
Name of post office: Kaysville
Name of post office: Kelton
Name of post office: Lake Point
Name of post office: Leeds
Name of post office: Lehi City
Name of post office: Levan
Name of post office: Liberty
Name of post office: Logan
Name of post office: Lynne
Name of post office: Manti
Name of post office: Meadow
Name of post office: Mendon
Name of post office: Midway
Name of post office: Mill Creek
Name of post office: Millville
Name of post office: Minersville
Name of post office: Mona
Name of post office: Monroe
Name of post office: Morgan
Name of post office: Moroni
Name of post office: New Harmony
Name of post office: Newton
Name of post office: North Ogden
Name of post office: Ogden City
Name of post office: Ophir
Name of post office: Paradise
Name of post office: Paragonah
Name of post office: Parowan
Name of post office: Payson
Name of post office: Peoa
Name of post office: Petersburgh
Name of post office: Peterson
Name of post office: Pine Valley
```

```
Name of post office: Pinto
Name of post office: Plain City
Name of post office: Pleasant Grove
Name of post office: Portage
Name of post office: Providence
Name of post office: Provo City
Name of post office: Richfield
Name of post office: Richmond
Name of post office: Riverdale
Name of post office: Rockville
Name of post office: Saint George
Name of post office: Salina
Name of post office: Salt Creek
Name of post office: Salt Lake City
Name of post office: Santaquin
Name of post office: Scipio
Name of post office: Silver City
Name of post office: Slatersville
Name of post office: Smithfield
Name of post office: South Cottonwood
Name of post office: Spanish Fork
Name of post office: Spring City
Name of post office: Springdale
Name of post office: Springville
Name of post office: Stockton
Name of post office: Stoker
Name of post office: Summit
Name of post office: Tokersville
Name of post office: Tooele
Name of post office: Uintah
Name of post office: Union
Name of post office: Virgin City
Name of post office: Wales
Name of post office: Wallsburg
Name of post office: Wanship
Name of post office: Wasatch
Name of post office: Washington
Name of post office: Wellsville
Name of post office: West Jordan
Name of post office: Willard
Name of post office: Winsor
Name of post office: Woods Cross
/var/folders/35/7877ljzn6nj5tdww9cfrpbz40000gn/T/ipykernel_7342/2508161898.py:7: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  print(f"Name of post office:", row[0])
```

Out[11]:

In [12]:
```
# 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
```

Out[12]:

In [13]:
```
# alter map appearance
def create_circle_markers(row, map_name):
    folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                        radius=5,
                        color='red',
                        fill=True,
                        fill_color='green',
                        fill_opacity=0.5,
                        popup=folium.Popup(f"Post Office: {row['PO_Name'].tit
                        tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                        ).add_to(map_name)

# 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
```

Out[13]:

In [14]:
```
# make new function to create circle markers sized by postmaster salary - th
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'].tit
                        tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                        ).add_to(map_name)

# 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
```

Out[14]:

In [15]:
```
utah_df_locations.describe()
```

Out[15]:

	PM_Salary	Latitude	Longitude
count	132.000000	132.000000	132.000000
mean	101.098485	39.907743	-112.161190
std	344.225580	1.505091	0.593117
min	4.000000	37.006375	-113.819415
25%	12.000000	38.874099	-112.379230
50%	22.000000	40.380926	-111.973830
75%	70.000000	41.080917	-111.819912
max	3600.000000	42.187500	-111.281850

In [16]:
```
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
```

In [17]:
```
#test out the function
add_salary_buckets(2000)
```

Out[17]: 'Very High Salary'

In [18]:
```
utah_df_locations['Salary_Bucket'] = utah_df_locations['PM_Salary'].apply(ad
utah_df_locations.head()
```

```
/var/folders/35/7877ljzn6nj5tdww9cfrpbz40000gn/T/ipykernel_7342/2959544325.p
y: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
  utah_df_locations['Salary_Bucket'] = utah_df_locations['PM_Salary'].apply(
add_salary_buckets)
```

Out[18]:

	PO_Name	County	State	PM_Salary	Latitude	Longitude	Salary_Bucket
0	Adamsville	Beaver	UT	10	38.258303	-112.793835	Low Salary
1	Alma	Weber	UT	12	41.248833	-112.078275	Low Salary
2	Alpine City	Utah	UT	27	40.453283	-111.777986	Low Salary
3	American Fork	Utah	UT	130	40.375229	-111.796320	Medium Salary
4	Bellevue	Washington	UT	20	37.340815	-113.274116	Low Salary

In [19]:
```
# 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')
```

Out[19]: 12

In [20]:
```
utah_df_locations['Marker_Size'] = utah_df_locations['Salary_Bucket'].apply(
utah_df_locations.head(10)
```

