```python
#MSDS 460 - Asisgnment 3 - Linear Programming Code to allocoate Congressional Districts for the state of New Jersey
# May 8, 2022
# Vybav Hirasave, Grayson Matera, Dan Noel, Alex Skinner, Victor Tran

# our LP solution draws upon the following mode to define the objective function and establish constraints
# https://towardsdatascience.com/how-to-draw-congressional-districts-in-python-with-linear-programming-b1e33c80bc52

import geopandas as gpd
import numpy as np
import pandas as pd
import collections

pd.set_option('display.max_rows', 50)
pd.set_option('display.max_columns', None)

from PIL import Image, ImageOps
from plotnine import (ggplot, aes, geom_map, geom_text, geom_label,
                      ggtitle, element_blank, element_rect,
                      scale_fill_manual, theme_minimal, theme)
from pulp import (LpProblem, LpMinimize, LpVariable, lpSum,
                  PULP_CBC_CMD, GLPK_CMD, LpStatus, value)


# define the counties in New Jersey that will have the allocations - we have removed the six largest counties because they each get their own district

county_id = np.arange(0, 15)
county_names = np.array(['Atlantic','Burlington','Camden','Cape May','Cumberland', 'Gloucester', 'Hunterdon', 'Mercer', 'Morris', 'Passaic', 'Salem', 'Somerset', 'Sussex', 'Union', 'Warren'])
population_by_county = pd.DataFrame({'County_ID': county_id,
                                     'County_Name': county_names,
                                     'Population2020' : [274534,461860,523485,95263,154152,302294,128947,387340,59285,524118,64837,345361,144221,575345,109632]})

df_county_names = pd.DataFrame(county_names, columns = ['County'])
df = pd.DataFrame()
df['County']  = county_names
df['CountySort'] = county_id

county_populations = np.array(population_by_county['Population2020'])
state_population = sum(county_populations)
population_by_county.sort_values('Population2020', ascending=False).head()

n_counties = 15
n_districts = 6

# Create the linear programming model.

model = LpProblem("Supply-Demand-Problem", LpMinimize)
variable_names = [str(i)+'-'+str(j) for j in range(1, n_districts+1) \
                                    for i in range(1, n_counties+1)]
variable_names.sort()

```

```
print([item for item, count in collections.Counter(variable_names).items() if count > 1])

# The Decision Variable is 1 if the county is assigned to the district.
DV_variable_y = LpVariable.matrix("Y", variable_names, cat="Binary")
assignment = np.array(DV_variable_y).reshape(n_counties,n_districts)

# The Decision Variable is the population allocated to the district.
DV_variable_x = LpVariable.matrix("X", variable_names, cat="Integer",
                                  lowBound=0)
allocation = np.array(DV_variable_x).reshape(n_counties,n_districts)

# This objective minimizes the counties split among multiple districts.
objective_function = lpSum(assignment)
model += objective_function

# Initial Assignment / Allocation Constraints

# Allocate 100% of the population from each county.
for i in range(n_counties):
    model += lpSum(allocation[i][j] for j in range(n_districts)) == county_populations[i] , "Allocate All " + str(i)

# This constraint makes assignment required for allocation.
# sum(county_populations) is the "big M"
# At least 20% of population must be allocated for any assignment.
# while we do not end up splitting counties with these constraints, counnties are split with differt max/min population constraints
for i in range(n_counties):
    for j in range(n_districts):
        model += allocation[i][j] <= sum(county_populations)*assignment[i][j] , "Allocation assignment " + str(i) + '-' + str(j)
        if assignment[i][j] == 1:
             model += allocation[i][j] >= assignment[i][j]*0.20*county_populations[i] , "Allocation min " + str(i) + '-' + str(j)



ATLANTIC = 0
BURLINGTON = 1
CAMDEN = 2
CAPEMAY = 3
CUMBERLAND = 4
GLOUCESTER = 5
HUNTERDON = 6
MERCER = 7
MORRIS = 8
PASSAIC = 9
SALEM = 10
SOMERSET = 11
SUSSEX = 12
UNION = 13
WARREN = 14


for j in range(n_districts):

    #Atlantic County - Burlington/Camden/Cap May/Cumberland/Glocester
    model += assignment[ATLANTIC][j] <= assignment[BURLINGTON][j]+assignment[CAMDEN][j]+assignment[CAPEMAY][j]+assignment[CUMBERLAND][j]+assignment[GLOUCESTER][j]
