Final Project

AUTHOR
Abu Bakar Siddique

PUBLISHED
December 6, 2024

```
import pandas as pd
# Load the NYC civil jobs data set in data frame
df = pd.read_csv("Jobs_NYC_Postings.csv")
df.head()
```

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level
0	634145	DEPT OF HEALTH/MENTAL HYGIENE	External	1	Quality Management Coordinator, Bureau of the ...	CITY RESEARCH SCIENTIST	Non-Competitive-5	21744	02
1	689496	DEPARTMENT OF TRANSPORTATION	External	1	School Safety Inspector - RIS	ASSISTANT TRANSPORTATION SPECI	Competitive-1	22306	00
2	613824	OFFICE OF THE COMPTROLLER	External	1	Senior Attorney â€“ Litigation Unit	EXECUTIVE AGENCY COUNSEL	Non-Competitive-5	95005	M3
3	620326	DEPT OF DESIGN & CONSTRUCTION	Internal	1	CADD Designer	ASSISTANT CIVIL ENGINEER	Competitive-1	20210	00
4	622056	DEPT OF ENVIRONMENT PROTECTION	External	2	Senior Inspector	Associate Air Pollution Inspr	Competitive-1	31316	01

5 rows × 30 columns

```
print(f'Initially the data has {df.shape[0]} records')
```

```
Initially the data has 5527 records
```

```
#### There are multiple job postings because of various reasons. Displaying a few cases w

df[df['Job ID'] == df['Job ID'].value_counts().idxmax()]
```

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level	Job C
4067	687773	DEPARTMENT OF TRANSPORTATION	Internal	1	BOB-Staff Manager	ASSOCIATE STAFF ANALYST	Competitive-1	12627	00	Admin & Hun Resou
4068	687773	DEPARTMENT OF TRANSPORTATION	Internal	1	BOB-Staff Manager	ASSOCIATE STAFF ANALYST	Competitive-1	12627	00	Admin & Hun Resou
5303	687773	DEPARTMENT OF TRANSPORTATION	External	1	BOB-Staff Manager	ASSOCIATE STAFF ANALYST	Competitive-1	12627	00	Admin & Hun Resou

3 rows × 30 columns

```
df[df['Job ID'] == 686510]
```

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level	Job Catego
726	686510	NYC POLICE PENSION FUND	Internal	2	Disability System Representative	CLERICAL AIDE	Competitive-1	10250	00	Administrati & Human Resources
727	686510	NYC POLICE PENSION FUND	Internal	2	Disability System Representative	CLERICAL AIDE	Competitive-1	10250	00	Administrati & Human Resources
1626	686510	NYC POLICE	External	2	Disability System Representative	CLERICAL AIDE	Competitive-1	10250	00	Administrati & Human Resources

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level	Job Catego
		PENSION FUND								

3 rows × 30 columns

```
### Dropping multiple instances of the Job postings

df_unique = df.drop_duplicates(subset=['Job ID'])
```

```
print(f'After removing duplicate postings there are {df_unique.shape[0]} postingss with u
```

```
After removing duplicate postings there are 2812 postingss with unique JOb IDs
```

```
## Reading the dataframe with coordinates information

##Reading the coordinates from the address is a time consume step. I have seprately found
```

```
df_coordinates = pd.read_csv("NYC_addresses_with_coordinates_final.csv")
```

```
#Removing the address column
df_coordinates = df_coordinates[['Job ID', 'latitude', 'longitude']]
```

```
#Removing the duplicate Job IDs
df_coordinates_unique = df_coordinates.drop_duplicates(subset=['Job ID'])
```

```
df_coordinates_unique.describe()
```

	Job ID	latitude	longitude
count	2812.000000	1873.000000	1873.000000
mean	635509.648649	40.712840	-73.500992
std	43104.870558	3.282052	9.918110
min	469953.000000	-37.786861	-122.112585
25%	617993.000000	40.706215	-74.009067
50%	637592.500000	40.715476	-73.999811
75%	681133.500000	40.749583	-73.920374
max	690149.000000	53.350017	152.980057

```
merged_df = pd.merge(df_unique, df_coordinates_unique, on='Job ID', how='outer')
```

```
#The merged dataframe has 2812 records
merged_df.sample()
```

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level	Job Category
1617	640913	DEPT OF DESIGN & CONSTRUCTION	Internal	2	Project Manager	PROJECT MANAGER	Competitive-1	22426	00	Engineerin Architectu & Plannin

1 rows × 32 columns

```
### Drop the job postings with NaN latitude
merged_df.dropna(subset=['latitude'], inplace=True)
merged_df.shape
```

```
(1873, 32)
```

```
merged_df.head()
```

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level	Job Catego
0	469953	HRA/DEPT OF SOCIAL SERVICES	Internal	1	CONTRACT ANALYST	STAFF ANALYST	Competitive-1	12626	02	Administrati & Human Resources Social Services
1	481622	ADMIN FOR CHILDREN'S SVCS	Internal	1	Child Protective Manager	DIRECTOR OF FIELD OPERATIONS (	Non-Competitive-5	95600	M1	Social Services
2	483894	HRA/DEPT OF SOCIAL SERVICES	Internal	1	SENIOR PROJECT MANAGER	COMPUTER SPECIALIST (SOFTWARE)	Competitive-1	13632	03	Technology, Data & Innovation Social Services
3	484513	HRA/DEPT OF SOCIAL	Internal	1	CONTROL CLERK	CLERICAL ASSOCIATE	Competitive-1	10251	03	Social Services

	Job ID	Agency	Posting Type	# Of Positions	Business Title	Civil Service Title	Title Classification	Title Code No	Level	Job Catego
		SERVICES								
4	487203	HRA/DEPT OF SOCIAL SERVICES	Internal	2	UNIT CLERK	CLERICAL ASSOCIATE	Competitive-1	10251	03	Administrati & Human Resources Social Services

5 rows × 32 columns

```
##Identifying distinct keywords splitting  the 'Job Category' values by `,` and `&`
lst_Job_Category = merged_df['Job Category']
lst_Job_Category = list(lst_Job_Category)
lst2 = [i.strip(' ').strip('&').strip(' ') for item in lst_Job_Category for i in item.spl
Job_Categories = sorted(list(set([i.strip(' ') for item in lst2 for i in item.split('&')]
print(Job_Categories)
```

```
['Accounting', 'Administration', 'Analysis', 'Analysis Public Safety', 'Analysis Social
Services', 'Architecture', 'Building Operations', 'Communications', 'Community Programs',
'Community Programs Building Operations', 'Community Programs Communications', 'Community
Programs Engineering', 'Community Programs Finance', 'Community Programs Green Jobs
Building Operations', 'Community Programs Health', 'Community Programs Health Building
Operations', 'Community Programs Health Legal Affairs Public Safety', 'Community Programs
Health Legal Affairs Social Services', 'Community Programs Health Policy', 'Community
Programs Health Public Safety', 'Community Programs Health Social Services', 'Community
Programs Health Technology', 'Community Programs Legal Affairs', 'Community Programs
Legal Affairs Policy', 'Community Programs Legal Affairs Public Safety', 'Community
Programs Legal Affairs Social Services', 'Community Programs Policy', 'Community Programs
Public Safety', 'Community Programs Social Services', 'Community Programs Technology',
'Constituent Services', 'Data', 'Enforcement', 'Enforcement Social Services',
'Engineering', 'Finance', 'Green Jobs', 'Green Jobs Building Operations', 'Green Jobs
Engineering', 'Green Jobs Health Policy', 'Green Jobs Health Public Safety', 'Green Jobs
Policy', 'Green Jobs Public Safety', 'Green Jobs Social Services', 'Green Jobs
Technology', 'Health', 'Health Building Operations', 'Health Legal Affairs', 'Health
Policy', 'Health Public Safety', 'Health Social Services', 'Health Technology', 'Human
Resources', 'Human Resources Building Operations', 'Human Resources Communications',
'Human Resources Constituent Services', 'Human Resources Engineering', 'Human Resources
Finance', 'Human Resources Green Jobs', 'Human Resources Green Jobs Policy', 'Human
Resources Green Jobs Social Services', 'Human Resources Health', 'Human Resources Health
Policy', 'Human Resources Health Public Safety', 'Human Resources Legal Affairs', 'Human
Resources Legal Affairs Public Safety', 'Human Resources Legal Affairs Social Services',
'Human Resources Policy', 'Human Resources Public Safety', 'Human Resources Social
Services', 'Human Resources Technology', 'Innovation', 'Innovation Legal Affairs',
'Innovation Legal Affairs Policy', 'Innovation Legal Affairs Public Safety', 'Innovation
Policy', 'Innovation Public Safety', 'Innovation Social Services', 'Inspections',
'Intergovernmental Affairs', 'Intergovernmental Affairs Engineering', 'Intergovernmental
Affairs Finance', 'Intergovernmental Affairs Health', 'Intergovernmental Affairs Health
Policy', 'Intergovernmental Affairs Legal Affairs', 'Intergovernmental Affairs Legal
Affairs Policy', 'Intergovernmental Affairs Legal Affairs Public Safety',
'Intergovernmental Affairs Legal Affairs Social Services', 'Intergovernmental Affairs
Policy', 'Intergovernmental Affairs Public Safety', 'Intergovernmental Affairs Social
Services', 'Intergovernmental Affairs Technology', 'Legal Affairs', 'Legal Affairs
Policy', 'Legal Affairs Public Safety', 'Legal Affairs Social Services', 'Maintenance',
'Maintenance Policy', 'Maintenance Public Safety', 'Maintenance Social Services', 'Mental
Health Health Legal Affairs', 'Planning', 'Planning Building Operations', 'Planning
Finance', 'Planning Health', 'Planning Health Policy', 'Planning Policy', 'Planning
Public Safety', 'Planning Technology', 'Policy', 'Procurement', 'Procurement Building
Operations', 'Procurement Health', 'Procurement Legal Affairs', 'Procurement Legal
Affairs Policy', 'Procurement Legal Affairs Public Safety', 'Procurement Policy',
'Procurement Public Safety', 'Procurement Social Services', 'Procurement Technology',
'Public Safety', 'Research', 'Social Services', 'Technology']
```

```
len(Job_Categories)
```

```
124
```

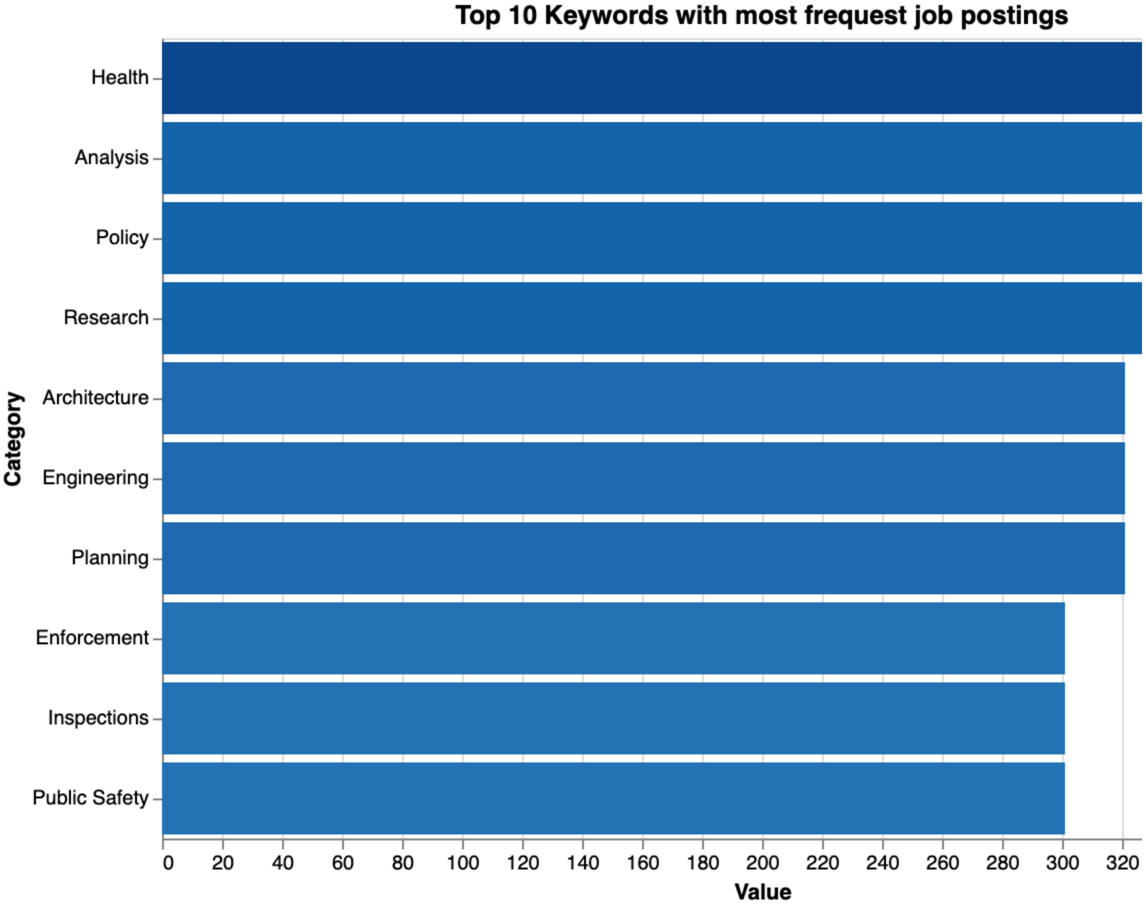
```
## Find number of Job postings for each 'Job Category' keyword
# Find occurrences for each substring
counts = {substring: merged_df['Job Category'].str.contains(substring).sum() for substrin
sorted_counts = dict(sorted(counts.items(), key=lambda item: item[1], reverse=True))
print(sorted_counts)
```

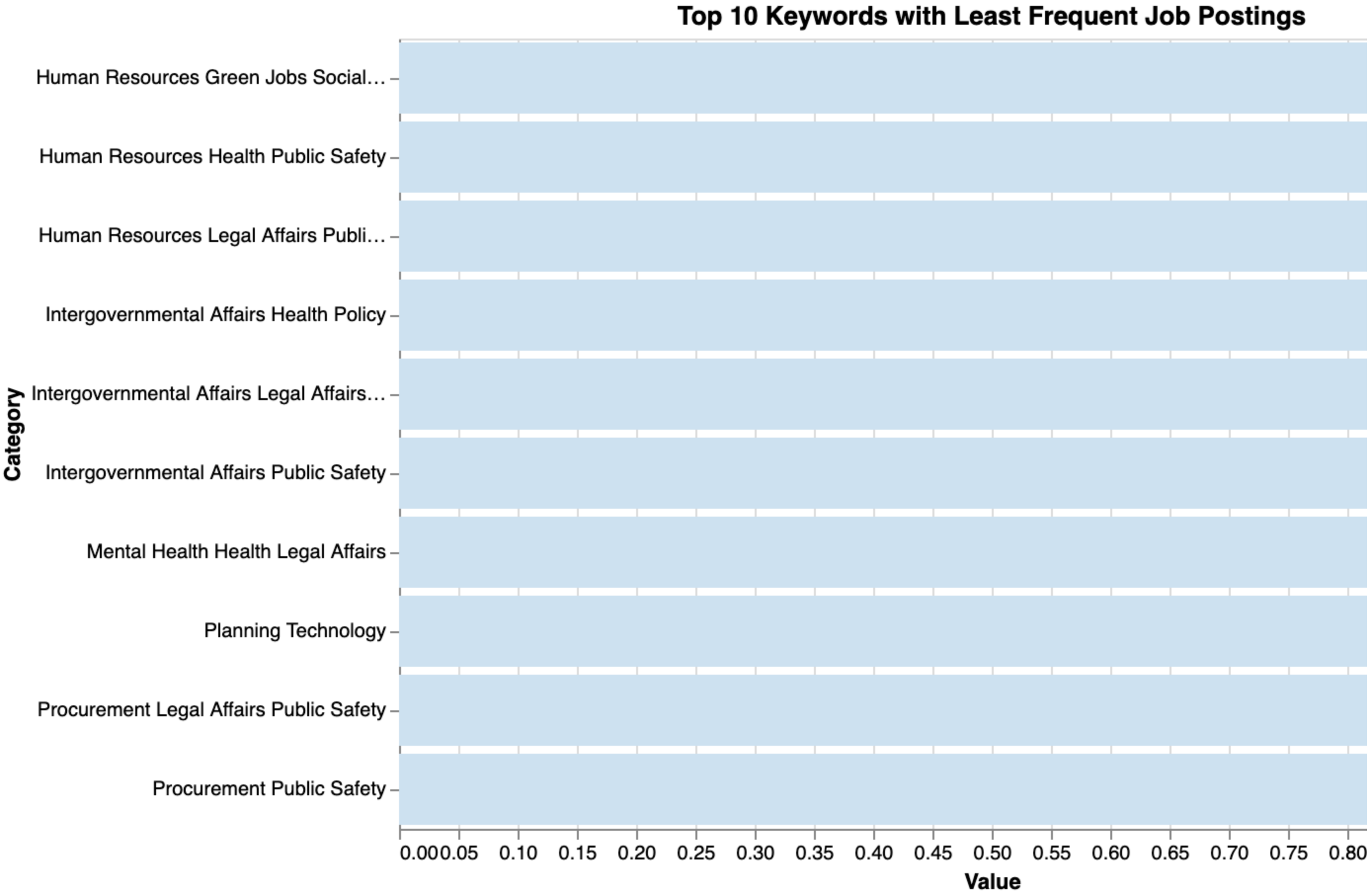
```
{'Health': np.int64(393), 'Analysis': np.int64(336), 'Policy': np.int64(336), 'Research':
np.int64(336), 'Architecture': np.int64(321), 'Engineering': np.int64(321), 'Planning':
np.int64(321), 'Enforcement': np.int64(301), 'Inspections': np.int64(301), 'Public
Safety': np.int64(301), 'Social Services': np.int64(294), 'Community Programs':
np.int64(261), 'Constituent Services': np.int64(261), 'Administration': np.int64(241),
'Human Resources': np.int64(241), 'Legal Affairs': np.int64(214), 'Data': np.int64(181),
'Innovation': np.int64(181), 'Technology': np.int64(181), 'Accounting': np.int64(153),
'Finance': np.int64(153), 'Procurement': np.int64(153), 'Health Policy': np.int64(120),
'Community Programs Health': np.int64(100), 'Building Operations': np.int64(87),
'Maintenance': np.int64(87), 'Communications': np.int64(85), 'Intergovernmental Affairs':
np.int64(85), 'Planning Public Safety': np.int64(73), 'Community Programs Health Policy':
np.int64(67), 'Human Resources Social Services': np.int64(56), 'Legal Affairs Public
Safety': np.int64(50), 'Procurement Policy': np.int64(45), 'Analysis Public Safety':
np.int64(38), 'Green Jobs': np.int64(37), 'Health Public Safety': np.int64(35), 'Planning
Policy': np.int64(34), 'Community Programs Communications': np.int64(32), 'Innovation
Social Services': np.int64(23), 'Innovation Policy': np.int64(21), 'Enforcement Social
Services': np.int64(19), 'Health Technology': np.int64(19), 'Legal Affairs Policy':
np.int64(19), 'Analysis Social Services': np.int64(18), 'Human Resources Finance':
np.int64(17), 'Human Resources Constituent Services': np.int64(16), 'Intergovernmental
Affairs Social Services': np.int64(16), 'Community Programs Social Services':
```

```
np.int64(15), 'Green Jobs Engineering': np.int64(14), 'Human Resources Policy':
np.int64(14), 'Intergovernmental Affairs Policy': np.int64(14), 'Maintenance Public
Safety': np.int64(13), 'Community Programs Engineering': np.int64(12), 'Legal Affairs
Social Services': np.int64(12), 'Human Resources Legal Affairs': np.int64(11),
'Innovation Legal Affairs': np.int64(11), 'Human Resources Communications': np.int64(10),
'Intergovernmental Affairs Technology': np.int64(10), 'Health Legal Affairs':
np.int64(8), 'Health Social Services': np.int64(8), 'Community Programs Legal Affairs':
np.int64(7), 'Community Programs Policy': np.int64(7), 'Innovation Legal Affairs Public
Safety': np.int64(7), 'Intergovernmental Affairs Legal Affairs': np.int64(7), 'Innovation
Public Safety': np.int64(6), 'Procurement Legal Affairs': np.int64(6), 'Green Jobs
Policy': np.int64(5), 'Maintenance Policy': np.int64(5), 'Planning Building Operations':
np.int64(5), 'Community Programs Finance': np.int64(4), 'Community Programs Legal Affairs
Public Safety': np.int64(4), 'Community Programs Public Safety': np.int64(4), 'Green Jobs
Health Public Safety': np.int64(4), 'Health Building Operations': np.int64(4), 'Human
Resources Green Jobs': np.int64(4), 'Human Resources Health': np.int64(4), 'Human
Resources Technology': np.int64(4), 'Intergovernmental Affairs Finance': np.int64(4),
'Planning Health': np.int64(4), 'Procurement Legal Affairs Policy': np.int64(4),
'Community Programs Health Legal Affairs Public Safety': np.int64(3), 'Green Jobs
Building Operations': np.int64(3), 'Green Jobs Social Services': np.int64(3), 'Human
Resources Engineering': np.int64(3), 'Human Resources Legal Affairs Social Services':
np.int64(3), 'Intergovernmental Affairs Engineering': np.int64(3), 'Intergovernmental
Affairs Health': np.int64(3), 'Intergovernmental Affairs Legal Affairs Policy':
np.int64(3), 'Procurement Building Operations': np.int64(3), 'Procurement Health':
np.int64(3), 'Procurement Technology': np.int64(3), 'Community Programs Green Jobs
Building Operations': np.int64(2), 'Human Resources Building Operations': np.int64(2),
'Human Resources Green Jobs Policy': np.int64(2), 'Human Resources Health Policy':
np.int64(2), 'Human Resources Public Safety': np.int64(2), 'Innovation Legal Affairs
Policy': np.int64(2), 'Intergovernmental Affairs Legal Affairs Public Safety':
np.int64(2), 'Maintenance Social Services': np.int64(2), 'Planning Finance': np.int64(2),
'Planning Health Policy': np.int64(2), 'Procurement Social Services': np.int64(2),
'Community Programs Building Operations': np.int64(1), 'Community Programs Health
Building Operations': np.int64(1), 'Community Programs Health Legal Affairs Social
Services': np.int64(1), 'Community Programs Health Public Safety': np.int64(1),
'Community Programs Health Social Services': np.int64(1), 'Community Programs Health
Technology': np.int64(1), 'Community Programs Legal Affairs Policy': np.int64(1),
'Community Programs Legal Affairs Social Services': np.int64(1), 'Community Programs
Technology': np.int64(1), 'Green Jobs Health Policy': np.int64(1), 'Green Jobs Public
Safety': np.int64(1), 'Green Jobs Technology': np.int64(1), 'Human Resources Green Jobs
Social Services': np.int64(1), 'Human Resources Health Public Safety': np.int64(1),
'Human Resources Legal Affairs Public Safety': np.int64(1), 'Intergovernmental Affairs
Health Policy': np.int64(1), 'Intergovernmental Affairs Legal Affairs Social Services':
np.int64(1), 'Intergovernmental Affairs Public Safety': np.int64(1), 'Mental Health
Health Legal Affairs': np.int64(1), 'Planning Technology': np.int64(1), 'Procurement
Legal Affairs Public Safety': np.int64(1), 'Procurement Public Safety': np.int64(1)}
```

```python
import altair as alt

# Convert to DataFrame
df_t = pd.DataFrame(list(sorted_counts.items()), columns=["Category", "Value"])
```

```python
# Sort by values
df_sorted = df_t.sort_values(by="Value", ascending=False)

# Top 10 largest values
top_10 = df_sorted.head(10)

# Bottom 10 smallest values
bottom_10 = df_sorted.tail(10)

# Visualization for top 10
top_10_chart = alt.Chart(top_10).mark_bar().encode(
    x=alt.X("Value:Q", title="Value"),
    y=alt.Y("Category:N", sort="-x", title="Category"),
    color=alt.Color("Value:Q", scale=alt.Scale(scheme="blues"), title="Value"),
    tooltip=["Category", "Value"]
).properties(
    title="Top 10 Keywords with most frequest job postings",
    width=600,
    height=400
)

# Visualization for bottom 10
bottom_10_chart = alt.Chart(bottom_10).mark_bar().encode(
    x=alt.X("Value:Q", title="Value"),
    y=alt.Y("Category:N", sort="x", title="Category"),
    color=alt.Color("Value:Q", scale=alt.Scale(scheme="reds"), title="Value"),
    tooltip=["Category", "Value"]
).properties(
    title="Top 10 Keywords with Least Frequent Job Postings",
    width=600,
    height=400
)

# Display the charts
(top_10_chart & bottom_10_chart).show()
```


Top 10 Keywords with most frequest job postings
Category
Health
Analysis
Policy
Research
Architecture
Engineering
Planning
Enforcement
Inspections
Public Safety
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
Value

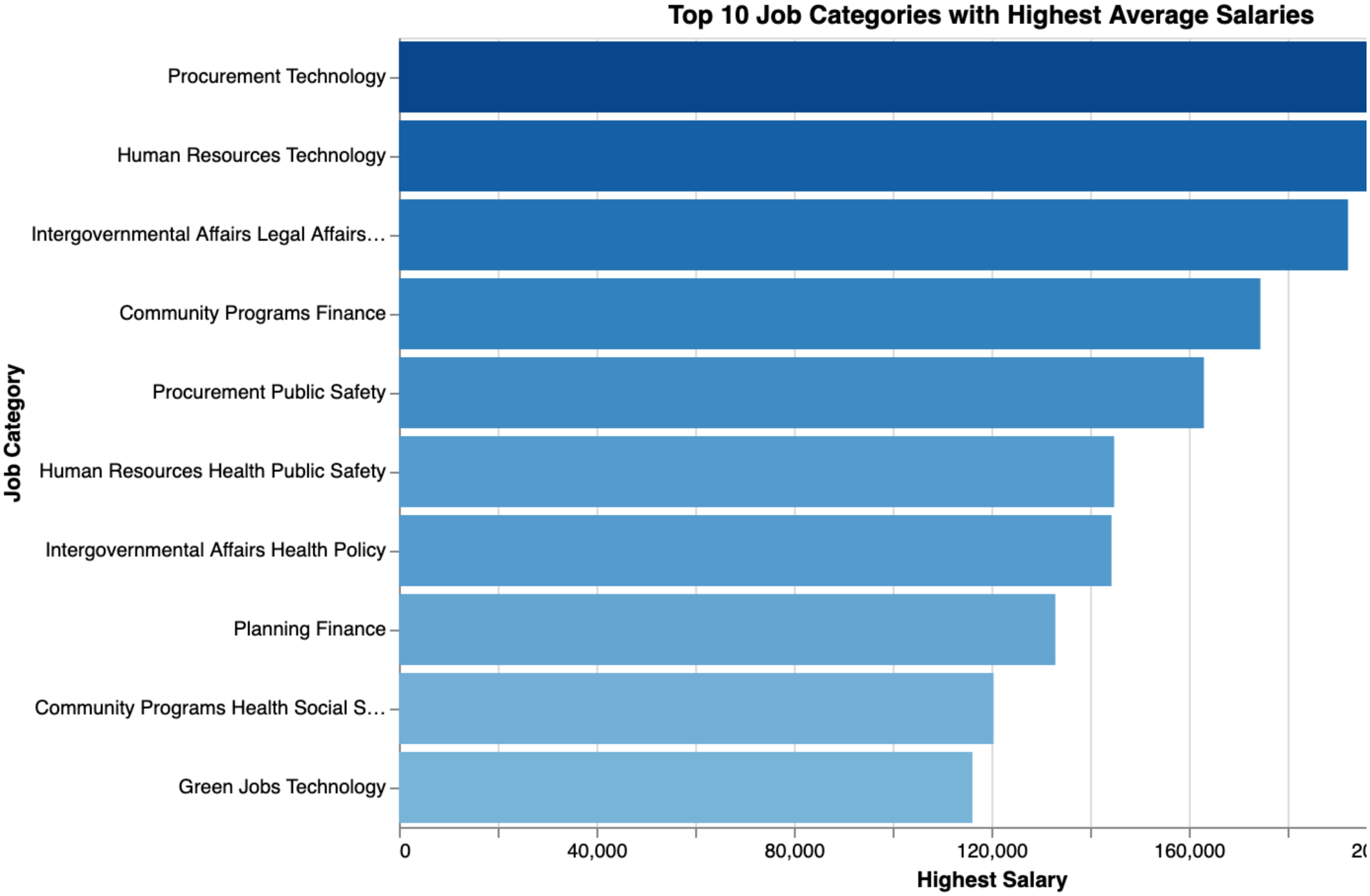

Top 10 Keywords with Least Frequent Job Postings
Category
Human Resources Green Jobs Social…
Human Resources Health Public Safety
Human Resources Legal Affairs Publi…
Intergovernmental Affairs Health Policy
Intergovernmental Affairs Legal Affairs…
Intergovernmental Affairs Public Safety
Mental Health Health Legal Affairs
Planning Technology
Procurement Legal Affairs Public Safety
Procurement Public Safety
0.000.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
0.75
0.80
Value

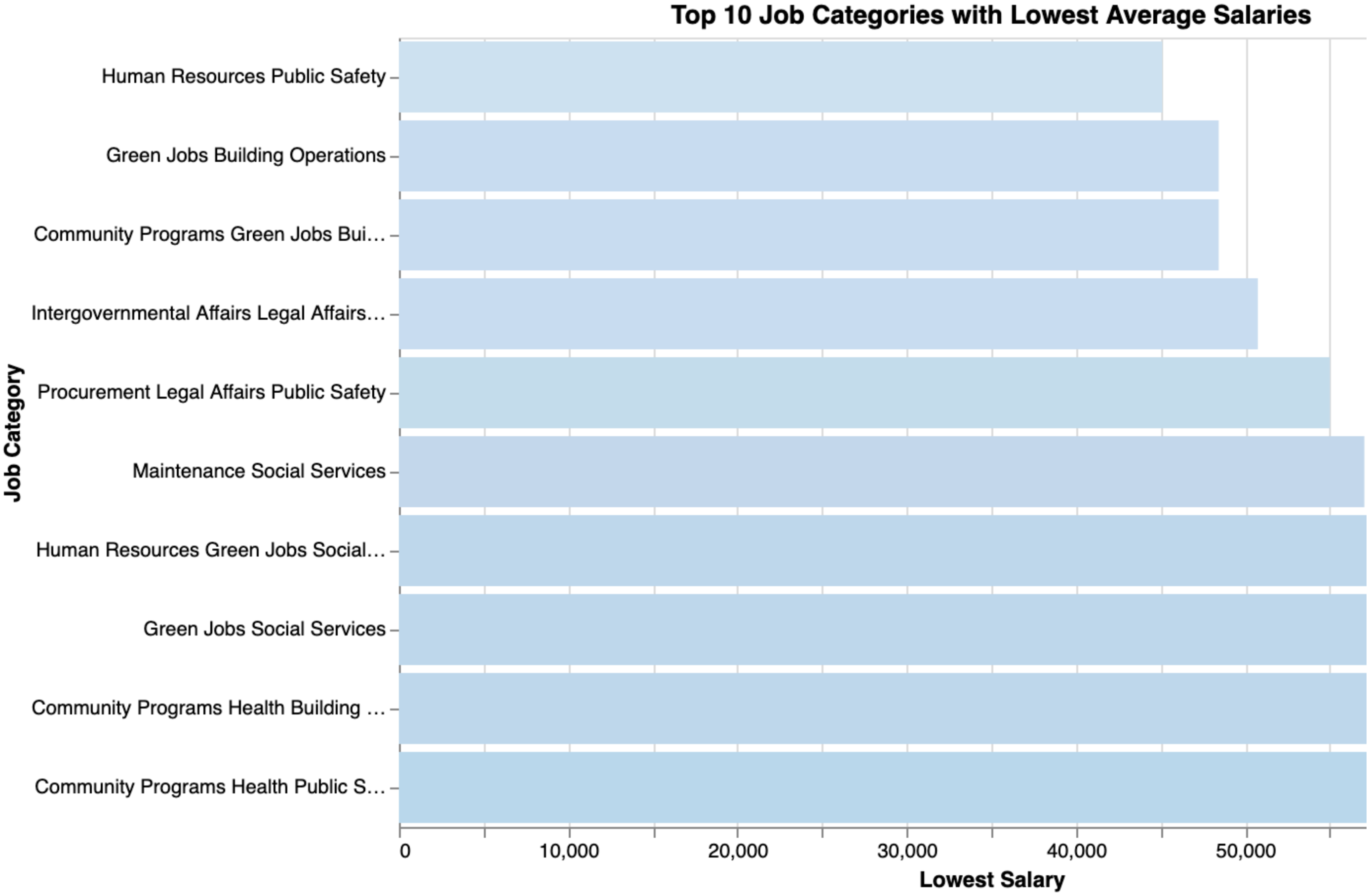
```
# Assuming merged_df is already loaded and contains the required data

# Convert Salary Range columns to numeric
merged_df['Salary Range From'] = pd.to_numeric(merged_df['Salary Range From'], errors='co
merged_df['Salary Range To'] = pd.to_numeric(merged_df['Salary Range To'], errors='coerce

# Calculate the average salary for each job posting
merged_df['Average Salary'] = (merged_df['Salary Range From'] + merged_df['Salary Range T

# Initialize a dictionary to store the results
salary_summary = {}

# Iterate through the keys in the data_dict
for key in sorted_counts.keys():
    # Filter rows where the 'Job Category' contains the key as a substring
    filtered_df = merged_df[merged_df['Job Category'].str.contains(key, case=False, na=Fa

    # Calculate highest, lowest, and average salaries for this key
    highest_salary = filtered_df['Average Salary'].max()
    lowest_salary = filtered_df['Average Salary'].min()
    average_salary = filtered_df['Average Salary'].mean()

    # Store the results in the dictionary
    salary_summary[key] = {
        'Highest Salary': highest_salary,
        'Lowest Salary': lowest_salary,
        'Average Salary': average_salary
    }

# Convert the results to a DataFrame for better readability
salary_summary_df = pd.DataFrame.from_dict(salary_summary, orient='index')
salary_summary_df.reset_index(inplace=True)
salary_summary_df.rename(columns={'index': 'Job Category'}, inplace=True)

salary_summary_df.head()  # Displaying first few rows of the results
```

	Job Category	Highest Salary	Lowest Salary	Average Salary
0	Health	210000.0	17.70	68671.714627
1	Analysis	236649.0	17.25	85485.854131
2	Policy	236649.0	17.25	85485.854131
3	Research	236649.0	17.25	85485.854131
4	Architecture	163063.0	16.25	87861.513172

```
# Sort the DataFrame by 'Highest Salary' and 'Lowest Salary'
top_10_highest_paid = salary_summary_df.sort_values(by='Average Salary', ascending=False)
top_10_lowest_paid = salary_summary_df.sort_values(by='Highest Salary', ascending=True).h
```

```python
# Create a bar chart for the top 10 highest paid job categories
highest_paid_chart = alt.Chart(top_10_highest_paid).mark_bar().encode(
    x=alt.X('Highest Salary:Q', title='Highest Salary'),
    y=alt.Y('Job Category:N', sort='-x', title='Job Category'),
    color=alt.Color('Highest Salary:Q', scale=alt.Scale(scheme='blues')),
    tooltip=['Job Category', 'Highest Salary', 'Lowest Salary', 'Average Salary']
).properties(
    title='Top 10 Job Categories with Highest Average Salaries',
    width=600,
    height=400
)

# Create a bar chart for the top 10 lowest paid job categories
lowest_paid_chart = alt.Chart(top_10_lowest_paid).mark_bar().encode(
    x=alt.X('Lowest Salary:Q', title='Lowest Salary'),
    y=alt.Y('Job Category:N', sort='x', title='Job Category'),
    color=alt.Color('Lowest Salary:Q', scale=alt.Scale(scheme='reds')),
    tooltip=['Job Category', 'Highest Salary', 'Lowest Salary', 'Average Salary']
).properties(
    title='Top 10 Job Categories with Lowest Average Salaries',
    width=600,
    height=400
)

# Display the charts
(highest_paid_chart & lowest_paid_chart).show()
```

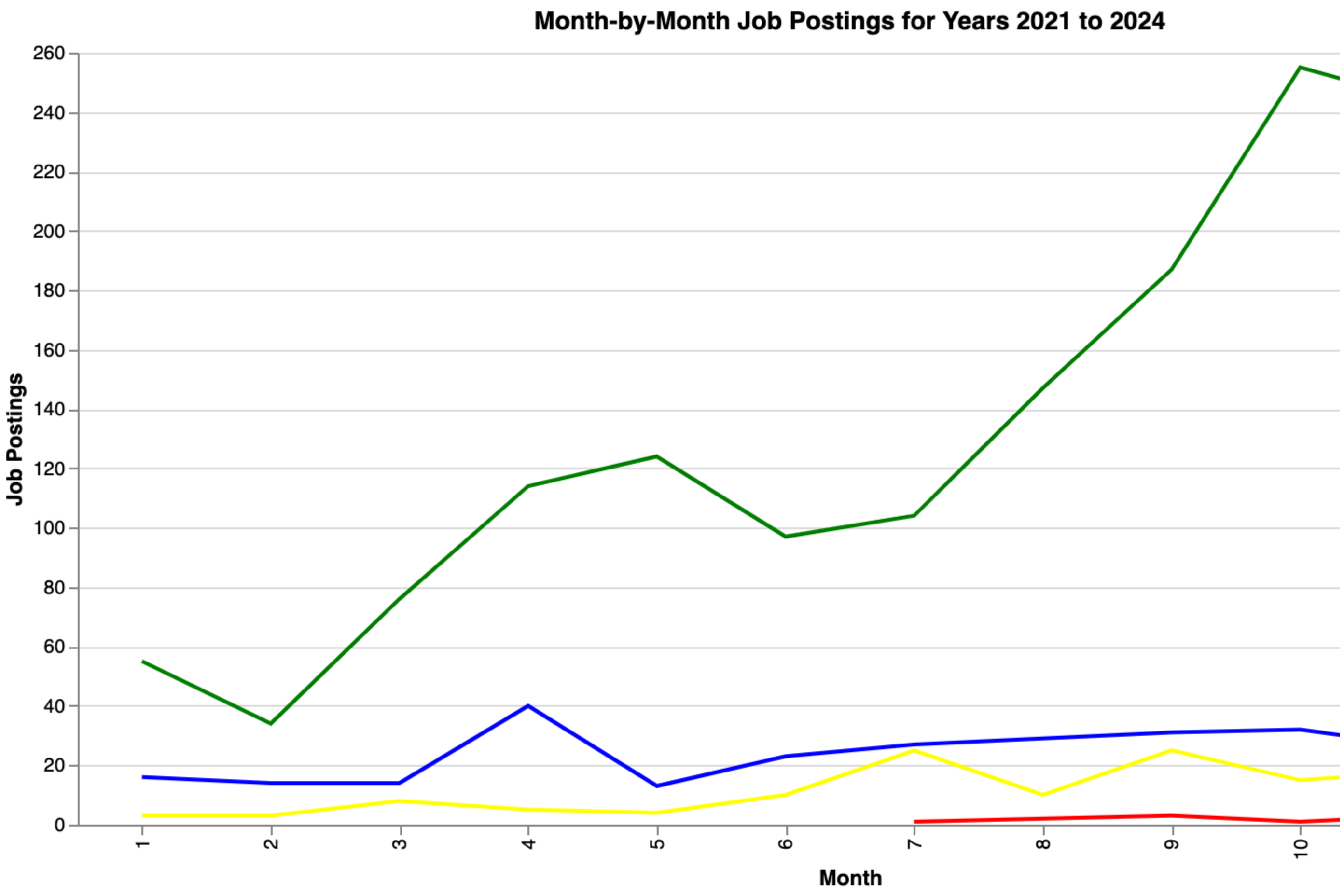

Top 10 Job Categories with Highest Average Salaries
Job Category
Procurement Technology
Human Resources Technology
Intergovernmental Affairs Legal Affairs…
Community Programs Finance
Procurement Public Safety
Human Resources Health Public Safety
Intergovernmental Affairs Health Policy
Planning Finance
Community Programs Health Social S…
Green Jobs Technology
0
40,000
80,000
120,000
160,000
20
Highest Salary


Top 10 Job Categories with Lowest Average Salaries
Job Category
Human Resources Public Safety
Green Jobs Building Operations
Community Programs Green Jobs Bui…
Intergovernmental Affairs Legal Affairs…
Procurement Legal Affairs Public Safety
Maintenance Social Services
Human Resources Green Jobs Social…
Green Jobs Social Services
Community Programs Health Building …
Community Programs Health Public S…
0
10,000
20,000
30,000
40,000
50,000
Lowest Salary

```
salary_summary_df.sample(15)
```

	Job Category	Highest Salary	Lowest Salary	Average Salary
2	Policy	236649.0	17.25000	85485.854131
68	Planning Building Operations	86850.0	40875.00000	65697.600000
80	Community Programs Health Legal Affairs Public...	175000.0	55825.00000	101112.000000
86	Intergovernmental Affairs Health	144346.0	93033.50000	110137.666667
48	Green Jobs Engineering	148676.5	60536.50000	85288.214286
6	Planning	163063.0	16.25000	87861.513172
50	Intergovernmental Affairs Policy	170000.0	22.84425	85657.524589
121	Planning Technology	91354.5	91354.50000	91354.500000
45	Human Resources Constituent Services	174500.0	40875.00000	96109.187500
4	Architecture	163063.0	16.25000	87861.513172
111	Green Jobs Health Policy	79769.5	79769.50000	79769.500000
34	Green Jobs	148676.5	44601.00000	77739.283784
67	Maintenance Policy	126196.0	40875.00000	81597.100000
31	Legal Affairs Public Safety	192250.0	40866.00000	92467.240000
78	Planning Health	110117.0	555.84000	71050.835000

```
# Ensure 'Posting Date' column is in datetime format
merged_df['Posting Date'] = pd.to_datetime(merged_df['Posting Date'], errors='coerce')

# Extract year and month from 'Posting Date'
merged_df['Year'] = merged_df['Posting Date'].dt.year
merged_df['Month'] = merged_df['Posting Date'].dt.month

# Group data by year and month to count job postings
monthly_job_postings = (
    merged_df.groupby(['Year', 'Month'])
    .size()
    .reset_index(name='Job Count')
)

# Define custom colors for years
color_scale = alt.Scale(
    domain=[2024, 2023, 2022, 2021],
    range=["green", "blue", "yellow", "red"]
)

# Create an interactive line chart with custom colors
chart = alt.Chart(monthly_job_postings).mark_line().encode(
    x=alt.X('Month:O', title='Month', sort=list(range(1, 13))),
    y=alt.Y('Job Count:Q', title='Job Postings'),
    color=alt.Color('Year:N', scale=color_scale, title='Year'),
    tooltip=['Year', 'Month', 'Job Count']
).interactive().properties(
    title='Month-by-Month Job Postings for Years 2021 to 2024',
    width=800,
    height=400
)

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

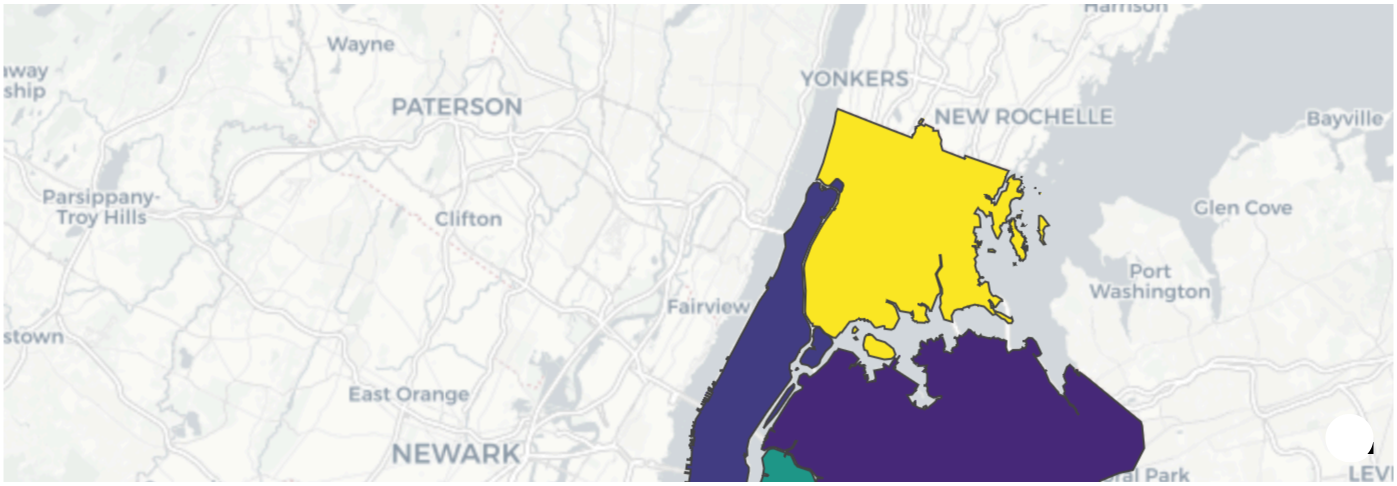


```
merged_df.to_csv('merged_df.csv')
```

```
import pandas as pd
import plotly.express as px
import geopandas as gpd

# Load NYC job data
file_path = 'merged_df.csv'
jobs_data = pd.read_csv(file_path)

# Load NYC borough boundaries (GeoJSON file)
nyc_boroughs_url = "https://data.cityofnewyork.us/resource/7t3b-ywvw.geojson"
```

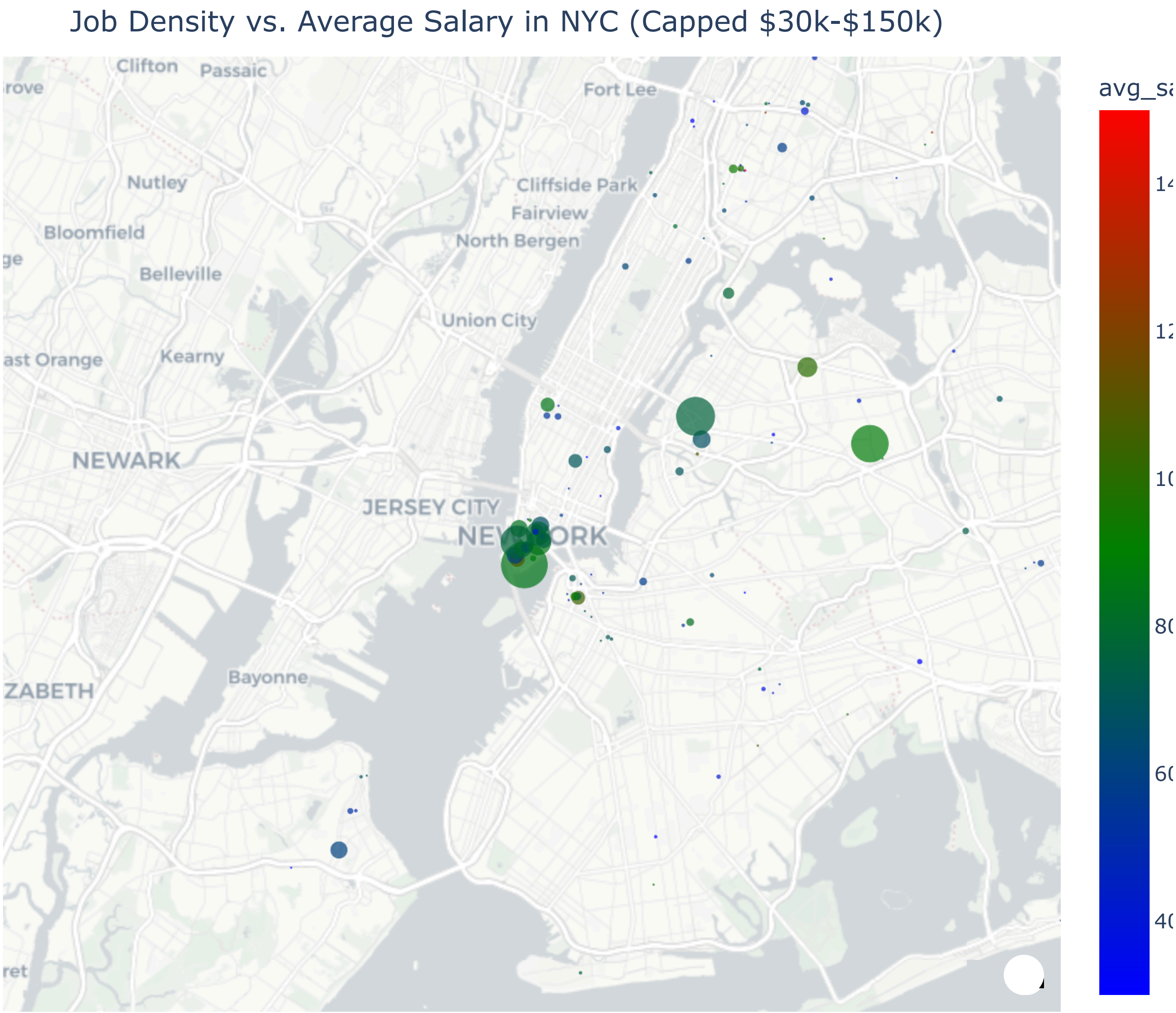
```
boroughs = gpd.read_file(nyc_boroughs_url)

# Extract borough from job locations (assuming 'Work Location' column exists)
# Simplified example if you have borough info directly in your dataset
jobs_data['Borough'] = jobs_data['Work Location'].str.extract(r'(Manhattan|Brooklyn|Queen

# Aggregate job counts by borough
borough_job_counts = jobs_data['Borough'].value_counts().reset_index()
borough_job_counts.columns = ['Borough', 'Job Count']

# Merge job data with borough GeoJSON
boroughs['Borough'] = boroughs['boro_name']
merged = boroughs.merge(borough_job_counts, on='Borough', how='left')
merged['Job Count'] = merged['Job Count'].fillna(0)  # Fill NaN values with 0 for visuali

# Create the choropleth map
fig = px.choropleth_mapbox(
    merged,
    geojson=merged.geometry,
    locations=merged.index,
    color='Job Count',
    hover_name='Borough',
    mapbox_style="carto-positron",
    center={"lat": 40.7128, "lon": -74.0060},
    zoom=9,
    title="Job Density by Borough in NYC",
    color_continuous_scale="Viridis"
)

# Adjust layout
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r": 0, "t": 30, "l": 0, "b": 0})

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

Job Density by Borough in NYC



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

# Load the data
file_path = 'merged_df.csv'
data = pd.read_csv(file_path)

# Aggregate job density and average salary by location
location_data = data.groupby(['latitude', 'longitude']).agg(
    job_density=('Job ID', 'count'),  # Counting Job IDs to get density
    avg_salary=('Average Salary', 'mean')  # Mean salary per location
).reset_index()


# Cap the salary range between $30,000 and $150,000
location_data['avg_salary_capped'] = location_data['avg_salary'].clip(lower=30000, upper=

# Create the map plot with the capped salary range
fig = px.scatter_mapbox(
    location_data,
    lat="latitude",
    lon="longitude",
    size="job_density",
    color="avg_salary_capped",
    color_continuous_scale=[
        (0.0, "blue"),   # Low salaries
        (0.5, "green"),  # Midpoint
        (1.0, "red")     # High salaries
    ],
    size_max=20,
    zoom=10,
    center={"lat": 40.7128, "lon": -74.0060},  # NYC coordinates
    height=600,
    title="Job Density vs. Average Salary in NYC (Capped $30k-$150k)"
)

# Set the map style and layout margins
```

```
fig.update_layout(mapbox_style="carto-positron")
fig.update_layout(margin={"r": 0, "t": 30, "l": 0, "b": 0})

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



Job Density vs. Average Salary in NYC (Capped $30k-$150k)