Final Project: Get Data

This script is meant to proceed the Clean_Data scipt. This tool removes extracts location information from Google Places API of existing and competing businesses for Costco, and extracts data from the Minnesota Geospatial Commons API.

In [ ]:
```
import requests
import pandas as pd
from zipfile import ZipFile
import os

import pprint
import json
```

Find existing Costco location using Google Places API

In [ ]:
```
def format_findsearch(in_search):
    ''' Remove spaces from find place from text query search location
    and format for google places api url

    Parameter
    ---------
    in_search: str
        address, name, or phone number of search location

    Return
    ------
    out: str
        formatted search location to be used for google places api url
    '''

    out = in_search.replace(",", "")
    out = out.replace(" ", "%20")
    return out
```

In [ ]:
```
key = "AIzaSyCYPFFiQg2gvFhLwv17r9FEjJalSiqwNrM"
```

In [ ]:
```
search = "costco"
```

In [ ]:
```
base_url = "https://maps.googleapis.com/maps/api/place/"
query_type = "textsearch"
out_type = "json"

inputtype = "textquery"
fields = "fields=formatted_address,name,geometry"

place = format_findsearch(search)
url = f"{base_url}{query_type}/{out_type}?input={place}&inputtype={inputtype}&
{fields}&key={key}"
```

In [ ]:
```
r = requests.get(url)
assert r.status_code is 200
out_search = r.json()
```

In [ ]:
```
names = []
address = []
lat = []
long = []
for item in out_search['results']:
    names.append(item['name'])
    address.append(item['formatted_address'])
    lat.append(item['geometry']['location']['lat'])
    long.append(item['geometry']['location']['lng'])
```

In [ ]:
```
costco_loc = pd.DataFrame(
    {"name": names,
     "address": address,
     "lat": lat,
     "long": long})
```

In [ ]:
```
out_dir = r"C:\Users\msong\Desktop\arc2proj\output_data"
costco_loc.to_csv(os.path.join(out_dir, "costco_loc.csv"), index=False)
```

In [ ]:
```
# create point features of costco locations
out_coord_system = arcpy.SpatialReference('WGS 1984')
arcpy.management.XYTableToPoint(r"C:\Users\msong\Desktop\arc2proj\output_data
\costco_loc.csv",
                                r"C:\Users\msong\Desktop\arc2proj\Business_Fuz
zyLogic\temp.gdb\costcos",
                                "long",
                                "lat",
                                None,
                                out_coord_system)
```

In [ ]:
```
# reproject to NAD 1983 UTM 15N
out_coord_system = arcpy.SpatialReference("NAD 1983 UTM ZONE 15N")

arcpy.management.Project(r"C:\Users\msong\Desktop\arc2proj\Business_FuzzyLogic
\temp.gdb\costcos",
                         r"C:\Users\msong\Desktop\arc2proj\Business_FuzzyLogic
\temp.gdb\costcos_projected",
                         out_coord_system,
                         "WGS_1984_(ITRF00)_To_NAD_1983")
```

In [ ]:
```
# import census tracts into geodatabase
arcpy.conversion.FeatureClassToFeatureClass(r"C:\Users\msong\Desktop\arc2proj
\data\Census2010RealignTract.shp",
                                            r"C:\Users\msong\Desktop\arc2proj
\Business_FuzzyLogic\Business_FuzzyLogic.gdb",
                                            "metro_tracts")
```

In [ ]:
```
# clip to Metropolitan Counties
arcpy.analysis.Clip(r"C:\Users\msong\Desktop\arc2proj\Business_FuzzyLogic\tem
p.gdb\costcos_projected",
                    "metro_tracts",
                    "costcos_metro",
                    None)
```

In [ ]:
```

```

In [ ]:
```

```

Get locations of competing stores

In [ ]:
```
# Find locations of competing businesses to costco
search_list = ["Target Store Minnesota", "Walmart Store Minnesota", "Sam's Clu
b Minnesota"]
names = []
address = []
lat = []
long = []

for item in search_list:
    search = item
    base_url = "https://maps.googleapis.com/maps/api/place/"
    query_type = "textsearch"
    out_type = "json"

    inputtype = "textquery"
    fields = "fields=formatted_address,name,geometry"

    place = format_findsearch(search)
    url = f"{base_url}{query_type}/{out_type}?input={place}&inputtype={inputty
pe}&{fields}&key={key}"

    r = requests.get(url)
    assert r.status_code is 200
    out_search = r.json()

    for item in out_search['results']:
        names.append(item['name'])
        address.append(item['formatted_address'])
        lat.append(item['geometry']['location']['lat'])
        long.append(item['geometry']['location']['lng'])
```