```
/var/folders/35/7877ljzn6nj5tdww9cfrpbz40000gn/T/ipykernel_7342/3689819186.p
y: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
  utah_df_locations['Marker_Size'] = utah_df_locations['Salary_Bucket'].appl
y(add_marker_sizes)
```

Out[20]:

	PO_Name	County	State	PM_Salary	Latitude	Longitude	Salary_Bucket
0	Adamsville	Beaver	UT	10	38.258303	-112.793835	Low Salary
1	Alma	Weber	UT	12	41.248833	-112.078275	Low Salary
2	Alpine City	Utah	UT	27	40.453283	-111.777986	Low Salary
3	American Fork	Utah	UT	130	40.375229	-111.796320	Medium Salary
4	Bellevue	Washington	UT	20	37.340815	-113.274116	Low Salary
5	Bingham Canyon	Salt Lake	UT	12	40.541613	-112.147997	Low Salary
7	Brigham City	Box Elder	UT	400	41.510213	-112.015501	High Salary
8	Bullion	Piute	UT	12	38.410000	-112.339000	Low Salary
9	Cedar City	Iron	UT	200	37.676644	-113.057171	Medium Salary
10	Cedar Valley	Utah	UT	13	40.327171	-112.104385	Low Salary

In [21]:
```
# 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'].tit
                        tooltip=f"Postmaster Salary: ${row['PM_Salary']}"
                        ).add_to(map_name)

# 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
```

Out[21]:

In [22]:
```
usa_1877_df = pd.read_csv('1877-official-register.csv')
usa_1877_df.dtypes
```

Out[22]:
```
Name          object
State         object
Department    object
Type          object
People         int64
Latitude     float64
Longitude    float64
dtype: object
```

In [23]:
```
usa_1877_df.sample(5)
```

Out[23]:

	Name	State	Department	Type	People	Latitude	Longitu
835	Missoula	MT	War Department	Other	32	46.862121	-113.9882
502	San Luis Pass	TX	Treasury Department	Customs Service	1	29.079466	-95.1197
33	Beatrice	NE	Department of the Interior	General Land Office Receivers	1	40.265927	-96.7466
650	Omaha	NE	Treasury Department	Internal Revenue_Storekeepers	3	41.252363	-95.9979
623	Omaha	NE	Treasury Department	Internal Revenue_Gaugers	2	41.252363	-95.9979

In [24]:
```
us_map_empty = folium.Map(location=[40, -105], zoom_start=4)
us_map_empty
```

Out[24]:

In [25]:
```
def create_empty_map():
    return folium.Map(location=[40, -105], zoom_start=4)

usa_map = create_empty_map()
usa_map
```

Out[25]:

In [26]:
```
usa_1877_df_locations = usa_1877_df[usa_1877_df['Latitude'].notna()]
print(len(usa_1877_df))
print(len(usa_1877_df_locations))
```

```
903
903
```

In [27]:
```
def create_map_markers(row, map_name):
    folium.Marker(location=[row['Latitude'], row['Longitude']],
                  popup=row['Department']).add_to(map_name)
```

In [28]:
```
# Method 2: Using .apply() to add markers with our function for all rows
# initialize an empty map
usa_map = us_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
usa_1877_df.apply(
    create_map_markers, # The function to apply
    map_name=usa_map, # Additional argument to pass to the function
    axis='columns' # Process row by row instead of column by column
)

usa_map
```

Out[28]:

In [29]:
```
#expanding popup information
def create_popups(row, map_name):
    folium.Marker(location=[row['Latitude'], row['Longitude']],
                  popup=folium.Popup(f"{row['Name'].title()} {row['Departmer

usa_map = us_map_empty

usa_1877_df.apply(
    create_popups,
    map_name=usa_map,
    axis='columns'
)

usa_map
```

Out[29]:

In [64]:
```
us_map_empty = folium.Map(location=[40, -105], zoom_start=4)
us_map_empty
```

Out[64]:

In [66]:
```
def create_circles(row, map_name):
    folium.CircleMarker(location=[row['Latitude'], row['Longitude']],
                        radius=row['People']/10,
                        fill=True,
```

