```
# Grp 3 Project Code by Mychael Solis-Wheeler, Celina Turner, & Mike Owona Ongbwa
04/24/2019

# Import pandas
import pandas as pd

# Note: Each headers in each dataset were checked with spaces removed in each
# Additionally, each dataset was modified to include zipcodes and longtitude
# & latitude if not already present. Furthermore, commas were removed in total
# population column in neighborhood file.

# Variables needed for ease of file access
file_1 = 'Affordable_Housing_Directory_2019.csv'
file_2 = 'Neighborhood_Zipcodes_FULL_2010.csv'
file_3 = 'Austin_Google_FiberGB_Speed_Internet_Locations_2019.csv'
file_4 = 'Austin_Public_Health_Locations_2019.csv'
file_5 = 'Austin_HS_Grad_Rates_2016.csv'
outpath = '/Users/mykeysw/Desktop/Grp3ProjectFolder/' # May need to update outpath
based zipfile downloaded
fileout1 = 'Austin_Public_Sites.csv'
fileout2 = 'Austin_Affordable_Housing_Demographics.csv'

# Use panda data frames to read data from ALL csv files
dfhousing = pd.read_csv(outpath + file_1)
dfhoods = pd.read_csv(outpath + file_2)
dfinternet = pd.read_csv(outpath + file_3)
dfhealth = pd.read_csv(outpath + file_4)
dfschools = pd.read_csv(outpath + file_5)

# Check how data is organized in ALL files
dfhousing.head(10)
dfhoods.head(10)
dfinternet.head(10)
dfhealth.head(10)
dfschools.head(10)
# No delimiters found in ANY files
#-------------------------------------------------------------------------------
-

# Check for any nulls, fill in nulls with 0's, & check ALL file counts again
dfhousing.count() # 3 columns have nulls in housing file
dfhousing['PropertyManagementCompany'].fillna('Other', inplace=True)
dfhousing['Phone'].fillna(0, inplace=True)
dfhousing['Website'].fillna(0, inplace=True)
dfhousing.count() # All nulls filled

dfhoods.count() # No nulls found in neighborhood file

dfinternet.count() # 6 columns have nulls in internet location file
dfinternet['ConnectionStatus'].fillna(0, inplace=True)
dfinternet['ConnectionColor'].fillna(0, inplace=True)
dfinternet['LocationName'].fillna(0, inplace=True)
dfinternet['FileName'].fillna(0, inplace=True)
dfinternet['CouncilDistrict'].fillna(0, inplace=True)
dfinternet['ApplicationDocument'].fillna(0, inplace=True)
dfinternet.count()  # All nulls filled

dfhealth.count() # 11 columns have nulls in public health file
dfhealth['Hours'].fillna(0, inplace=True)
```

```
dfhealth['Website'].fillna(0, inplace=True)
dfhealth['PhoneNumber'].fillna(0, inplace=True)
dfhealth['OtherPhone'].fillna(0, inplace=True)
dfhealth['BuildingID'].fillna(0, inplace=True)
dfhealth['OwnershipStatus'].fillna(0, inplace=True)
dfhealth['Owner'].fillna('Other', inplace=True)
dfhealth['OccupyingDivision'].fillna(0, inplace=True)
dfhealth['OccupancyType'].fillna('Other', inplace=True)
dfhealth['SqFt'].fillna(0, inplace=True)
dfhealth['YearBuilt'].fillna(0, inplace=True)
dfhealth.count() # All nulls filled

dfschools.count() # 13 columns have nulls in school file
dfschools['District'].fillna(0, inplace=True)
dfschools['2015ClassSize'].fillna(0, inplace=True)
dfschools['2015Graduated'].fillna(0, inplace=True)
dfschools['2015Rate'].fillna(0, inplace=True)
dfschools['2014Graduated'].fillna(0, inplace=True)
dfschools['2014ClassSize'].fillna(0, inplace=True)
dfschools['2014Rate'].fillna(0, inplace=True)
dfschools['2013ClassSize'].fillna(0, inplace=True)
dfschools['2013Graduated'].fillna(0, inplace=True)
dfschools['2013Rate'].fillna(0, inplace=True)
dfschools['2012ClassSize'].fillna(0, inplace=True)
dfschools['2012Graduated'].fillna(0, inplace=True)
dfschools['2012Rate'].fillna(0, inplace=True)
dfschools.count() # All nulls filled
#-------------------------------------------------------------------------------
-