In [ ]:
```
competing_stores = pd.DataFrame(
    {"name": names,
     "address": address,
     "lat": lat,
     "long": long})
```

In [ ]:
```
out_dir = r"C:\Users\msong\Desktop\arc2proj\output_data"
competing_stores.to_csv(os.path.join(out_dir, "competing.csv"), index=False)
```

In [ ]:
```
# create point features of competing locations
out_coord_system = arcpy.SpatialReference('WGS 1984')
arcpy.management.XYTableToPoint(r"C:\Users\msong\Desktop\arc2proj\output_data
\competing.csv",
                                r"C:\Users\msong\Desktop\arc2proj\Business_Fuz
zyLogic\temp.gdb\competing_stores",
                                "long",
                                "lat",
                                None,
                                out_coord_system)
```

In [ ]:
```
out_coord_system = arcpy.SpatialReference("NAD 1983 UTM ZONE 15N")

arcpy.management.Project(r"C:\Users\msong\Desktop\arc2proj\Business_FuzzyLogic
\temp.gdb\competing_stores",
                         r"C:\Users\msong\Desktop\arc2proj\Business_FuzzyLogic
\temp.gdb\competing_projected",
                         out_coord_system,
                         "WGS_1984_(ITRF00)_To_NAD_1983")
```

In [ ]:
```
# clip to Metropolitan Counties
arcpy.analysis.Clip(r"C:\Users\msong\Desktop\arc2proj\Business_FuzzyLogic\tem
p.gdb\competing_projected",
                    "metro_tracts",
                    "competing_metro",
                    None)
```

In [ ]:
```

```

Get MN Major Roads and Highways

In [ ]:
```
roads_url = r"https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_m
etc/trans_fnctnl_cls_rds/shp_trans_fnctnl_cls_rds.zip"
r = requests.get(roads_url)
assert r.status_code is 200

with open(os.path.join(out_path, "roads.zip"), "wb") as file:
    file.write(r.content)

with ZipFile(os.path.join(out_path, "roads.zip"), "r") as zipped:
    zipped.extractall(out_path)
```

Get planned land use in Metropolitan area, Minnesota

In [ ]:
```
LU_url = r"https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_met
c/plan_pland_land_use/shp_plan_pland_land_use.zip"
r = requests.get(LU_url)
assert r.status_code is 200

with open(os.path.join(out_path, "plannedLU.zip"), "wb") as file:
    file.write(r.content)

with ZipFile(os.path.join(out_path, "plannedLU.zip"), "r") as zipped:
    zipped.extractall(out_path)
```

In [ ]:
```
arcpy.conversion.FeatureClassToFeatureClass(r"C:\Users\msong\Desktop\arc2proj
\data\plannedLU\PlannedLandUse.shp",
                                            r"C:\Users\msong\Desktop\arc2proj
\Business_FuzzyLogic\Business_FuzzyLogic.gdb",
                                            "planned_LU")
```

Get census tracts

In [ ]:
```
tracts_url = r"https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_
metc/society_census2010realign/shp_society_census2010realign.zip"
r = requests.get(tracts_url)
assert r.status_code is 200

with open(os.path.join(out_path, "census_tracts.zip"), "wb") as file:
    file.write(r.content)

with ZipFile(os.path.join(out_path, "census_tracts.zip"), "r") as zipped:
    zipped.extractall(out_path)
```

In [ ]:
```
# I wasn't able to figure out the IPUMS API, but this was what I had.
#IPUMS API KEY
my_key = "" # removed key

my_headers = {"Authorization": my_key}
url = "https://api.ipums.org/extracts/?product=nhgis&version=v1"
```

In [ ]:
```
er = """

{
  "datasets": {
    "2015_2019_ACS5a": {
      "years": ["2019"],
      "breakdown_values": ["bs30.si0762", "bs30.si2026"],
      "data_tables": [
        "B02001"
      ],
      "geog_levels": [
        "census tract"
      ]
    }
  },
  "data_format": "csv_no_header",
  "description": "sample6",
  "breakdown_and_data_type_layout": "single_file"
}

"""
result = requests.post(url, headers=my_headers, json=json.loads(er))
my_extract_number = result.json()["number"]
print(my_extract_number)

# Results
```

In [ ]:
```
my_headers = {"Authorization": my_key}
dataset = "B02001"
url = "https://api.ipums.org/metadata/nhgis/datasets/1990_STF1/data_tables/NP19?version=v1"
nhgis_metadata = requests.get(url, headers=my_headers)
pprint(nhgis_metadata.json())
```

In [ ]:
```

```