Redlining Impacts ~90 Years Later

Amy Tan

The Background

Redlining refers to "a discriminatory practice that consists of the systematic denial of services such as mortgages, insurance loans, and other financial services to residents of certain areas, based on their race or ethnicity."[1] The practice of redlining first emerged in the 1920s and 1930s, when U.S. Government's homeownership programs were established, and segregated cities and metropolitan areas based on race. The Home Owner's Loan Corporation (or HOLC) was one of these entitites that played a significant role in redlining - they graded neighborhoods based on their perceived mortgage-lending risk between 1930-1940.[2] These grades consisted of the following:

A: "Best"

B: "Desirable"

C: "Declining"

D: "Hazardous"

As a result of these grades, white folks were given loans and opportunities in A and B areas, and people of color (primarily targeting black folks) were relegated to C and D areas - essentially segregating folks due to the color of their skin. Although this practice of redlining was outlawed in 1968 in the Fair Housing Act, this does not mean there are lingering impacts to this day.[3] Thus, with this context, I aim to answer the following questions in this project.

The Questions

In this project, I aimed to answer two questions:

1. After ~90 years, are previously redlined areas still segregated?
2. If so, what states, divisions, and regions are the most segregated (i.e. are there patterns to more/less demographic inequity?)

The Data

To answer these questions, I utilized a dataset from FiveThirtyEight. The dataset contains 2020 population total estimates by race/ethnicity for combined zones of each redlining grade from the HOLC maps originally drawn in 1935-40. The population and race/ethnicity data comes from the 2020 U.S. decennial census: White, Black and Asian data excludes those who indicated Hispanic or Latino ethnicity. Hispanic/Latino data includes all who indicated Hispanic or Latino ethnicity, regardless of race. Other race data includes all population counts that did not fall under white, Black, Asian or Latino groups. Additionally, only micro- and metropolitan areas with both A- ("best") and D-rated ("hazardous") zones in their redlining map are included leaving 138 of 143 metropolitan areas in the data from Mapping Inequality.

About LQs

Each metropolitan area included in the dataset is further grouped by their HOLC score, which has its own LQ (Location Quotient) score. LQs are small-area measures of segregation that specifically compare one racial/ethnic group's proportion in a granular geography to their proportion in a larger surrounding geography. Below is the equation used to compute LQ scores:

$$LQ = \frac{\frac{x_{im}}{x_i}}{\frac{X_m}{X}}$$

- x_{im}: the population of a racial group m within a smaller geography i
- x_i: the total population of the same geography i
- X_m: the population of the racial group m within a bigger geography
- X: the total population of the bigger geography.

LQ Interpretation

- A value of 1 for this measure suggests that the racial group is perfectly represented
- A value greater than 1 for this measure suggests that the racial group is over represented
- A value less than 1 for this measure suggests that the racial group is under represented

Thus, ideally, if segregation no longer an issue (i.e., if redlining no longer has an impact), we should see LQ scores of around 1 for all racial groups.

General Imports

I began by importing necessary packages, loading in the dataset, and getting a general understanding of the data provided.

In [1]:
```
# general import statements
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
from plotly import express as px
```

In [2]:
```
# import dataset and show first few lines:
redline = pd.read_csv('data/metro-grades.csv')
redline.head()
```

Out[2]:

	metro_area	holc_grade	white_pop	black_pop	hisp_pop	asian_pop	other_pop	total_pop	pct_white	pct_black	...	surr_area_white_pop	surr_area_black_pop	su
0	Akron, OH	A	24702	8624	956	688	1993	36963	66.83	23.33	...	304399	70692	
1	Akron, OH	B	41531	16499	2208	3367	4211	67816	61.24	24.33	...	304399	70692	
2	Akron, OH	C	73105	22847	3149	6291	7302	112694	64.87	20.27	...	304399	70692	
3	Akron, OH	D	6179	6921	567	455	1022	15144	40.80	45.70	...	304399	70692	
4	Albany-Schenectady-Troy, NY	A	16989	1818	1317	1998	1182	23303	72.91	7.80	...	387016	68371	

5 rows × 28 columns

In [3]:
```
redline.shape
```

Out[3]:
```
(551, 28)
```

In [4]:
```
redline.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 551 entries, 0 to 550
Data columns (total 28 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   metro_area           551 non-null    object 
 1   holc_grade           551 non-null    object 
 2   white_pop            551 non-null    int64  
 3   black_pop            551 non-null    int64  
 4   hisp_pop             551 non-null    int64  
 5   asian_pop            551 non-null    int64  
 6   other_pop            551 non-null    int64  
 7   total_pop            551 non-null    int64  
 8   pct_white            551 non-null    float64
 9   pct_black            551 non-null    float64
 10  pct_hisp             551 non-null    float64
 11  pct_asian            551 non-null    float64
 12  pct_other            551 non-null    float64
 13  lq_white             551 non-null    float64
 14  lq_black             551 non-null    float64
 15  lq_hisp              551 non-null    float64
 16  lq_asian             551 non-null    float64
 17  lq_other             551 non-null    float64
 18  surr_area_white_pop  551 non-null    int64  
 19  surr_area_black_pop  551 non-null    int64  
 20  surr_area_hisp_pop   551 non-null    int64  
 21  surr_area_asian_pop  551 non-null    int64  
 22  surr_area_other_pop  551 non-null    int64  
 23  surr_area_pct_white  551 non-null    float64
 24  surr_area_pct_black  551 non-null    float64
 25  surr_area_pct_hisp   551 non-null    float64
 26  surr_area_pct_asian  551 non-null    float64
 27  surr_area_pct_other  551 non-null    float64
dtypes: float64(15), int64(11), object(2)
memory usage: 120.7+ KB
```

Q1: After ~90 years, are previously redlined areas still segregated?

To get a general idea of 2020 demographic distributions based upon the HOLC grades given, I grouped by the HOLC grades and see the average percentage of the demographic groups, in addition to the average LQ scores. Again, a score above 1 indicates that the group is overrepresented relative to the surrounding areas, and a score below 1 indicates that the group is underrepresented relative to the surrounding areas. I plotted the scores below, the gray dashed line at an LQ score of 1.

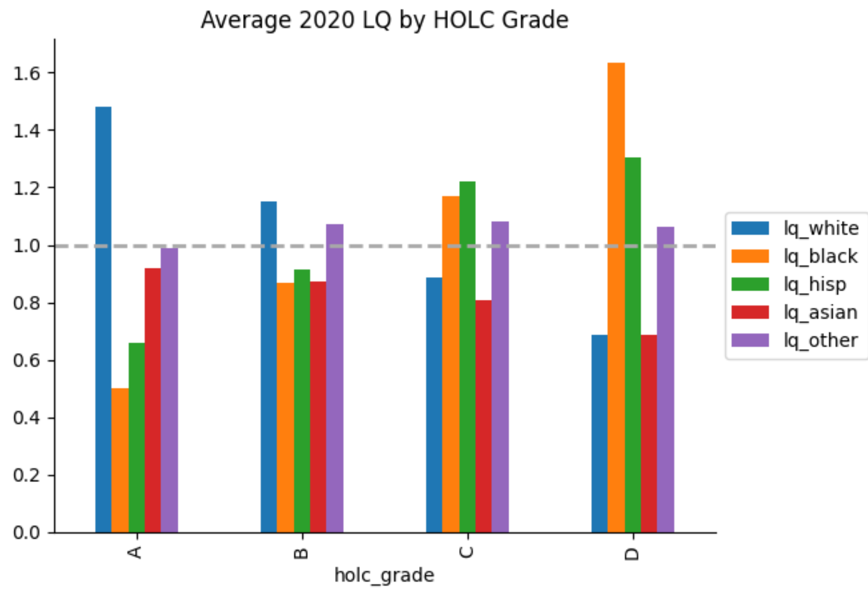
In [5]:
```
# group by the HOLC grade and keep lq scores
by_holc = redline.drop('metro_area', axis = 1).groupby(by = 'holc_grade', axis = 0).mean()[['lq_white', 'lq_black', 'lq_hisp', 'lq_asian', '
by_holc
```

Out[5]:

	holc_grade	lq_white	lq_black	lq_hisp	lq_asian	lq_other
0	A	1.478478	0.498768	0.658841	0.921087	0.990145
1	B	1.150870	0.869058	0.916377	0.870797	1.071087
2	C	0.884745	1.167664	1.219124	0.807883	1.083358
3	D	0.685652	1.634783	1.303478	0.686159	1.062319

In [6]:
```
by_holc.plot(x = 'holc_grade', kind = 'bar',
             title = 'Average 2020 LQ by HOLC Grade')
```

```
plt.legend(loc="center left", bbox_to_anchor=(1, 0.5))
plt.axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
```



As shown in the bar graph, there is an overrepresentation of White folks in HOLC designated A and B areas, coupled with an overrepresentation of Black and Hispanic folks in HOLC designated C and D areas - following the exact pattern of what the HOLC set out to do back in the 1930s. The data suggests that, on average, the impacts of redlining are still felt almost 90 years later.