```
                        popup=folium.Popup(f"Post Office: {row['People']}", m
                        tooltip=f"Employees: {row['People']}"
                        ).add_to(map_name)

usa_1877_df.apply(
    create_circles,
    map_name=us_map_empty,
    axis='columns'
)

us_map_empty
```

Out[66]:

In [70]:
```
usa_1877_df.describe()
```

Out[70]:

	People	Latitude	Longitude
count	903.000000	903.000000	903.000000
mean	19.176080	38.974548	-108.959857
std	66.293517	5.508786	10.516498
min	1.000000	25.898333	-124.736600
25%	1.000000	35.584746	-120.409982
50%	1.000000	39.150171	-107.779780
75%	4.000000	42.871109	-97.743061
max	881.000000	48.966377	-93.322350

In [82]:
```
def add_pop_buckets(pop):
   # Create a new column for the salary bucket
   if pop < 3:
       bucket = 'Low Employees'
   elif pop >= 4 and pop < 8:
       bucket = 'Medium Employees'
   elif pop >= 8 and pop < 100:
       bucket = 'High Employees'
   else:
       bucket = 'Very High Employees'
   return bucket
```

In [84]:
```
usa_1877_df['Pop_Bucket'] = usa_1877_df['People'].apply(add_pop_buckets)
usa_1877_df.sample(10)
```

Out[84]:

	Name	State	Department	Type	People	Latitude	Lor
90	Norfolk	NE	Department of the Interior	General Land Office Registers	1	42.032723	-97.
277	Topeka	KS	Judicial	Commissioner	4	39.055824	-95.
463	Helena	MT	Treasury Department	Assay Office	5	46.588371	-112.(
836	Fort Abercrombie	ND	War Department	Outpost	40	46.444722	-96
447	Tyler	TX	Judicial	Court	2	32.351260	-95.
653	San Antonio	TX	Treasury Department	Internal Revenue_Storekeepers	1	29.424122	-98.4
119	Santa Fe	NM	Department of the Interior	Governors and Secretaries of the Territories	2	35.686975	-105.'
294	Carroll	MT	Judicial	Commissioner	1	47.573611	-108.
183	Uintah Agency	UT	Department of the Interior	Office of Indian Affairs	8	40.288010	-109.8
621	Helena	MT	Treasury Department	Internal Revenue_Gaugers	1	46.588371	-112.(

In [86]:
```
def add_marker_sizes(category):
    if category == 'Low Employees':
        return 5
    elif category == 'Medium Employees':
        return 10
    elif category == 'High Employees':
        return 15
    else:
        return 20
```

In [88]:
```
usa_1877_df['Marker_Size'] = usa_1877_df['Pop_Bucket'].apply(add_marker_size
usa_1877_df.head(10)
```

Out[88]:

	Name	State	Department	Type	People	Latitude	Longitude	Pop_Bu
0	Prescott	AZ	Department of the Interior	General Land Office Receivers	1	34.540024	-112.468503	Empl
1	Florence	AZ	Department of the Interior	General Land Office Receivers	1	33.031451	-111.387343	Empl
2	San Francisco	CA	Department of the Interior	General Land Office Receivers	1	37.774929	-122.419416	Empl
3	Marysville	CA	Department of the Interior	General Land Office Receivers	1	39.145725	-121.591355	Empl
4	Humboldt	CA	Department of the Interior	General Land Office Receivers	1	40.745005	-123.869509	Empl
5	Stockton	CA	Department of the Interior	General Land Office Receivers	1	37.957702	-121.290780	Empl
6	Visalia	CA	Department of the Interior	General Land Office Receivers	1	36.330228	-119.292059	Empl
7	Sacramento	CA	Department of the Interior	General Land Office Receivers	1	38.581572	-121.494400	Empl
8	Los Angeles	CA	Department of the Interior	General Land Office Receivers	1	34.052234	-118.243685	Empl
9	Shasta	CA	Department of the Interior	General Land Office Receivers	1	40.598119	-122.490757	Empl

In [96]:

```
def create_bucket_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"Office: {row['Department'].title
                        tooltip=f"Employees: {row['People']}"
                        ).add_to(map_name)


