Michelle Helfman Term Project Milestone 5

Moving Starter Kit

The Moving Starter Kit contains basic demographic, economic, education, and additional location-based information to be used as a starting point to finding a new city to live or confirm the current location is the best place to be.

This combines the data from the previous milestones into 1 final dataset.

In [1]:
```
# Import Functions/Libraries

import pandas as pd
import numpy as np
import urllib
import os
import pyodbc
import sqlalchemy

#from scipy import stats
from sqlalchemy import create_engine
from sqlalchemy import text as sa_text
from sqlalchemy import Table, MetaData, Column, Integer, select

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Read in the Moving Starter Kit files.
# Delete output file.

# Read Demographics file
MSK_demographics_df = pd.read_excel('MSK Milestone 2.xlsx')

# Read Regions file
MSK_regions_df = pd.read_excel('MSK Milestone 3.xlsx')

# Read Weather file
MSK_weather_df = pd.read_excel('MSK Milestone 4.xlsx')

# Delete the existing output file.
file = 'MSK Milestone 5.xlsx'
location = "C:/DSC540_Data/"
path = os.path.join(location, file)

# Remove the file
try:
    os.remove(path)

except:
    print('No Prior File Deleted')
```

In [3]:
```
# Drop Unnecessary 1st columns and
# rename columns where necessary

# Demographics table
# Delete Unnamed: 0 column from Regions
del MSK_demographics_df['Unnamed: 0']

# Rename Demographics DF Columns
new_demo_headers = {'Retirement_Quality_of_Life Ranking': 'Retirement_Quality_of_Life_Ranking'}
MSK_demographics_df.rename(columns = new_demo_headers, inplace = True)

# Regions table
# Delete Unnamed: 0 column from Regions
del MSK_regions_df['Unnamed: 0']

# Rename Regions DF Columns
new_region_headers = {'State Name': 'State_Name', 'Region': 'Region',
                      'State Capital': 'State_Capital',
                      'Capital and State': 'Capital_and_State',
                      'Capital and Abbrev': 'Capital_and_Abbrev',
                      'Create Date': 'Create_Date'}
MSK_regions_df.rename(columns = new_region_headers, inplace = True)

# Weather table
# Delete Unnamed: 0 column from Weather
del MSK_weather_df['Unnamed: 0']
```

In [4]:
```
# Set up to use SQLAlchemy to access the database
# Drop existing tables

# Connect to SQL Server
db_conn = 'mssql+pyodbc://DESKTOP-L6D2PBJ/DSC540_Data_Preparation?driver=SQL+Server'
engine = create_engine(db_conn)
try:
    conn = engine.connect()
    print("Passed")

except Exception as e:
    print(e)

# Delete the tables if they exist.
engine.execute(sa_text('''DROP TABLE IF EXISTS MSK_demographics''').execution_options(autocommit=True))
engine.execute(sa_text('''DROP TABLE IF EXISTS MSK_regions''').execution_options(autocommit=True))
engine.execute(sa_text('''DROP TABLE IF EXISTS MSK_weather''').execution_options(autocommit=True))
```

```
Passed
```

Out[4]:
```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x1830efc5b80>
```