To further explore and visualize the relationship between HOLC grade and demographics, I randomly sampled the `redline` dataset, using a `random state = 1234` for the purposes of reproduceability. Following that, the 5 samples were plotted by percent to keep comparisons equal across the 5 metro areas.

In [8]:
```
redline.sample(n = 5, random_state = 1234)
```

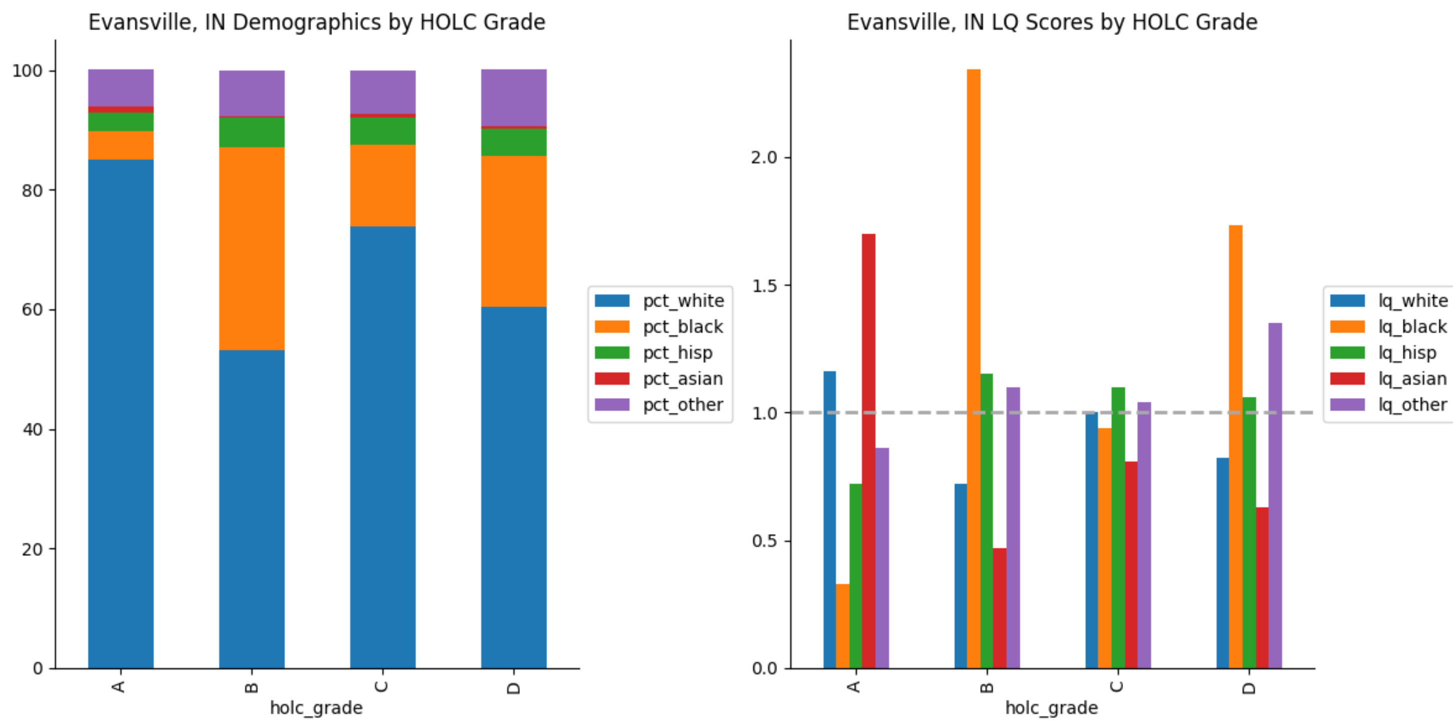
Out[8]:

	metro_area	holc_grade	white_pop	black_pop	hisp_pop	asian_pop	other_pop	total_pop	pct_white	pct_black	...	surr_area_white_pop	surr_area_black_pop	s
169	Evansville, IN-KY	B	1764	1135	159	11	257	3325	53.04	34.13	...	63444	12579	
355	Phoenix-Mesa-Chandler, AZ	D	2518	2553	9668	368	819	15926	15.81	16.03	...	55623	12810	
527	Waterloo-Cedar Falls, IA	A	4060	462	325	129	312	5287	76.79	8.74	...	33764	11293	
221	Jacksonville, FL	B	13253	3868	1312	380	1120	19931	66.49	19.41	...	82335	105268	
516	Utica-Rome, NY	B	5390	418	546	473	318	7146	75.43	5.86	...	64346	11130	