```

```
    #Bergen County
    # gets own district

    #Burlington County - Atlantic/Camden/Mercer/Monmouth
    model += assignment[BURLINGTON][j] <= assignment[ATLANTIC][j]+assignment[CAMDEN][j]+assignment[MERCER][j]

    #Camden County - Atlantic/Burlington/Gloucester
    model += assignment[CAMDEN][j] <= assignment[ATLANTIC][j]+assignment[BURLINGTON][j]+assignment[GLOUCESTER][j]

    #Cape May County  - Atlantic/Cumberland
    model += assignment[CAPEMAY][j] <= assignment[ATLANTIC][j]+assignment[CUMBERLAND][j]

    #Cumberland County - Atlantic/Cape May/Gloucester/Salem
    model += assignment[CUMBERLAND][j] <= assignment[ATLANTIC][j]+assignment[CAPEMAY][j]+assignment[GLOUCESTER][j]+assignment[SALEM][j]

    #Essex County
    # gets own district

    #Gloucester County  - Atlantic/Camden/Cumberland/Salem
    model += assignment[GLOUCESTER][j] <= assignment[ATLANTIC][j]+assignment[CAMDEN][j]+assignment[CUMBERLAND][j]+assignment[SALEM][j]


    #Hudson County
    # gets own district

    #Hunterdon County - Mercer/Morris/Somerset/Warren
    model += assignment[HUNTERDON][j] <= assignment[MERCER][j]+assignment[MORRIS][j]+assignment[SOMERSET][j]+assignment[WARREN][j]

    #Mercer County - Burlington/Hunterdon/Somerset
    model += assignment[MERCER][j] <= assignment[BURLINGTON][j]+assignment[HUNTERDON][j]+assignment[SOMERSET][j]

    #Middlesex County
    # gets own district

    #Monmouth County  - Burlington/Mercer/Ocean
    # gets own district

    #Morris County - Hunterdon/Passaic/Somerset/Sussex/Union/Warren
    model += assignment[MORRIS][j] <= assignment[HUNTERDON][j]+assignment[PASSAIC][j]+assignment[SOMERSET][j]+assignment[SUSSEX][j]+assignment[UNION][j]+assignment[WARREN][j]

    #Ocean County      - Atlantic/Burlington/Monmouth
    # gets own disttrict

    #Passaic County   - Morris/Sussex
    model += assignment[PASSAIC][j] <= assignment[MORRIS][j]+assignment[SUSSEX][j]


    #Salem County      - Cumberland/Glocester
    model += assignment[SALEM][j] <= assignment[CUMBERLAND][j]+assignment[GLOUCESTER][j]
```

```python

    #Somerset County - Hunterdon/Mercer/Morris/Union
    model += assignment[SOMERSET][j] <= assignment[HUNTERDON][j]+assignment[MERCER][j]+assignment[MORRIS][j]+assignment[WARREN][j]

    #Sussex County   - Morris/Passaic/Warren
    model += assignment[SUSSEX][j] <= assignment[MORRIS][j]+assignment[PASSAIC][j]+assignment[WARREN][j]

    #Union County    - Morris/Somerset
    model += assignment[UNION][j] <= assignment[MORRIS][j]+assignment[SOMERSET][j]


    #Warren County   - Hunterdon/Morris/Sussex
    model += assignment[WARREN][j] <= assignment[HUNTERDON][j]+assignment[MORRIS][j]+assignment[SUSSEX][j]


# use wide max and min values to respect county lines - more narrow constraints will begin to split counties
for j in range(n_districts):
    model += lpSum(allocation[i][j] for i in range(n_counties)) <= 850000 , "District Size Maximum " + str(j)
    model += lpSum(allocation[i][j] for i in range(n_counties)) >= 400000 , "District Size Minimum " + str(j)

# for j in range(n_districts):
#    model += lpSum(allocation[i][j] for i in range(n_counties)) <= 700000 , "District Size Maximum " + str(j)
#    model += lpSum(allocation[i][j] for i in range(n_counties)) >= 600000 , "District Size Minimum " + str(j)

# Only allow counties that meet certain critera to be split among multiple districts
# A county must have population > 220,000 to be split among up to two districts
for i in range(n_counties): # added
    # if county_populations[i] <= 220000:
  if county_populations[i] <= 220000:
        model += lpSum(assignment[i][j] for j in range(n_districts)) <= 1  , "Unique Assignment " + str(i)
  else:
        model += lpSum(assignment[i][j] for j in range(n_districts)) <= 2  , "Up-to-two Assignments " + str(i)


# Solve the model
# !glpsol --help # for details on solver parameters
model.solve(GLPK_CMD(options=["--mipgap", "0.05", "--gomory"]))

print('The model status is: ',LpStatus[model.status])
print('The objective value is: ', value(objective_function))

for i in range(n_counties):
    for j in range(n_districts):
        if allocation[i][j].value() > 0:
            print('County %d assigned to district %d: ' % (i, j), allocation[i][j].value())




```