# Drop columns in each file not needed & add 1 column to internet file
dfhousing.count() # Look at housing file count to select columns to drop
dfhousing.drop(columns= ['ProjectName'], inplace=True, axis = 1)
dfhousing.drop(columns= ['Address'], inplace=True, axis = 1)
dfhousing.drop(columns= ['TotalUnits'], inplace=True, axis = 1)
dfhousing.drop(columns= ['HousingType'], inplace=True, axis = 1)
dfhousing.drop(columns= ['Status'], inplace=True, axis = 1)
dfhousing.drop(columns= ['Phone'], inplace=True, axis = 1)
dfhousing.drop(columns= ['Website'], inplace=True, axis = 1)
dfhousing.drop(columns= ['Location'], inplace=True, axis = 1)
dfhousing.count() # Look at remaining columns to ensure Zipcode, Total Afforable
Units,
# Unit Type, 1 Person Household, 2 Person Household, 3 Person Household,
# 4 Person Household, 5 Person Household, Longitude, Latitude,
# & Property Management Company present

# Drop columns in each file not needed
dfhoods.count() # Look at housing file count to select columns to drop
dfhoods.drop(columns= ['Neighborhood'], inplace=True, axis = 1)
dfhoods.drop(columns= ['OccupiedHousingUnits'], inplace=True, axis = 1)
dfhoods.drop(columns= ['VacantHousingUnits'], inplace=True, axis = 1)
dfhoods.drop(columns= ['PercentOwnerOccupiedHousingUnits'], inplace=True, axis = 1)
dfhoods.drop(columns= ['PerPerAcre'], inplace=True, axis = 1)
dfhoods.drop(columns= ['DensityRanking'], inplace=True, axis = 1)
dfhoods.drop(columns= ['Acres'], inplace=True, axis = 1)
dfhoods.count() # Look at remaining columns to ensure Zipcode, Total Population,
# Non-Hispanic White Percentage, African American Percentage, Hispanic Percentage,
# Other Percentage, & Total Housing Units present
```

```
dfinternet.count() # Look at internet file count to select columns to drop
dfinternet.drop(columns= ['Category'], axis=1, inplace=True)
dfinternet.drop(columns= ['ConnectionStatus'], axis=1, inplace=True)
dfinternet.drop(columns= ['ConnectionColor'], axis=1, inplace=True)
dfinternet.drop(columns= ['LocationName'], axis=1, inplace=True)
dfinternet.drop(columns= ['CouncilDistrict'], axis=1, inplace=True)
dfinternet.drop(columns= ['LocationAddress'], axis=1, inplace=True)
dfinternet.drop(columns= ['LocationCity'], axis=1, inplace=True)
dfinternet.drop(columns= ['LocationState'], axis=1, inplace=True)
dfinternet.drop(columns= ['LocationCounty'], axis=1, inplace=True)
dfinternet.drop(columns= ['FileName'], axis=1, inplace=True)
dfinternet.drop(columns= ['ApplicationDocument'], axis=1, inplace=True)
dfinternet.drop(columns= ['Location'], axis=1, inplace=True)
# Create new column 'Internet Type' into internet file
dfinternet['InternetType'] = 'PublicInternet'
dfinternet.count()  # Look at remaining columns to ensure Zipcode, Organization
Name,
# Location, & Internet Type present

dfschools.count()  # Look at school file count to select columns to drop
dfschools.drop(columns= ['District'], inplace=True, axis = 1)
dfschools.drop(columns= ['StreetAddress'], inplace=True, axis = 1)
dfschools.drop(columns= ['2016Graduated'], inplace=True, axis = 1)
dfschools.drop(columns= ['2016ClassSize'], inplace=True, axis = 1)
dfschools.drop(columns= ['2016Rate'], inplace=True, axis = 1)
dfschools.drop(columns= ['2015ClassSize'], inplace=True, axis = 1)
dfschools.drop(columns= ['2015Graduated'], inplace=True, axis = 1)
dfschools.drop(columns= ['2015Rate'], inplace=True, axis = 1)
dfschools.drop(columns= ['2014ClassSize'], inplace=True, axis = 1)
dfschools.drop(columns= ['2014Rate'], inplace=True, axis = 1)
dfschools.drop(columns= ['2014Graduated'], inplace=True, axis = 1)
dfschools.drop(columns= ['2013ClassSize'], inplace=True, axis = 1)
dfschools.drop(columns= ['2013Rate'], inplace=True, axis = 1)
dfschools.drop(columns= ['2013Graduated'], inplace=True, axis = 1)