5 rows × 28 columns

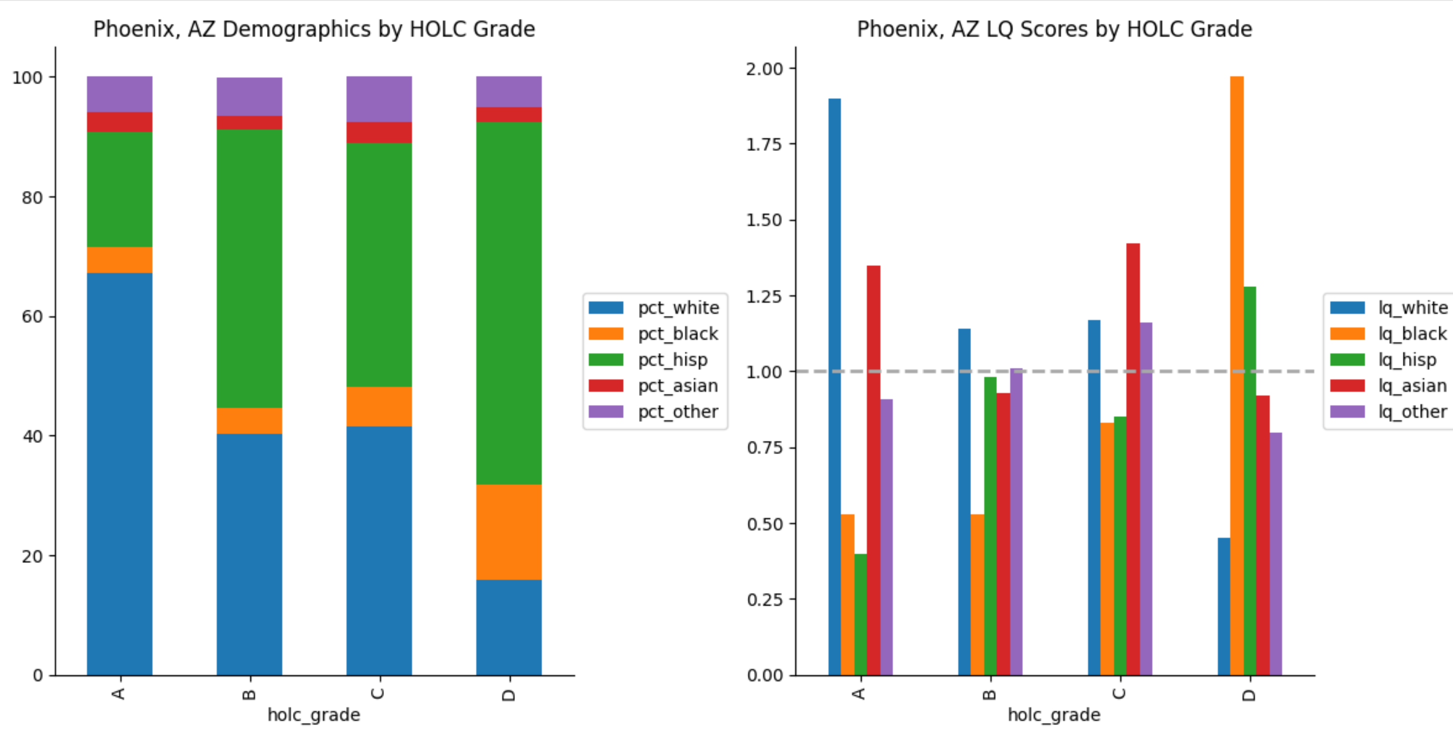
Plotting Sample 1: Evansville, IN

In [9]:
```
sample1_pct = redline[redline['metro_area']== 'Evansville, IN-KY'][['holc_grade','pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other']]
sample1_lq = redline[redline['metro_area']== 'Evansville, IN-KY'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian', 'lq_other']]

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
sample1_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
             title='Evansville, IN Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
sample1_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
             title='Evansville, IN LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



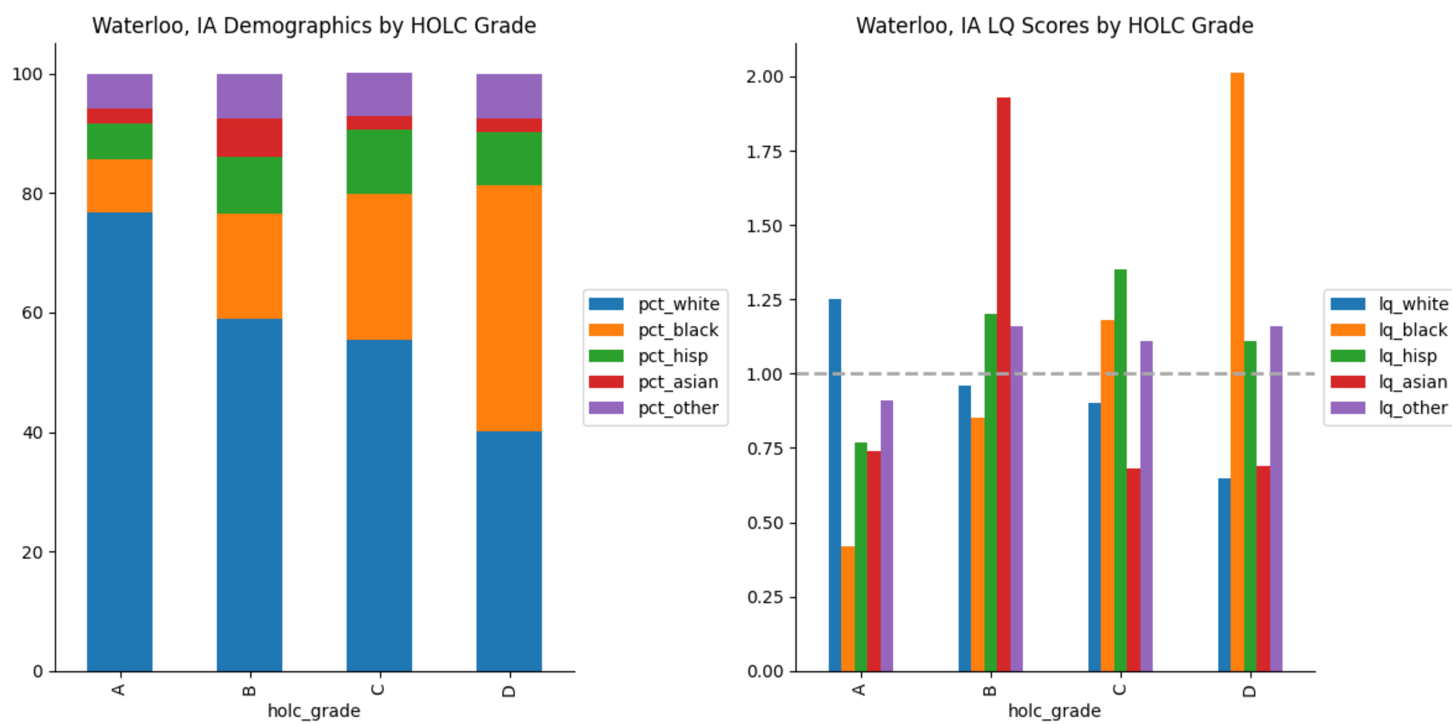
Plotting Sample 2: Phoenix, AZ

In [10]:
```
sample2_pct = redline[redline['metro_area']== 'Phoenix-Mesa-Chandler, AZ'][['holc_grade','pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other']]
sample2_lq = redline[redline['metro_area']== 'Phoenix-Mesa-Chandler, AZ'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian', 'lq_ot

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
sample2_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
             title='Phoenix, AZ Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
sample2_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
             title='Phoenix, AZ LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--',
              linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