# initialize an empty map
usa_final_map = create_empty_map()

# call our function for each row
usa_1877_df.apply(
    create_bucket_circle_markers, # The function to apply
    map_name=usa_final_map, # Additional argument to pass to the function
    axis='columns' # Process row by row instead of column by column
)

usa_final_map
```

Out[96]:

In [98]:
```
usa_1877_df
```

Out[98]:

	Name	State	Department	Type	People	Latitude	Longitude	Pop_
0	Prescott	AZ	Department of the Interior	General Land Office Receivers	1	34.540024	-112.468503	Em
1	Florence	AZ	Department of the Interior	General Land Office Receivers	1	33.031451	-111.387343	Em
2	San Francisco	CA	Department of the Interior	General Land Office Receivers	1	37.774929	-122.419416	Em
3	Marysville	CA	Department of the Interior	General Land Office Receivers	1	39.145725	-121.591355	Em
4	Humboldt	CA	Department of the Interior	General Land Office Receivers	1	40.745005	-123.869509	Em
...	...	...	...	...	...	...	...	
898	Fort McKinney	WY	War Department	Outpost	158	44.355791	-106.688741	Ve Em
899	Camp Stambaugh	WY	War Department	Outpost	40	41.680278	-108.755556	Em
900	Fort Bridger	WY	War Department	Outpost	100	41.317778	-110.391944	Ve Em
901	Camp Brown	WY	War Department	Outpost	146	42.653889	-109.698611	Ve Em
902	Fort Fetterman	WY	War Department	Outpost	138	42.840278	-105.479722	Ve Em

903 rows × 9 columns

In [108…
```
top_ten_df = usa_1877_df.sort_values(by="People", ascending=False).head(10)
```

In [114…
```
usa_new_map = create_empty_map()
```

In [116…
```
top_ten_df.apply(
    create_circles, # The function to apply
```

```
    map_name=usa_new_map, # Additional argument to pass to the function
    axis='columns' # Process row by row instead of column by column
)

usa_new_map
```

Out[116…

In [121…
```
import plotly.express as px
```

In [127…
```
fig = px.bar(
    top_ten_df,
    x='Name',
    y='People',
    title='Top 10 Government employers (1877)',
    labels={'Name': 'Location', 'People': '# of people'},
    color='People',                    # Color bars by population
    color_continuous_scale='YlOrRd',  # Use a reversed color scale
    template='plotly_dark'      # Use a clean white template
)

# Update layout with additional customizations
fig.update_layout(
    xaxis_title='Location of building',    # Customize x-axis title
    yaxis_title='Number of People',        # Customize y-axis title
    xaxis_tickangle=-45,                   # Rotate x-axis labels 45 degrees
    height=400,                            # Set chart height in pixels
    width=800,                             # Set chart width in pixels
    title_font=dict(size=30,),              # Change title font size
    plot_bgcolor='white',                   # Set plot background color
```

```
    margin=dict(l=80, r=50, t=80, b=80),   # Adjust margins (left, right, to
    showlegend=True,                       # Show the color scale legend
    legend_title_text='Number of People',        # Set legend title
    hoverlabel=dict(                       # Customize hover label appearanc
        bgcolor="black",
        font_size=12,
        font_family="Wingdings")
)

# Display the chart
fig.show()
```

In [130…

```
most_employees = usa_1877_df.groupby("State")["People"].sum()
most_employees
```

Out[130…
```
State
AZ    1303
BC       1
CA    1212
CO     289
ID     621
KS     990
MT    1447
ND     857
NE    1296
NM     892
NV     187
OK    1107
OR     322
SD     930
TX    3643
UT     262
WA     311
WY    1646
Name: People, dtype: int64
```

In [136…
```
most_employees_filt = most_employees.sort_values(ascending=False)
most_employees_filt
```

Out[136…
```
State
TX    3643
WY    1646
MT    1447
AZ    1303
NE    1296
CA    1212
OK    1107
KS     990
SD     930
NM     892
ND     857
ID     621
OR     322
WA     311
CO     289
UT     262
NV     187
BC       1
Name: People, dtype: int64
```

In [138…
```
departments = usa_1877_df.groupby("Department")["People"].sum().sort_values(
departments
```

Out[138…
```
Department
War Department                14940
Treasury Department            1080
Department of the Interior      891
Judicial                        405
Name: People, dtype: int64
```

In [141…
```
locations = usa_1877_df.groupby("State")["Name"].count().sort_values(ascendi
locations
```

Out[141…
```
State
CA    167
TX    138
KS     69
OR     62
NE     61
WA     55
MT     47
AZ     42
NM     41
CO     39
SD     37
WY     34
NV     25
ID     25
ND     22
UT     21
OK     17
BC      1
Name: Name, dtype: int64
```

In [ ]: