Integrating CSV files with geojson files.

In [1]:
```
# import necessary libraries.
import geopandas as gpd
import numpy as np
import pandas as pd
import os
print("Imports successful!")
```

```
Imports successful!
```

In [2]:
```
print(os.getcwd()) # check at the working directory.
```

```
C:\Users\user\Documents\Kenya Secondary Schools Enrollment
```

In [3]:
```
# Load attribute file
attr_file = pd.read_csv(f"{os.getcwd()}//Data//SecondarySchoolsEnrollment.csv")
attr_file.tail()
```

Out[3]:

	County	Number of Students Selected	Number of Students Transitioned	Number of Students not yet transitioned	Total	% Transition Rate	% Failed Transition
42	Turkana	11674	11612	62	11674	99.468905	0.531095
43	Uasin Gishu	39434	34418	5016	39434	87.280012	12.719988
44	Vihiga	21874	19841	2033	21874	90.705861	9.294139
45	Wajir	8070	7908	162	8070	97.992565	2.007435
46	West Pokot	19525	16773	2752	19525	85.905250	14.094750

In [4]:
```
# Load vector file
vect_file = gpd.read_file(f"{os.getcwd()}//Data//kenya_counties.GeoJSON")
vect_file.tail()
```

Out[4]:

	GID_0	Name	GID_1	County	Type	County_No	geometry
42	KEN	Kenya	KEN.5_1	Elgeyo Marakwet	County	28	MULTIPOLYGON (((35.58806 0.17453, 35.58784 0.1...
43	KEN	Kenya	KEN.6_1	Embu	County	14	MULTIPOLYGON (((37.30928 -0.14896, 37.30914 -0...
44	KEN	Kenya	KEN.7_1	Garissa	County	7	MULTIPOLYGON (((40.43329 -1.93512, 40.41827 -1...
45	KEN	Kenya	KEN.8_1	Homa Bay	County	43	MULTIPOLYGON (((34.48337 -0.79247, 34.48297 -0...
46	KEN	Kenya	KEN.9_1	Isiolo	County	11	MULTIPOLYGON (((38.56964 -0.01509, 38.56928 -0...

In [5]:
```
# Check at the datatype of both files
type(vect_file)
type(attr_file)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

In [6]:
```
# Build a function to merge the two files
def attr_merge():
    """
    This function merges two files, that is spatial and
    aspatial. It is a fuction for attribute join only
    """
    attr_file = pd.read_csv(f"{os.getcwd()}//Data//SecondarySchoolsEnrollment.csv")
    vect_file = gpd.read_file(f"{os.getcwd()}//Data//kenya_counties.GeoJSON") # bre
    kenya_Sch_trans = attr_file.merge(vect_file, on='County')
    return kenya_Sch_trans
merged = attr_merge()
```

In [7]:
```
merged.tail()
```

Out[7]:

	County	Number of Students Selected	Number of Students Transitioned	Number of Students not yet transitioned	Total	% Transition Rate	% Failed Transition	GID_0	Name	
42	Turkana	11674	11612	62	11674	99.468905	0.531095	KEN	Kenya	KEI
43	Uasin Gishu	39434	34418	5016	39434	87.280012	12.719988	KEN	Kenya	KEI
44	Vihiga	21874	19841	2033	21874	90.705861	9.294139	KEN	Kenya	KEI
45	Wajir	8070	7908	162	8070	97.992565	2.007435	KEN	Kenya	KEI
46	West Pokot	19525	16773	2752	19525	85.905250	14.094750	KEN	Kenya	KEI

In [8]:
```
merged.shape
```

Out[8]:
```
(47, 13)
```

In [9]:
```
type(merged) # checking datatype of merged files
```

Out[9]:
```
pandas.core.frame.DataFrame
```

Writing as a GeoJson File.

In [10]:
```
# A function of writing it to a file
def writeFile():
    """
    This function write the merged files as
    a Geojson file
    """
```

```
    #converting to geodataframe.
    gdf = gpd.GeoDataFrame(merged, geometry='geometry')
    # write useing geojson driver
    return gdf.to_file("KC_SCH_Enrol.geojson", driver='GeoJSON')


writtenFile = writeFile() # Calling the function
```

Developing Kenyan Counties Form One (2024) Secondary Schools Transition Dashboard

A picture is worth a thousand words. Data visuals disclose many information that tabular data can't. A visualized plot provides just as much value. It does this by providing a different look at tabular data, perhaps in the form of simple line charts, histogram distribution and more elaborate pivot charts. In this section, I'm developing an interactive web dashboard, that is user friendly. But wait, before this dashboard is developed, I need to note the following:

```
i. I need to perform an Exploratory Data Analysis. This will enable
me to gain data understanding.
ii.I need to identify key metrics for tracking what matters such as
County with the highest transion rate.
iii.I need to decide on charts to best visualize key metrics
iv.I need to group related metrics together.
v.I need to use clear and concise labels to describe my metrics.
```

Exploratory Data Analysis (EDA).

EDA is an iterative process of data understanding that entails asking and answering questions through investigative work of analyzing the data. In essence my dashboard starts as a blank canvas and EDA provides a pragmatic approach for coming up with compelling data visuals that tells a story.

In [11]:
```
# import necessary library.
import geopandas as gpd
import os
import pandas as pd

# Load in the file
in_file = gpd.read_file(f"{os.getcwd()}//KC_SCH_Enrol.GeoJSON")
in_file.head()
```

Out[11]:

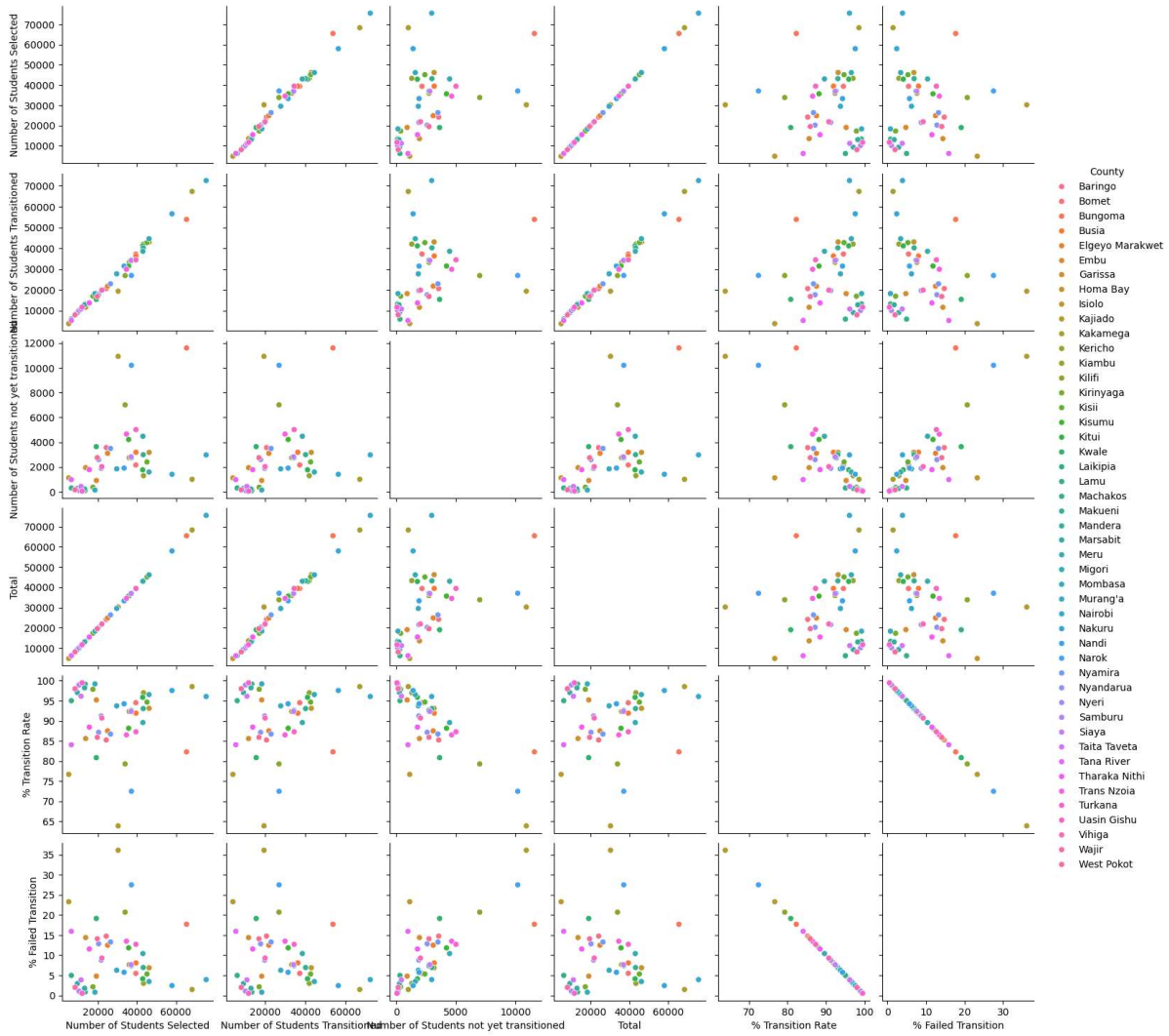
	County	Number of Students Selected	Number of Students Transitioned	Number of Students not yet transitioned	Total	% Transition Rate	% Failed Transition	GID_0	Name	C
0	Baringo	24146	20581	3565	24146	85.235650	14.764350	KEN	Kenya	KE
1	Bomet	39383	37218	2165	39383	94.502704	5.497296	KEN	Kenya	KE
2	Bungoma	65472	53871	11601	65472	82.280975	17.719025	KEN	Kenya	KE
3	Busia	39444	36254	3190	39444	91.912585	8.087415	KEN	Kenya	KE
4	Elgeyo Marakwet	24854	21747	3107	24854	87.498994	12.501006	KEN	Kenya	KE

In [12]:

```
# Checking at the shape of the tabular data
print(f"The data has a dimension of {in_file.shape}, rows and columns respectively.

# Descriptive statitics
print(f"Below are the descriptive statistics of the data: \n{in_file.describe()}")
```

```
The data has a dimension of (47, 13), rows and columns respectively.

Below are the descriptive statistics of the data:
       Number of Students Selected  Number of Students Transitioned  \
count                     47.000000                        47.000000
mean                   29669.021277                     26982.659574
std                    17013.147263                     16058.727845
min                     4834.000000                      3707.000000
25%                    16361.500000                     14515.500000
50%                    29532.000000                     22857.000000
75%                    39439.000000                     36736.000000
max                    75539.000000                     72562.000000

       Number of Students not yet transitioned         Total  \
count                                47.000000     47.000000
mean                               2686.382979  29669.042553
std                                2647.944901  17013.164164
min                                  62.000000   4834.000000
25%                                 997.000000  16361.500000
50%                                2033.000000  29532.000000
75%                                3185.500000  39439.000000
max                               11601.000000  75539.000000

       % Transition Rate  % Failed Transition
count          47.000000            47.000000
mean           90.804521             9.195529
std             7.546623             7.546589
min            63.867853             0.531095
25%            86.936425             3.659932
50%            92.603391             7.396609
75%            96.340068            13.063575
max            99.468905            36.132147
```

In [13]:
```
# Distribution plots
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(in_file, hue="County")
plt.show()
```



In [14]:
```
import folium

data = in_file

m = folium.Map(location=(0.18, 37.91), zoom_start=6, tiles="cartodb positron")
folium.GeoJson(in_file).add_to(m)
m
```

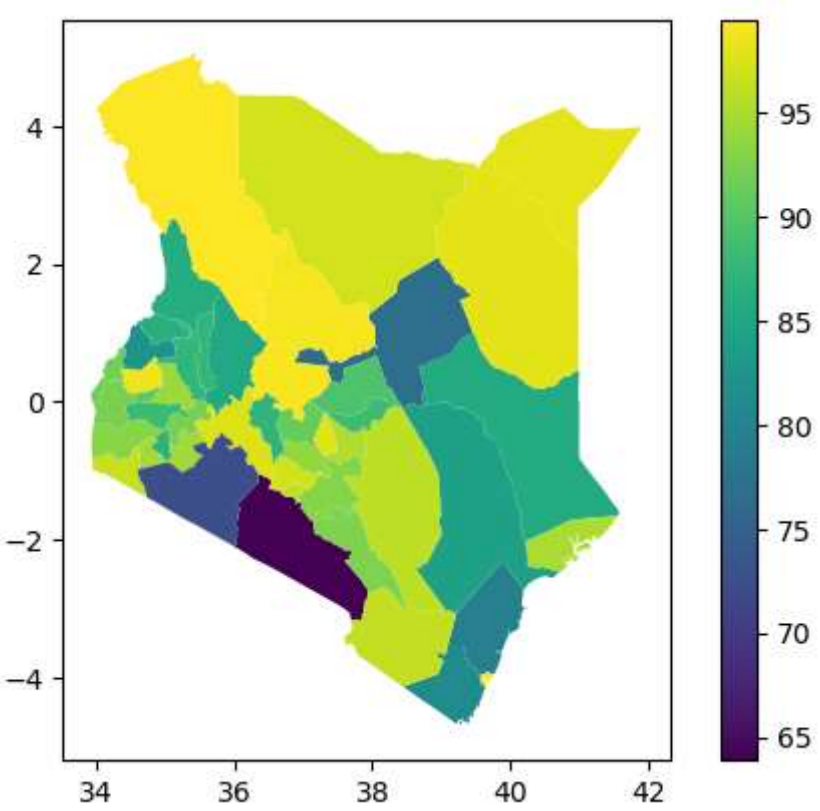
Out[14]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [15]:
```
import plotly.express as px
import pandas as pd
import geopandas as gpd
print("Import successfull!")
```

```
Import successfull!
```

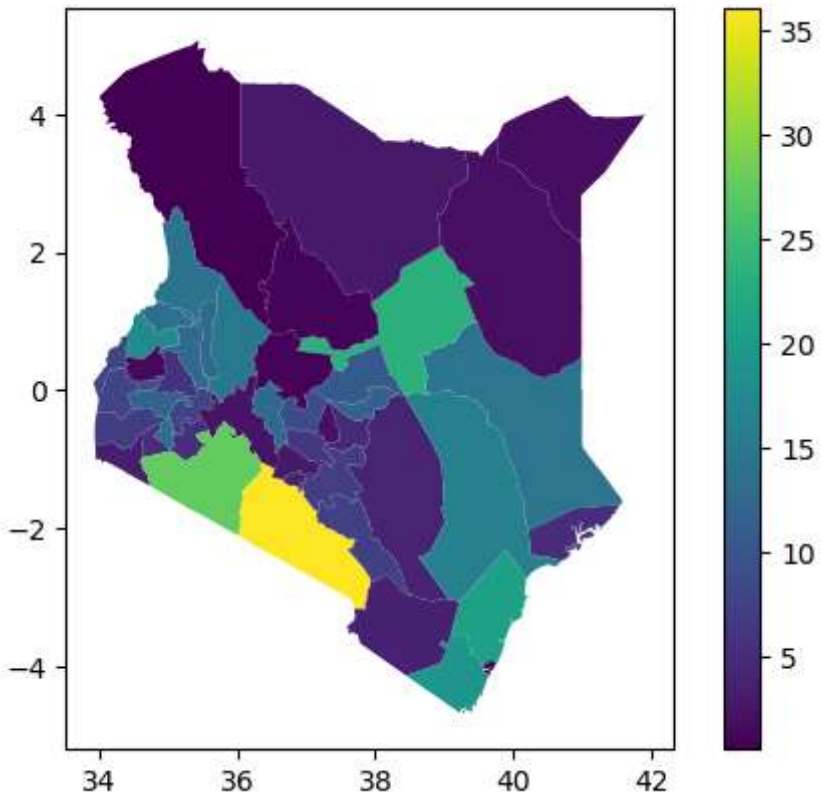
In [16]:
```
# Plot by % Transition Rate
in_file.plot(column="% Transition Rate", legend=True)
```

Out[16]:
```
<Axes: >
```



In [17]:
```
# Plot by failed transition rate
in_file.plot(column="% Failed Transition", legend=True)
```

Out[17]:
```
<Axes: >
```



In [18]:
```
# What is the minimum transition rate?
df = pd.DataFrame(in_file)

df1 = df.copy()
df1 = df1["% Transition Rate"]
df1 = pd.DataFrame(df1)
print(f"The minimum transition rate: {int(df1.min())}%")
# if df["% Transition Rate"].min
```

```
The minimum transition rate: 63%
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_9256\1864703180.py:7: FutureWarning: Ca
lling int on a single element Series is deprecated and will raise a TypeError in t
he future. Use int(ser.iloc[0]) instead
  print(f"The minimum transition rate: {int(df1.min())}%")
```

In [19]:
```
# What is the maximum transition rate?
print(f"The maximum transition rate: {int(df1.max())}%")
```

```
The maximum transition rate: 99%
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_9256\586124936.py:2: FutureWarning: Cal
ling int on a single element Series is deprecated and will raise a TypeError in th
e future. Use int(ser.iloc[0]) instead
  print(f"The maximum transition rate: {int(df1.max())}%")
```

In [20]:
```
df = pd.DataFrame(in_file[["County", "% Transition Rate", "% Failed Transition"]])
type(df)
```

Out[20]:
```
pandas.core.frame.DataFrame
```

In [21]:
```
df.head()
```

Out[21]:

	County	% Transition Rate	% Failed Transition
0	Baringo	85.235650	14.764350
1	Bomet	94.502704	5.497296
2	Bungoma	82.280975	17.719025
3	Busia	91.912585	8.087415
4	Elgeyo Marakwet	87.498994	12.501006

In [22]:
```
# geo_data = gpd.GeoDataFrame(in_file.columns[6:-1])
# print(type(geo_data))

geo_data = gpd.GeoDataFrame(in_file[["County", "GID_0", "Name", "Type", "County_No'
```

In [23]:
```
geo_data.head()
```

Out[23]:

	County	GID_0	Name	Type	County_No	geometry
0	Baringo	KEN	Kenya	County	30	MULTIPOLYGON (((35.67241 1.07306, 35.67228 1.0...
1	Bomet	KEN	Kenya	County	36	MULTIPOLYGON (((35.26193 -1.01562, 35.26177 -1...
2	Bungoma	KEN	Kenya	County	39	MULTIPOLYGON (((34.45476 0.47293, 34.45463 0.4...
3	Busia	KEN	Kenya	County	40	MULTIPOLYGON (((34.24632 0.31446, 34.24586 0.3...
4	Elgeyo Marakwet	KEN	Kenya	County	28	MULTIPOLYGON (((35.58806 0.17453, 35.58784 0.1...

Choropleth Mapping.

In [42]:
```
m = folium.Map(location=[0.18, 37.91], zoom_start=6) # initializing folium map

folium.Choropleth(
    geo_data=geo_data,
    name="Form One Transition Rates (%)",
    data=df,
    columns=["County", "% Transition Rate"],
    key_on="feature.properties.County",
    fill_opacity=0.7,
    fill_color="YlGn",
    line_opacity=0.2,
    legend_name="% Transitions"
).add_to(m)

cp = folium.Choropleth(
    geo_data=geo_data,
    name="Form One Failed Transition Rates (%)",
    data=df,
    columns=["County", "% Failed Transition"],
    key_on="feature.properties.County",
    fill_opacity=0.7,
    fill_color="Reds",
    line_opacity=0.2,
    legend_name="% Transitions"
```

```
).add_to(m)

# # creating County indexed version of the dataframe so we can lookup values
# df_indexed = df.set_index('County')

# # Looping through geojson object and adding a new property(% Transition Rate)
# for s in cp.geojson.data["features"]:
#     s['properties']['Transition Rate'] = df_indexed.loc[s['County_No'], '% Transi
#     folium.GeoJsonTooltip(['name', 'Transition Rate', 'Failed Transition Rate']).
folium.LayerControl().add_to(m)
m
```

Out[42]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [43]:
```
# cp.geojson.data
```

In [45]:
```
outfp = f"{os.getcwd()}//Data//choropleth_map.html"
m.save(outfp)
```

In [ ]:
```
(f"{os.getcwd()}//Data//SecondarySchoolsEnrollment.csv")
```