In [5]:
```
# Create Moving Starter Kit tables on the SQL Server

# Create demographics table
demographics_table = sa_text('CREATE TABLE MSK_demographics ' +
                             '(Metropolitan_Area varchar(100), ' +
                             'Metropolitan_Short varchar(100), ' +
                             'State varchar(20), ' +
                             'State_Code char(2), ' +
                             'Total_Population int, ' +
                             'Anchor_City varchar(50), ' +
                             'Anchor_City_Population int, ' +
                             'Median_Age numeric(4,1), ' +
                             'Male_PCT numeric(4,1), ' +
                             'Female_PCT numeric(4,1), ' +
                             'White_PCT numeric(4,1), ' +
                             'Black_PCT numeric(4,1), ' +
                             'Asian_PCT numeric(4,1), ' +
                             'Latino_PCT numeric(4,1), ' +
                             'American_Indian_Alaska_Native_PCT numeric(4,1), ' +
                             'Pacific_Islander_PCT numeric(4,1), ' +
                             'Mean_Income int, ' +
                             'Employment_PCT numeric(4,1), ' +
                             'High_School_Grad_Rate numeric(4,1), ' +
                             'College_Degree_PCT numeric(4,1), ' +
                             'Education_State_Ranking int, ' +
                             'Education_Quality_State_Ranking int, ' +
                             'Number_of_Airports int, ' +
                             'Income_Tax_Rate_Low numeric(4,2), ' +
                             'Income_Tax_Rate_High numeric(4,2), ' +
                             'State_Retirement_Ranking int, ' +
                             'Retirement_Affordability_Ranking int, ' +
                             'Retirement_Quality_of_Life_Ranking int, ' +
                             'Retirement_Health_Care_Ranking int, ' +
                             'Homes_With_Internet_PCT numeric(4,1), ' +
                             'Homes_Without_Internet_PCT numeric(4,1), ' +
                             'Violent_Crime_2019 int, ' +
                             'Property_Crime_2019 int, ' +
                             'Metro_Micro_Area varchar(20))')

# Check Results
demographics_results = engine.execute(demographics_table)
print('demographics', demographics_results)


# Create regions table
regions_table = sa_text('CREATE TABLE MSK_regions ' +
                        '(Abbreviation char(2), ' +
                        'State_Name varchar(20), ' +
                        'Region varchar(10), ' +
                        'State_Capital varchar(50), ' +
                        'Capital_and_State varchar(50), ' +
                        'Capital_and_Abbrev varchar(50), ' +
                        'Create_Date datetime)')

# Check Results
regions_results = engine.execute(regions_table)
print('regions', regions_results)

# Create weather table
weather_table = sa_text('CREATE TABLE MSK_weather ' +
                        '(metro_short varchar(100), ' +
                        'longitude char(8), ' +
                        'latitude char(7), ' +
                        'forecast_today varchar(200), ' +
                        'forecast_tom varchar(200), ' +
                        'state_code char(2), ' +
                        'anchor_city varchar(50), ' +
                        'create_date datetime)')

# Check Results
weather_results = engine.execute(weather_table)
```

```
print('weather', weather_results)
```

```
demographics <sqlalchemy.engine.cursor.LegacyCursorResult object at 0x000001830EF95640>
regions <sqlalchemy.engine.cursor.LegacyCursorResult object at 0x000001830F31E250>
weather <sqlalchemy.engine.cursor.LegacyCursorResult object at 0x000001830F320160>
```

In [6]:
```
# Upload files to SQL Server

# Upload the Demographics file
try:
    d_tableToWrite= 'MSK_demographics'
    MSK_demographics_df.to_sql(name = d_tableToWrite, con= engine, if_exists='append', index=False)
except Exception as e:
    print(e)

# Upload the Regions file
try:
    r_tableToWrite = 'MSK_regions'
    MSK_regions_df.to_sql(name = r_tableToWrite, con= engine, if_exists='append', index=False)
except Exception as e:
    print(e)

# Upload the Weather file
try:
    w_tableToWrite= 'MSK_weather'
    MSK_weather_df.to_sql(name = w_tableToWrite, con= engine, if_exists='append', index=False)
except Exception as e:
    print(e)
```

In [7]:

```
# Read the 3 datasets/tables and combine the information into 1 dataset.
# Adding the prefix d(demographics), r(regions), and w(weather) to
# signify which table the column comes from.

sqlstr = ('Select Metropolitan_Area as d_Metropolitan_Area, ' +
          'Metropolitan_Short as d_Metropolitan_Short, ' +
          'd.State as d_State, ' +
          'd.State_Code as d_State_Code, ' +
          'Total_Population as d_Total_Population, ' +
          'd.Anchor_City as d_Anchor_City, ' +
          'Anchor_City_Population as d_Anchor_City_Population, ' +
          'Median_Age as d_Median_Age, ' +
          'Male_PCT as d_Male_PCT, ' +
          'Female_PCT as d_Female_PCT, ' +
          'White_PCT as d_White_PCT, ' +
          'Black_PCT as d_Black_PCT, ' +
          'Asian_PCT as d_Asian_PCT, ' +
          'Latino_PCT as d_Latino_PCT, ' +
          'American_Indian_Alaska_Native_PCT as d_American_Indian_Alaska_Native_PCT, ' +
          'Pacific_Islander_PCT as d_Pacific_Islander_PCT, ' +
          'Mean_Income as d_Mean_Income, ' +
          'Employment_PCT as d_Employment_PCT, ' +
          'High_School_Grad_Rate as d_High_School_Grad_Rate, ' +
          'College_Degree_PCT as d_College_Degree_PCT, ' +
          'Education_State_Ranking as d_Education_State_Ranking, ' +
          'Education_Quality_State_Ranking as d_Education_Quality_State_Ranking, ' +
          'Number_of_Airports as d_Number_of_Airports, ' +
          'Income_Tax_Rate_Low as d_Income_Tax_Rate_Low, ' +
          'Income_Tax_Rate_High as d_Income_Tax_Rate_High, ' +
          'State_Retirement_Ranking as d_State_Retirement_Ranking, ' +
          'Retirement_Affordability_Ranking as d_Retirement_Affordability_Ranking, ' +
          'Retirement_Quality_of_Life_Ranking as d_Retirement_Quality_of_Life_Ranking, ' +
          'Retirement_Health_Care_Ranking as d_Retirement_Health_Care_Ranking, ' +
          'Homes_With_Internet_PCT as d_Homes_With_Internet_PCT, ' +
          'Homes_Without_Internet_PCT as d_Homes_Without_Internet_PCT, ' +
          'Violent_Crime_2019 as d_Violent_Crime_2019, ' +
          'Property_Crime_2019 as d_Property_Crime_2019, ' +
          'Metro_Micro_Area as d_Metro_Micro_Area, ' +
          'Region as r_Region, ' +
          'State_Capital as r_State_Capital, ' +
          'Capital_and_State as r_Capital_and_State, ' +
          'Capital_and_Abbrev as r_Capital_and_Abbrev, ' +
          'longitude as w_longitude, ' +
          'latitude as w_latitude, ' +
          'forecast_today as w_forecast_today, ' +
          'forecast_tom as w_forecast_tom, ' +
          'getdate() as file_create_date ' +
          'from MSK_demographics d, ' +
          'MSK_regions r, ' +
          'MSK_weather w ' +
          'where d.State_Code = r.Abbreviation ' +
          'and d.State_Code = w.state_code ' +
          'and d.Anchor_City = w.anchor_city')

try:
    MSK_data = pd.read_sql(sql = sqlstr, con = db_conn)

except Exception as e:
    print(e)

MSK_data.head()
```

Out[7]:

	d_Metropolitan_Area	d_Metropolitan_Short	d_State	d_State_Code	d_Total_Population	d_Anchor_City	d_Anchor_City_Populatior
0	Philadelphia-Camden-Wilmington, PA-NJ-DE-MD Me...	Philadelphia-Camden-Wilmington, PA-NJ-DE-MD	Delaware	DE	6228601	Philadelphia	1576251
1	Urban Honolulu, HI Metro Area	Urban Honolulu, HI	Hawaii	HI	1000890	Urban Honolulu	0
2	Huntington-Ashland, WV-KY-OH Metro Area	Huntington-Ashland, WV-KY-OH	Kentucky	KY	356581	Huntington	46025
3	Anchorage, AK Metro Area	Anchorage, AK	Alaska	AK	398807	Anchorage	0
4	Salinas, CA Metro Area	Salinas, CA	California	CA	437325	Salinas	162791

5 rows × 43 columns

In [8]:
```
# Close engine
try:
    engine.dispose()
    print("Closed")

except:
    print("failed!")
```

```
Closed
```

In [9]:
```
# Write to excel file

# Sort the information
MSK_data = MSK_data.sort_values('d_Metropolitan_Short')

# Write out the MSK Information
MSK_data.to_excel("C:/DSC540_Data/MSK Milestone 5.xlsx",
                  sheet_name = 'MSK 5', index = False)

print('The End')
```

```
The End
```