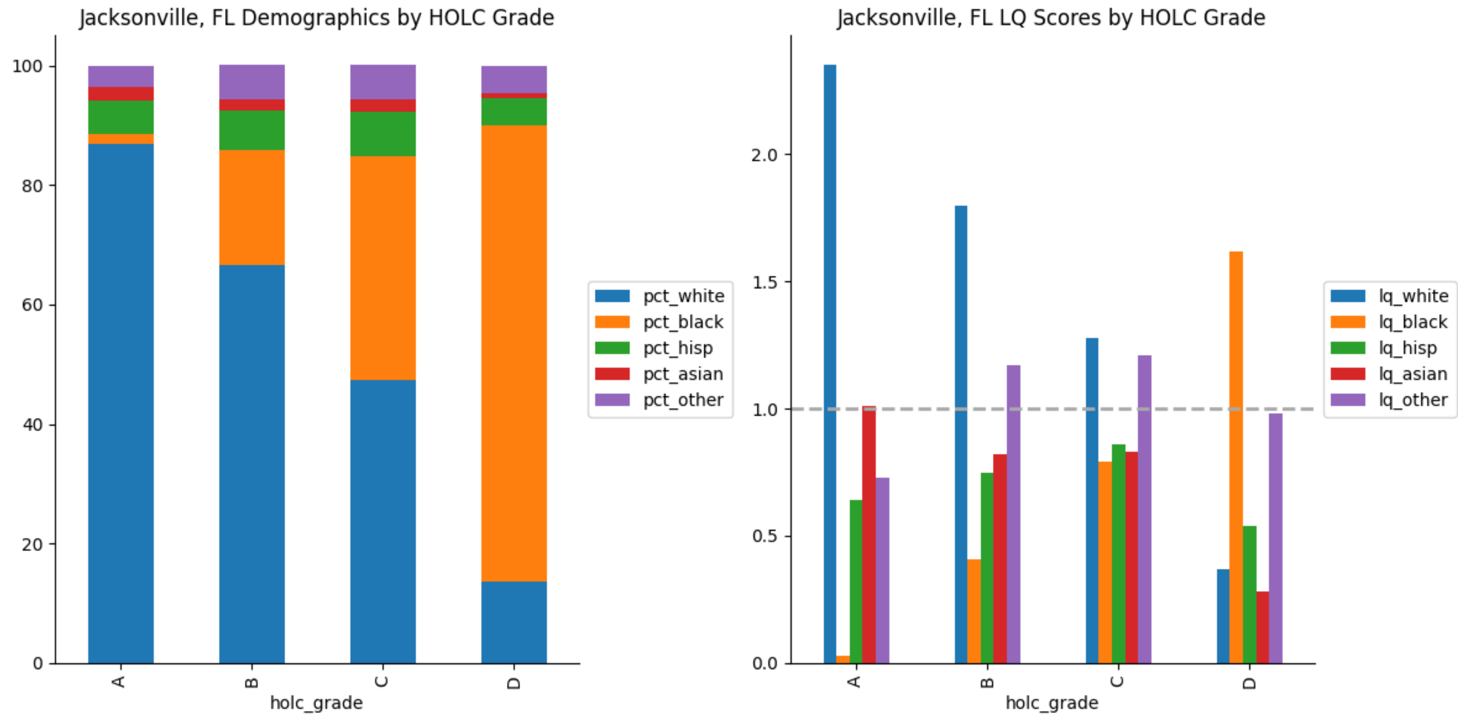
Plotting Sample 3: Waterloo, IA

In [11]:
```
sample3_pct = redline[redline['metro_area']== 'Waterloo-Cedar Falls, IA'][['holc_grade','pct_white', 'pct_black','pct_hisp', 'pct_asian', 'pc
sample3_lq = redline[redline['metro_area']== 'Waterloo-Cedar Falls, IA'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian', 'lq_oth

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
sample3_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
             title='Waterloo, IA Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
sample3_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
             title='Waterloo, IA LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



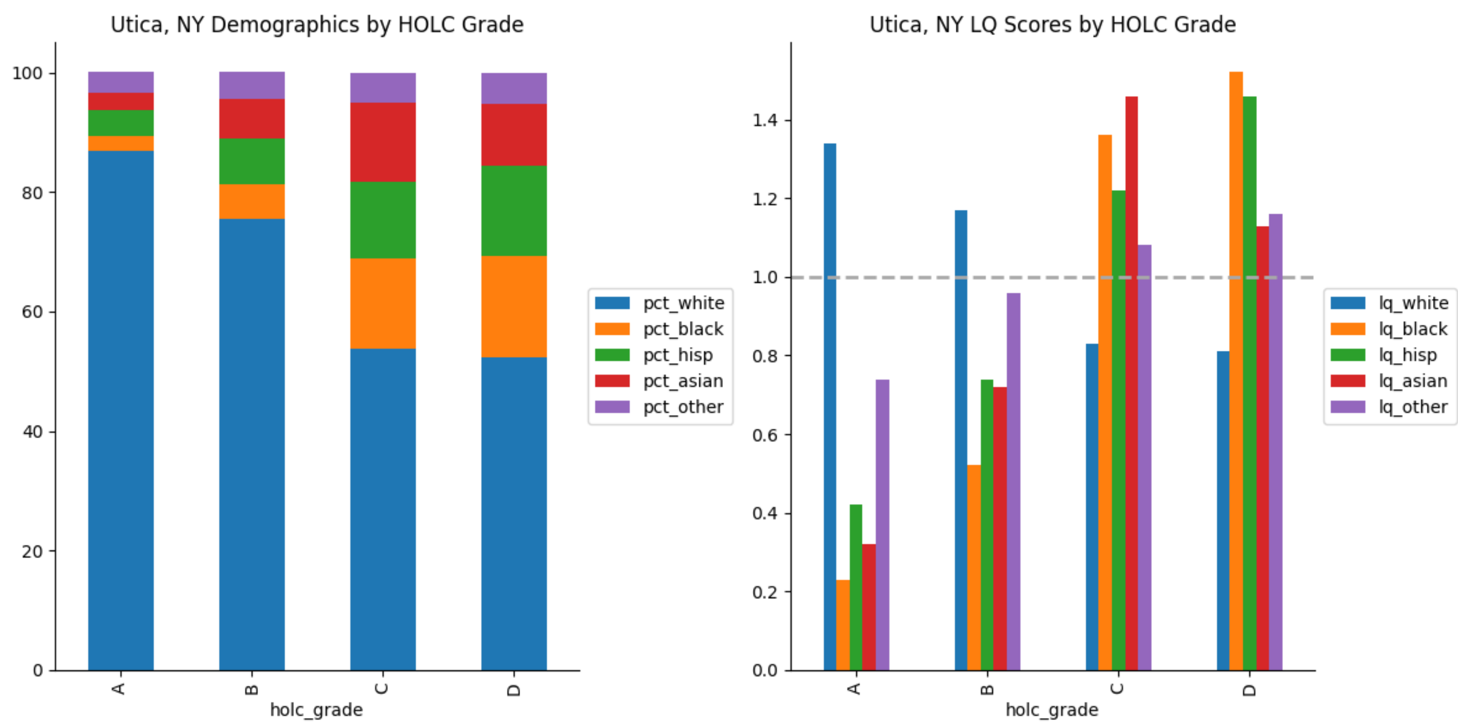
Plotting Sample 4: Jacksonville, FL

In [12]:
```
sample4_pct = redline[redline['metro_area']== 'Jacksonville, FL'][['holc_grade','pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other']]
sample4_lq = redline[redline['metro_area']== 'Jacksonville, FL'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian', 'lq_other']]


fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
sample4_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
             title='Jacksonville, FL Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
sample4_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
             title='Jacksonville, FL LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



Plotting Sample 5: Utica, NY

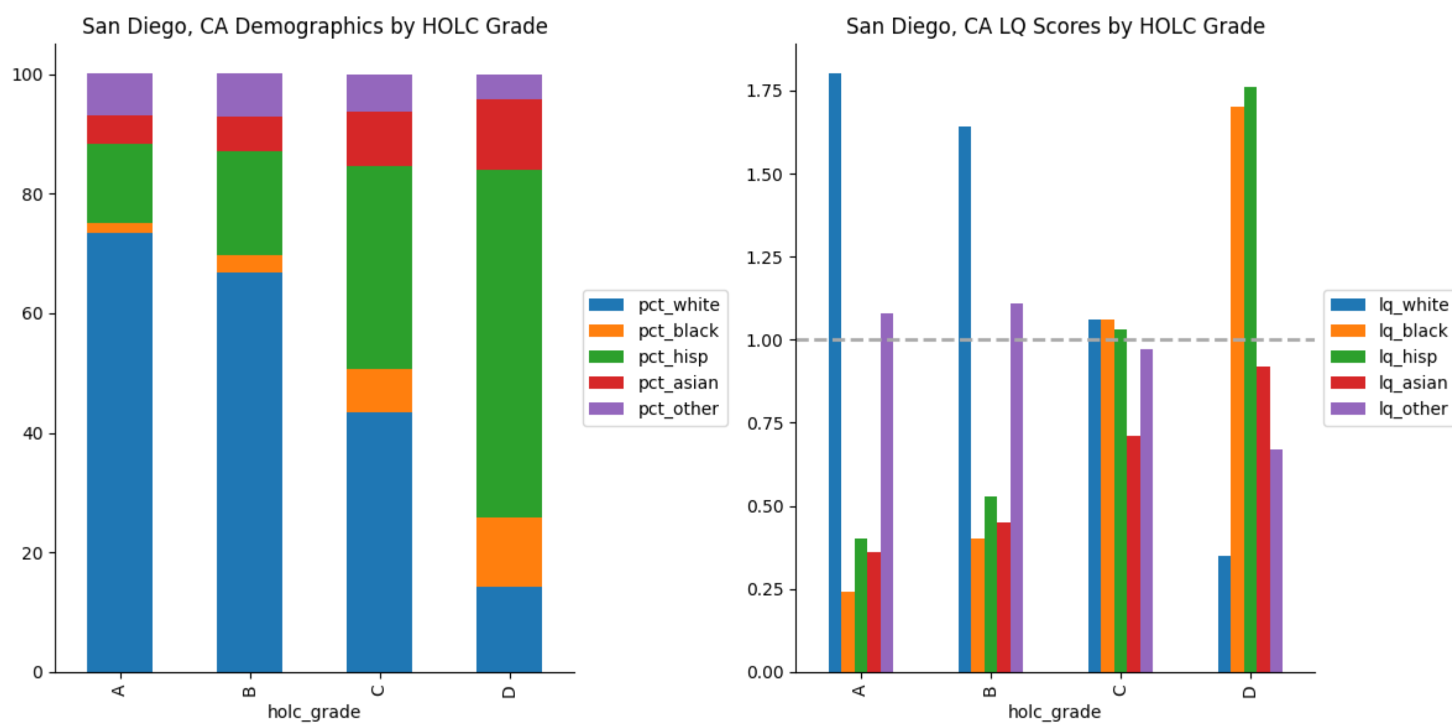
In [13]:
```
sample5_pct = redline[redline['metro_area']== 'Utica-Rome, NY'][['holc_grade','pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other']]
sample5_lq = redline[redline['metro_area']== 'Utica-Rome, NY'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian', 'lq_other']]


fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
sample5_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
             title='Utica, NY Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
sample5_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
             title='Utica, NY LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



The demographic distributions shown for all 5 random samples seem to indicate a pattern where higher HOLC grades have higher proportions of White people with fewer POC. The lower the HOLC grade, the lower proportions of White people and the higher the proportions of POC. This is most notable for Black and Hispanic people, confirming the previous finding that impacts of redlining are still felt today.

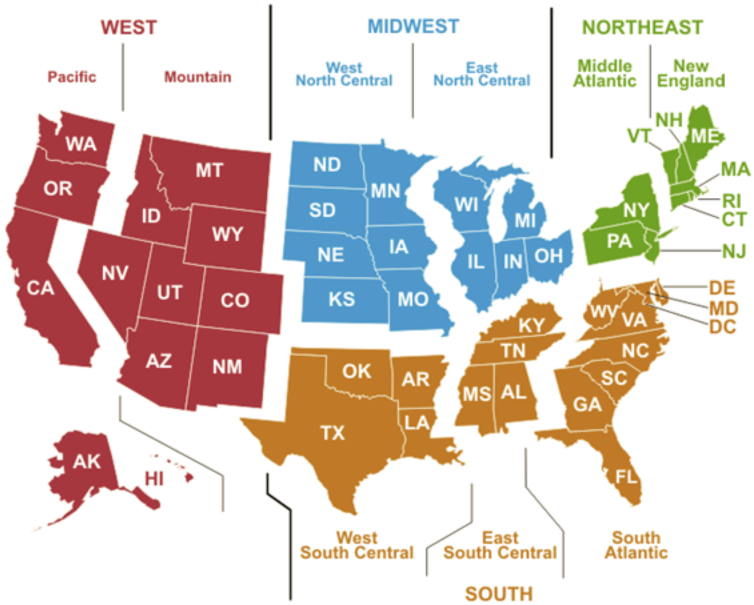
Additionally, out of curiousity, I examined the San Diego area, where the same patterns occurs again:

In [14]:
```
sd_pct = redline[redline['metro_area']== 'San Diego-Chula Vista-Carlsbad, CA'][['holc_grade','pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other']]
sd_lq = redline[redline['metro_area']== 'San Diego-Chula Vista-Carlsbad, CA'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian', '

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
sd_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
            title='San Diego, CA Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
sd_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
            title='San Diego, CA LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```



Q2: What states, divisions, and regions are the most segregated?

To examine whether there are patterns to more or less demographic inequity, I wanted to examine segregation on the state, division, and regional level. US divisions and regions are defined by the US Census, and can give broader grouping beyond the state level. I tentatively hypothesized that states, divisions, and regions with histories of slavery would be more segregated than those that did not.



States

To examine whether there were particular states where the impacts of redlining were felt more than others, I first created two functions (`get_state` and `get_city`) to extract both the state and city from the `metro_area` column.

Note that according to the dataset dictionary `metro_area` is the: "Official U.S. Census name of micro- or metropolitan area — defined as 'Core-Based Statistical Areas'. The first city and state listed are used as the display name for each micro/metropolitan area in the story (for example, "Chicago-Naperville-Elgin, IL-IN-WI" is referred to as "Chicago, IL")." Thus, this analysis is not fully representative and does not completely capture the naunce of metro areas extending beyond state and city boundaries. This point is expanded upon in the limitations at the end of the project.

In [15]:
```
# creating a function that will take the metro_area info and extract the state
def get_state(string):
    """
    Extracts the state from a string in the format "city, state abbreviation." If there is more than one
    state, the function returns the first state.

    Parameters:
    string: a string in the format "city, state abbreviation"

    Returns:
    The state.
    """
    new_string = string.split(',')[1].strip()
    # in case there is more than one state, take only the first state
    if '-' in new_string:
        new_string = new_string.split('-')[0].strip()
    return new_string

#creating a function that will take the metro_area info and extract the city
def get_city(string):
    """
    Extracts the city from a string in the format "city, state abbreviation."If there is more than one
    city, the function returns the first city.

    Parameters:
    string: a string in the format "city, state abbreviation"

    Returns:
    The city.
    """
    new_string = string.split(',')[0].strip()
    # in case there is more than one city, take only the first city
    if '-' in new_string:
        new_string = new_string.split('-')[0].strip()
    return new_string
```

I then applied both functions to the `redline` dataset, creating new columns to store both state and city names.

In [16]:
```
# applying get_state function to create a new column 'state'
redline['state'] = redline.metro_area.apply(get_state)

# applying get_city function to create a new column 'city'
redline['city'] = redline.metro_area.apply(get_city)

# reorder columns to have state by the metro_area
redline = redline[['metro_area', 'city', 'state', 'holc_grade', 'white_pop', 'black_pop', 'hisp_pop',
       'asian_pop', 'other_pop', 'total_pop', 'pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other', 'lq_white', 'lq_black', 'lq_hisp',
       'lq_asian', 'lq_other', 'surr_area_white_pop', 'surr_area_black_pop',
       'surr_area_hisp_pop', 'surr_area_asian_pop', 'surr_area_other_pop',
       'surr_area_pct_white', 'surr_area_pct_black', 'surr_area_pct_hisp',
       'surr_area_pct_asian', 'surr_area_pct_other']]
```

```python
# double-check this was successful
redline.head()
```

Out[16]:

	metro_area	city	state	holc_grade	white_pop	black_pop	hisp_pop	asian_pop	other_pop	total_pop	...	surr_area_white_pop	surr_area_black_pop	surr_area_
0	Akron, OH	Akron	OH	A	24702	8624	956	688	1993	36963	...	304399	70692	
1	Akron, OH	Akron	OH	B	41531	16499	2208	3367	4211	67816	...	304399	70692	
2	Akron, OH	Akron	OH	C	73105	22847	3149	6291	7302	112694	...	304399	70692	
3	Akron, OH	Akron	OH	D	6179	6921	567	455	1022	15144	...	304399	70692	
4	Albany-Schenectady-Troy, NY	Albany	NY	A	16989	1818	1317	1998	1182	23303	...	387016	68371	

5 rows × 30 columns

Creating a Broader Inequity Coefficient

In order to calculate an "overall" measure of over/under representation for a given city we can use the LQ scores for all demographic groups for all HOLC scores. Since an LQ score of 1 indicates a perfect representation of a given group, we can calculate the squared deviation scores to then calculate a city's LQ variance to get a sense of how equitable a city is overall. These LQ variance scores have been placed in a new table, `metro_area_demographics`. The closer the LQ variance is to 0, the less segregation that area has; the larger the LQ variance is, the more segregation that area has.

In [17]:
```python
# getting a list of unique metro_areas in the dataset
metro_areas_unique = redline['metro_area'].unique().tolist()

# getting a list of the lq score columns
lq_scores = ['lq_white', 'lq_black', 'lq_hisp','lq_asian', 'lq_other']
```

In [18]:
```python
# creating a function calc_sqdeviation to compute a given lq's deviation from 1 squared
def calc_sqdeviation(lq):
    """
    Calculates the squared deviation of a given lq score and 1.

    Parameters:
    lq: a float

    Returns: the lq's squared deviation from 1
    """
    return (1-lq)**2

metro_area_demographics = pd.DataFrame()
metro_area_demographics['metro_area'] = metro_areas_unique

# apply get_state and get_city functions for graphing and easier comprehension
metro_area_demographics['city'] = metro_area_demographics['metro_area'].apply(get_city)
metro_area_demographics['state'] = metro_area_demographics['metro_area'].apply(get_state)
```

In [19]:
```python
lq_variance = []

# loop through each unique metro_area
for area in metro_areas_unique:
    temp_dat = redline[redline['metro_area'] == area]
    temp_arr = []

    # loop through each demographic group's LQ scores and append each score into an array
    for demographic in lq_scores:
        temp_arr.extend(temp_dat[demographic].tolist())

    # apply the calc_sqdeviation function to the array, sum together, and divide by 20
    # append the lq_variance of lq scores for the city to the deviations list
    lq_variance.append(np.sum(calc_sqdeviation(np.array(temp_arr)))/20)

# add the variances to the metro_area_demographics
metro_area_demographics['state_lq_variance'] = lq_variance
```

In [20]:
```python
metro_area_demographics.sort_values(by = 'state_lq_variance', ascending=False)
```

Out[20]:

	metro_area	city	state	state_lq_variance
51	Huntington-Ashland, WV-KY-OH	Huntington	WV	1.689085
136	York-Hanover, PA	York	PA	1.003470
45	Fresno, CA	Fresno	CA	0.750280
137	Youngstown-Warren-Boardman, OH-PA	Youngstown	OH	0.642940
68	Macon-Bibb County, GA	Macon	GA	0.579285
...	...	...	...	...
83	Ogden-Clearfield, UT	Ogden	UT	0.044390
63	Lincoln, NE	Lincoln	NE	0.044145
112	Sioux City, IA-NE-SD	Sioux City	IA	0.039785
94	Pueblo, CO	Pueblo	CO	0.037970
40	Elmira, NY	Elmira	NY	0.033595

138 rows × 4 columns

While this provides an overall LQ variance for each metro_area, giving a full sense of its demographic over/underrepresentation, the question at hand is in regards to states. Therefore, some additional wrangling needs to take place.

In [21]:
```python
# creating a new df grouping metro_area_demographics by state, and averaging their LQ variances
state_demographics = metro_area_demographics.drop(['metro_area', 'city'], axis = 1).groupby('state').mean().reset_index()
state_demographics.sort_values(by = 'state_lq_variance', ascending = True)
```

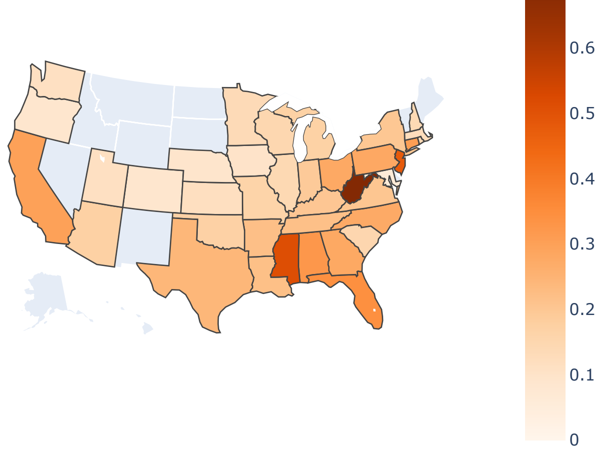
Out[21]:

	state	state_lq_variance
15	MD	0.057375
4	CO	0.086015
27	OR	0.087245
21	NE	0.092008
8	IA	0.098249
33	UT	0.111258
35	WA	0.112187
11	KS	0.114725
14	MA	0.122860
17	MN	0.134102
22	NH	0.136240
29	RI	0.136685
9	IL	0.140579
36	WI	0.147368
30	SC	0.149870
18	MO	0.160264
16	MI	0.168880
26	OK	0.169997
2	AZ	0.170895
10	IN	0.189189
24	NY	0.198504
12	KY	0.199458
34	VA	0.201569
31	TN	0.215592
13	LA	0.217280
1	AR	0.218440
32	TX	0.240018
20	NC	0.272092
25	OH	0.274982
7	GA	0.276953
28	PA	0.277278
3	CA	0.297331
5	CT	0.304915
0	AL	0.322798
6	FL	0.342768
23	NJ	0.478085
19	MS	0.509615
37	WV	0.690337

From there, we can plot the LQ variance per state. Most notably, West Virginia has the most inequity, with an LQ variance of 0.69. This led me to explore West Virgina further to see if there was a potential explanation for this value.

In [22]:
```python
fig = px.choropleth(state_demographics,
                    locations='state',
                    locationmode='USA-states',  # Recognizes state names or abbreviations
                    color='state_lq_variance',
                    scope="usa",  # Focus on the USA
                    color_continuous_scale='Oranges',
                    range_color=[0, 0.7])

fig.update_layout(title_text='Demographic Inequity By State', geo_scope='usa')

fig.show()
```

Demographic Inequity By State



The West Virigina Problem

To explore this issue further, I first selected for metro area's belonging to the state of West Virgina. Taking a look at the output, it becomes clear that the city of Huntington is an outlier compared to the LQ variance values of the other cities.

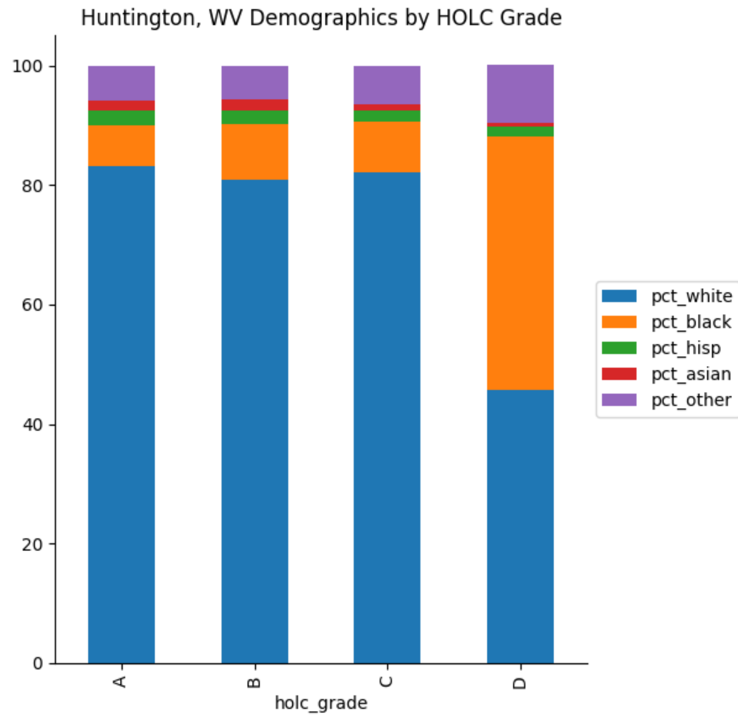
In [23]:
```
metro_area_demographics[metro_area_demographics['state'] == 'WV']
```

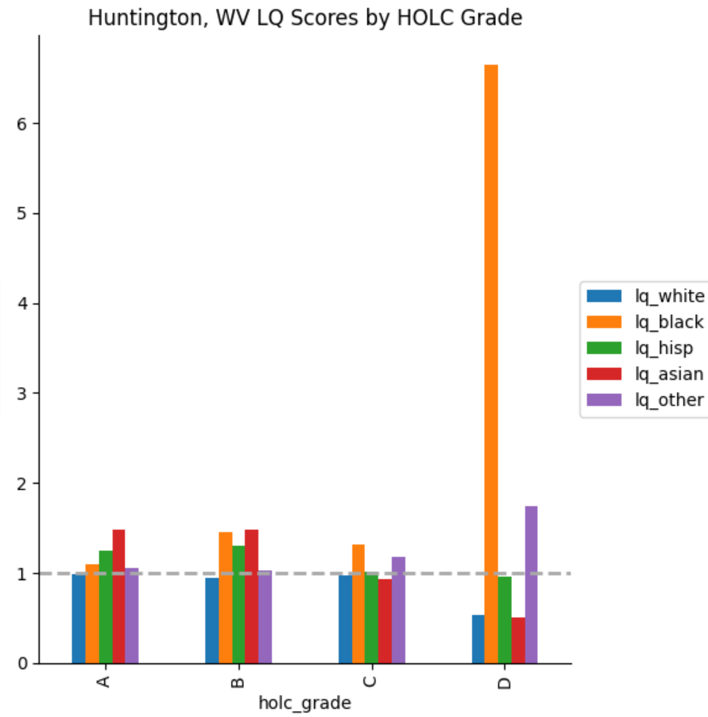
Out[23]:

	metro_area	city	state	state_lq_variance
20	Charleston, WV	Charleston	WV	0.140800
51	Huntington-Ashland, WV-KY-OH	Huntington	WV	1.689085
133	Wheeling, WV-OH	Wheeling	WV	0.241125

Thus, I wanted to explore Huntington further - I went ahead and plotted the metro area the same way I did in answering question one. From the graph, it becomes clear that the LQ score for black folks at a HOLC grade of D is extremely high in comparison to all other surrounding LQ scores. Although I researched into this particular city to see if there was a potential explanation for an LQ score this high, I did not find anything significant in regards to this. Additionally, this might have been a data entry error, but again, it's difficult to know without further information. Thus, for the sake of this project, I decided to remove the Huntington metro area from this project to avoid skewing results - this decision is expanded upon further in the Limitations section. The remainder of results, on the state, division, and regional level, are all done with the Huntington metro_area removed.

In [24]:
```
huntington_pct = redline[redline['metro_area'] == 'Huntington-Ashland, WV-KY-OH'][['holc_grade','pct_white', 'pct_black',
        'pct_hisp', 'pct_asian', 'pct_other']]
huntington_lq = redline[redline['metro_area'] == 'Huntington-Ashland, WV-KY-OH'][['holc_grade','lq_white', 'lq_black', 'lq_hisp', 'lq_asian',

fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# First plot
huntington_pct.plot(x='holc_grade', kind='bar', stacked=True, ax=ax[0],
             title='Huntington, WV Demographics by HOLC Grade')
ax[0].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[0].spines['top'].set_visible(False)
ax[0].spines['right'].set_visible(False)

# Second plot
huntington_lq.plot(x='holc_grade', kind='bar', ax=ax[1],
             title='Huntington, WV LQ Scores by HOLC Grade')
ax[1].legend(loc="center left", bbox_to_anchor=(1, 0.5))
ax[1].axhline(y=1, color='darkgray', linestyle='--', linewidth=2)
ax[1].spines['top'].set_visible(False)
ax[1].spines['right'].set_visible(False)

plt.tight_layout()
plt.show()
```





In [25]:
```
# remove the Huntington-Ashland, WV-KY-OH entry from the dataset
new_metro_area_demographics = metro_area_demographics[metro_area_demographics['metro_area'] != 'Huntington-Ashland, WV-KY-OH']

# create a new dataset with the state lq variance, without Huntington-Ashland, WV-KY-OH entry from the dataset
new_state_demographics = new_metro_area_demographics.drop(['metro_area', 'city'], axis = 1).groupby('state').mean().reset_index()
new_state_demographics.sort_values(by = 'state', ascending = True).head()
```

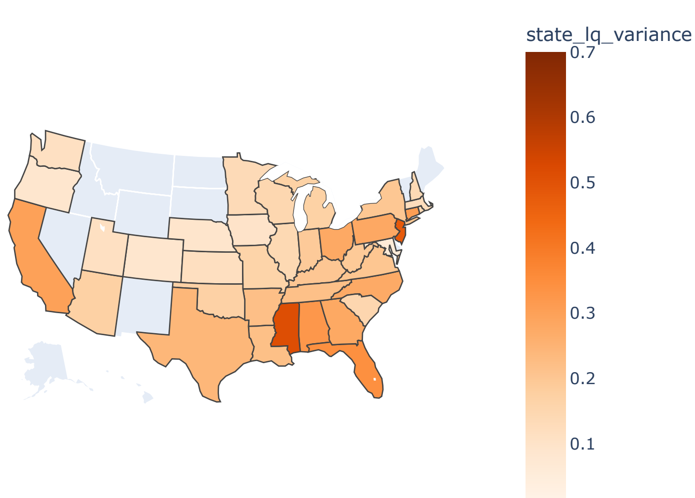
Out[25]:

	state	state_lq_variance
0	AL	0.322798
1	AR	0.218440
2	AZ	0.170895
3	CA	0.297331
4	CO	0.086015

Replotting the LQ variances by region now gives a much different picture than before. Mississpi and New Jersey are the most segregated states, whereas Maryland and Colorado are the least.

In [26]:
```
fig = px.choropleth(new_state_demographics,
                    locations='state',
                    locationmode='USA-states',  # Recognizes state names or abbreviations
                    color='state_lq_variance',
                    scope="usa",  # Focus on the USA
                    color_continuous_scale='Oranges',
                    range_color=[0, 0.7])

fig.update_layout(title_text='Demographic Inequity by State (Without Huntington)', geo_scope='usa')

fig.show()
```

Demographic Inequity by State (Without Huntington)



As a quick check, I looked at Mississippi and New Jersey to ensure there weren't any outliers like West Virginia impacting the calculations. Nothing stood out, so I moved forward with examining divisions and regions.

In [27]:
```
new_metro_area_demographics[new_metro_area_demographics['state'] == 'MS']
```

Out[27]:

	metro_area	city	state	state_lq_variance
54	Jackson, MS	Jackson	MS	0.509615

In [28]:
```
new_metro_area_demographics[new_metro_area_demographics['state'] == 'NJ']
```

Out[28]:

	metro_area	city	state	state_lq_variance
7	Atlantic City-Hammonton, NJ	Atlantic City	NJ	0.521725
127	Trenton-Princeton, NJ	Trenton	NJ	0.434445

Divisions and Regions

Creating a Divison and Region Column

Moving on to divisions and regions, I first created a division and region columns. To do this, I created a dictionary `state_to` that contains each states' division and region and created two functions, `get_division` and `get_region`, that I applied to the state column.

In [29]:
```
# create a dictionary where state abbreviations are the keys, and the [divison, region] is the value

state_to = {
    'WA': ['PACIFIC', 'WEST'],
    'OR': ['PACIFIC', 'WEST'],
    'CA': ['PACIFIC', 'WEST'],
    'HI': ['PACIFIC', 'WEST'],
    'AK': ['PACIFIC', 'WEST'],

    'MT': ['MOUNTAIN', 'WEST'],
    'ID': ['MOUNTAIN', 'WEST'],
    'WY': ['MOUNTAIN', 'WEST'],
    'NV': ['MOUNTAIN', 'WEST'],
    'UT': ['MOUNTAIN', 'WEST'],
    'CO': ['MOUNTAIN', 'WEST'],
    'AZ': ['MOUNTAIN', 'WEST'],
    'NM': ['MOUNTAIN', 'WEST'],



    'ND': ['WEST NORTH CENTRAL', 'MIDWEST'],
    'SD': ['WEST NORTH CENTRAL', 'MIDWEST'],
    'MN': ['WEST NORTH CENTRAL', 'MIDWEST'],
    'NE': ['WEST NORTH CENTRAL', 'MIDWEST'],
    'IA': ['WEST NORTH CENTRAL', 'MIDWEST'],
    'KS': ['WEST NORTH CENTRAL', 'MIDWEST'],
    'MO': ['WEST NORTH CENTRAL', 'MIDWEST'],

    'WI': ['EAST NORTH CENTRAL', 'MIDWEST'],
    'IL': ['EAST NORTH CENTRAL', 'MIDWEST'],
    'MI': ['EAST NORTH CENTRAL', 'MIDWEST'],
    'IN': ['EAST NORTH CENTRAL', 'MIDWEST'],
    'OH': ['EAST NORTH CENTRAL', 'MIDWEST'],



    'PA': ['MIDDLE ATLANTIC', 'NORTHEAST'],
    'NY': ['MIDDLE ATLANTIC', 'NORTHEAST'],
    'NJ': ['MIDDLE ATLANTIC', 'NORTHEAST'],

    'VT': ['NEW ENGLAND', 'NORTHEAST'],
    'ME': ['NEW ENGLAND', 'NORTHEAST'],
    'NH': ['NEW ENGLAND', 'NORTHEAST'],
    'MA': ['NEW ENGLAND', 'NORTHEAST'],
    'CT': ['NEW ENGLAND', 'NORTHEAST'],
    'RI': ['NEW ENGLAND', 'NORTHEAST'],



    'TX': ['WEST SOUTH CENTRAL', 'SOUTH'],
    'OK': ['WEST SOUTH CENTRAL', 'SOUTH'],
    'AR': ['WEST SOUTH CENTRAL', 'SOUTH'],
    'LA': ['WEST SOUTH CENTRAL', 'SOUTH'],

    'AL': ['EAST SOUTH CENTRAL', 'SOUTH'],
    'KY': ['EAST SOUTH CENTRAL', 'SOUTH'],
    'MS': ['EAST SOUTH CENTRAL', 'SOUTH'],
    'TN': ['EAST SOUTH CENTRAL', 'SOUTH'],

    'WV': ['SOUTH ATLANTIC', 'SOUTH'],
    'VA': ['SOUTH ATLANTIC', 'SOUTH'],
```

```python
    'MD': ['SOUTH ATLANTIC', 'SOUTH'],
    'DE': ['SOUTH ATLANTIC', 'SOUTH'],
    'NC': ['SOUTH ATLANTIC', 'SOUTH'],
    'SC': ['SOUTH ATLANTIC', 'SOUTH'],
    'GA': ['SOUTH ATLANTIC', 'SOUTH'],
    'FL': ['SOUTH ATLANTIC', 'SOUTH'],

}
```

In [30]:

```python
# creating a function that maps each state to its divison
def get_division(state):
    """
    Takes a state and uses the state_to dictionary to return its respective division.

    Parameters:
    state: a state's two letter abbreviation as a string

    Returns: the state's division

    """
    return state_to[state][0].title()

# creating a function that maps each state to its region
def get_region(state):
    """
    Takes a state and uses the state_to dictionary to return its respective region.

    Parameters:
    state: a state's two letter abbreviation as a string

    Returns: the state's region

    """
    return state_to[state][1].title()
```

In [32]:

```python
# applying get_divison and create a new column: division
redline['division'] = redline['state'].apply(get_division)

# applying get_region and create a new column: region
redline['region'] = redline['state'].apply(get_region)

# reorder columns
redline = redline[['metro_area', 'city', 'state', 'division', 'region', 'holc_grade', 'white_pop', 'black_pop', 'hisp_pop',
       'asian_pop', 'other_pop', 'total_pop', 'pct_white', 'pct_black',
       'pct_hisp', 'pct_asian', 'pct_other', 'lq_white', 'lq_black', 'lq_hisp',
       'lq_asian', 'lq_other', 'surr_area_white_pop', 'surr_area_black_pop',
       'surr_area_hisp_pop', 'surr_area_asian_pop', 'surr_area_other_pop',
       'surr_area_pct_white', 'surr_area_pct_black', 'surr_area_pct_hisp',
       'surr_area_pct_asian', 'surr_area_pct_other']]

# add an order to regions a general west -> east, north -> south for easier interpretation of plots
redline['region'] = pd.Categorical(redline['region'], categories=['West', 'Midwest', 'Northeast', 'South'], ordered=True)
```

In [33]:

```python
# adding divisions and regions to the metro_area_demographics dataset
new_metro_area_demographics['division']= new_metro_area_demographics.state.apply(get_division)
new_metro_area_demographics['region' ]= new_metro_area_demographics.state.apply(get_region)
new_metro_area_demographics
```

```
/var/folders/lp/wvqcc0x13jjb99jsy0mqncqr0000gn/T/ipykernel_21633/1547885690.py:2: 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

/var/folders/lp/wvqcc0x13jjb99jsy0mqncqr0000gn/T/ipykernel_21633/1547885690.py:3: 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
```

Out[33]:

	metro_area	city	state	state_lq_variance	division	region
0	Akron, OH	Akron	OH	0.233570	East North Central	Midwest
1	Albany-Schenectady-Troy, NY	Albany	NY	0.406110	Middle Atlantic	Northeast
2	Allentown-Bethlehem-Easton, PA-NJ	Allentown	PA	0.059175	Middle Atlantic	Northeast
3	Altoona, PA	Altoona	PA	0.149715	Middle Atlantic	Northeast
4	Amarillo, TX	Amarillo	TX	0.190055	West South Central	South
...	...	...	...	...	...	...
133	Wheeling, WV-OH	Wheeling	WV	0.241125	South Atlantic	South
134	Wichita, KS	Wichita	KS	0.136530	West North Central	Midwest
135	Winston-Salem, NC	Winston	NC	0.321435	South Atlantic	South
136	York-Hanover, PA	York	PA	1.003470	Middle Atlantic	Northeast
137	Youngstown-Warren-Boardman, OH-PA	Youngstown	OH	0.642940	East North Central	Midwest

137 rows × 6 columns

Division and Regional Segregation

After creating the division and regional columns, I grouped by division and region, averaging them out to get each respective one's LQ variance.

In [34]:
```
# create a table of division demographic inequity

division_demographics = new_metro_area_demographics.drop(['metro_area', 'city', 'state', 'region'], axis = 1).groupby('division').mean().rese
division_demographics.rename({'state_lq_variance':'division_lq_variance'}, axis = 1, inplace = True)

# create a table of region demographic inequity

region_demographics = new_metro_area_demographics.drop(['metro_area', 'city', 'state', 'division'], axis = 1).groupby('region').mean().reset_
region_demographics.rename({'state_lq_variance':'region_lq_variance'}, axis = 1, inplace = True)
```

In [35]:
```
# to plot by division/region, apply the get_division and get_region functions to make new columns by state

temp_state_dem = new_state_demographics
temp_state_dem['division'] = temp_state_dem.state.apply(get_division)
temp_state_dem['region'] = temp_state_dem.state.apply(get_region)

# create dictionaries where divisons/regions are the keys and division lq variance are the values
division_values = dict(division_demographics.sort_values(by = 'division_lq_variance', ascending = True).values)
region_values = dict(region_demographics.sort_values(by = 'region_lq_variance', ascending = True).values)

# create a list of the states from the temp_state_dem dataset
state_division = temp_state_dem['division'].to_list()
state_region = temp_state_dem['region'].to_list()

# loop through the divisions, and append its respective division lq variance to a list, create a new column from that list
temp_lst = []
for division in state_division:
    temp_lst.append(division_values[division])
temp_state_dem['division_lq_variance'] = temp_lst

# loop through the divisions, and append its respective division lq variance to a list, create a new column from that list
temp_lst = []
for region in state_region:
    temp_lst.append(region_values[region])
temp_state_dem['region_lq_variance'] = temp_lst

temp_state_dem.head()
```

Out[35]:

	state	state_lq_variance	division	region	division_lq_variance	region_lq_variance
0	AL	0.322798	East South Central	South	0.273930	0.245941
1	AR	0.218440	West South Central	South	0.224088	0.245941
2	AZ	0.170895	Mountain	West	0.113088	0.197225
3	CA	0.297331	Pacific	West	0.239293	0.197225
4	CO	0.086015	Mountain	West	0.113088	0.197225

In terms of division racial segregation, the East South Central (0.273930) and Middle Atlantic (0.262642) are the worst while the West North Central (0.121754) and Mountain (0.113088) are the best.

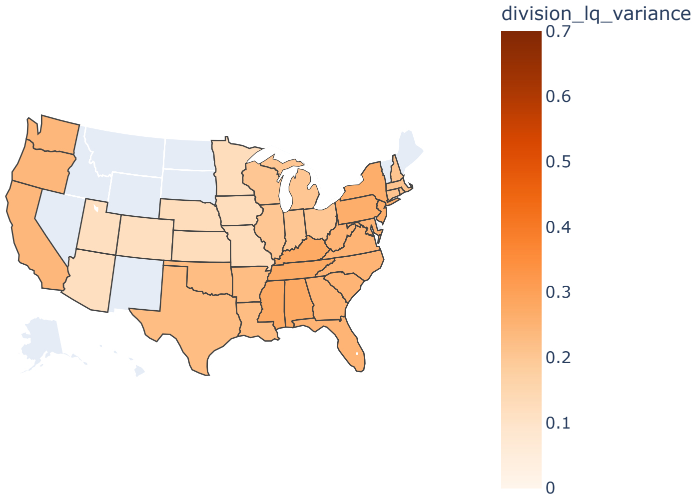
In [36]:
```
division_demographics.sort_values(by = 'division_lq_variance', ascending = True)
```

Out[36]:

	division	division_lq_variance
3	Mountain	0.113088
7	West North Central	0.121754
0	East North Central	0.200134
4	New England	0.204770
8	West South Central	0.224088
5	Pacific	0.239293
6	South Atlantic	0.246142
2	Middle Atlantic	0.262642
1	East South Central	0.273930

In [37]:
```
# plot division lq variances
fig = px.choropleth(temp_state_dem,
                    locations='state',
                    locationmode='USA-states',  # This still works for regions
                    hover_name='division',  # Display region name on hover
                    color='division_lq_variance',
                    scope="usa",  # Focus on the USA
                    color_continuous_scale='Oranges',
                    range_color=[0, 0.7])

fig.update_layout(title_text='Demographic Inequity by Division (Without VW Metro Area)', geo_scope='usa')

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

Demographic Inequity by Division (Without VW Metro Area)



In terms of regional racial segregation, the Midwest was the least segregated (0.175052) and the Northeast was the most segregated (0.248174), closely followed by the South (0.245941).

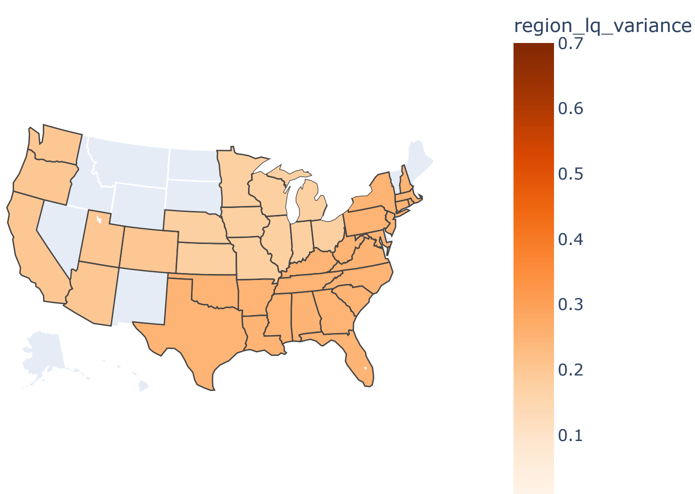
In [38]:
```
region_demographics.sort_values(by = 'region_lq_variance', ascending = True)
```

Out[38]:

	region	region_lq_variance
0	Midwest	0.175052
3	West	0.197225
2	South	0.245941
1	Northeast	0.248174

In [39]:
```
# plot region lq variances
fig = px.choropleth(temp_state_dem,
                    locations='state',
                    locationmode='USA-states',  # This still works for regions
                    hover_name='region',  # Display region name on hover
                    color='region_lq_variance',
                    scope="usa",  # Focus on the USA
                    color_continuous_scale='Oranges',
                    range_color=[0, 0.7])

fig.update_layout(title_text='Demographic Inequity by Region (Without VW Metro Area)', geo_scope='usa')

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

Demographic Inequity by Region (Without VW Metro Area)



Conclusions and Discussion

Based off the data, it is clear that the impacts of redlining are still felt today - previously redlined areas are still segregated the way they were intended to be back in the 1930s. My hypothesis (areas with histories of slavery would be more segregated) was disproven - the Northern region was actually slightly more segregated than the Southern region. A potential explanation for this would be laws enacted in response to the Great Migration from the 1910s-1970s.[4] During the Great Migration, approximately six million Black people moved from the American South to escape racial violence, pursue economic and educational opportunities, and obtain freedom from Jim Crow Laws. However, they were met with resitance, and faced injustices and difficulties after migrating. Redlining laws in part, emerged as a result of the influx of Black folks in predominately White areas, leading both the North and South to most significantly still show the effects of this.

Limitations

- The data set used only 38 contains states, in particular, it lacks states from Mountain Division, leading to an incomplete picture of the full scope of the impacts of redlining.
- Some metro areas cross states, reducing the accuracy of regional judgements. Currently, the maps created in this project are not as accurate as they could potentially be
- The “Other” Demographic group captures a huge range of ethnicities and groups. Additionally, there are limitations to the 2020 Census data due to it being collected during the COVID-19 pandemic. This may lead to the data collected not being entirely accurate.
- The issue of Huntington, West Virgina is unresolved - if the LQ score was an actual data point, rather than an error of some kind, it would significantly change what states, divisions, and regions are the most segregated.

Further Research

This current project provides a prelimary basis that can be expanded upon to further understanding of the impacts of redlining. Future projects can use GIS software to produce better and more accurate maps that would better reflect a metro-area that isn't forced into a particular state boundary. Additionally, it would be interesting to pair this demographic data with HOLC grades with other issues such as food insecurity, social vulnerability, life expectancy, etc. This data could also be mapped across time using census data from other years to examine how segregation patterns have changed across time.

References

1. ^ Cornell Law School
2. ^ NPR Fresh Race: A 'Forgotten History' Of How The U.S. Government Segregated America
3. ^ Federal Reserve History: Redlining
4. ^ The Great Migration (1910-1970)