dfschools.drop(columns= ['2012ClassSize'], inplace=True, axis = 1)
dfschools.drop(columns= ['2012Rate'], inplace=True, axis = 1)
dfschools.drop(columns= ['2012Graduated'], inplace=True, axis = 1)
dfschools.drop(columns= ['Location'], inplace=True, axis = 1)
dfschools.count()  # Look at remaining columns to ensure School Type, School,
# Zipcode, Longitude, & Latitude present

dfhealth.count()  # Look at school file count to select columns to drop
dfhealth.drop(columns= ['StreetAddress'], inplace=True, axis = 1)
dfhealth.drop(columns= ['Hours'], inplace=True, axis = 1)
dfhealth.drop(columns= ['Website'], inplace=True, axis = 1)
dfhealth.drop(columns= ['PhoneNumber'], inplace=True, axis = 1)
dfhealth.drop(columns= ['OtherPhone'], inplace=True, axis = 1)
dfhealth.drop(columns= ['BuildingID'], inplace=True, axis = 1)
dfhealth.drop(columns= ['OwnershipStatus'], inplace=True, axis = 1)
dfhealth.drop(columns= ['Owner'], inplace=True, axis = 1)
dfhealth.drop(columns= ['OccupyingDivision'], inplace=True, axis = 1)
dfhealth.drop(columns= ['SqFt'], inplace=True, axis = 1)
dfhealth.drop(columns= ['YearBuilt'], inplace=True, axis = 1)
dfhealth.drop(columns= ['Location'], inplace=True, axis = 1)
dfhealth.count()  # Look at remaining columns to ensure Facility Name,
# Zipcode, Occupancy Type, Longitude, & Latitude present
#-------------------------------------------------------------------------------
-
```

```
# Changed selected columns to the same column anme for JOINs in housing, internet,
school,
# & health files
dfhousing.count() # Look at housing file count to select columns to change name
# Change column name 'Property Management Company' into 'Site Name' &
# 'Unit Type' into 'Site Type'
dfhousing.rename(columns={'PropertyManagementCompany':'SiteName',
                          'UnitType':'SiteType'}, inplace=True)
dfhousing.count() # 'Site Name' & 'Site Type' updated


dfinternet.count() # Look at internet file count to select columns to change name
# Change column name 'Organization Name' into 'Site Name' &
# 'Internet Type' into 'Site Type'
dfinternet.rename(columns={'OrganizationName':'SiteName',
                          'InternetType':'SiteType'}, inplace=True)
dfinternet.count() # 'Site Name' & 'Site Type' updated


dfschools.count() # Look at internet file count to select columns to change name
# Change column name 'School' into 'Site Name' & 'School Type' into 'Site Type'
dfschools.rename(columns={'School':'SiteName',
                          'SchoolType':'SiteType'}, inplace=True)
dfschools.count() # 'Site Name' & 'Site Type' updated


dfhealth.count() # Look at internet file count to select columns to change name
# Change column name 'Facility Name' into 'Site Name' & 'Occupancy Type' into 'Site
Type'
dfhealth.rename(columns={'FacilityName':'SiteName',
                          'OccupancyType':'SiteType'}, inplace=True)
dfhealth.count() # 'Site Name' & 'Site Type' updated
#-------------------------------------------------------------------------------

# Create new dataframes from selected columns of initial dataframes for preparing
# to merge for eventual file 1 output
dfhousing2 = dfhousing.loc[:,
['Zipcode','SiteName','SiteType','Longitude','Latitude']]
dfhousing2.count()

dfinternet2 = dfinternet.loc[:,
['Zipcode','SiteName','SiteType','Longitude','Latitude']]
dfinternet2.count()

dfschools2 = dfschools.loc[:,
['Zipcode','SiteName','SiteType','Longitude','Latitude']]
dfschools2.count()

dfhealth2 = dfhealth.loc[:,
['Zipcode','SiteName','SiteType','Longitude','Latitude']]
dfhealth2.count()

#-------------------------------------------------------------------------------

# Calculating data by groupby commands for file 2 output
# Averaging household incomes & summing up affortable housing units across zipcodes
G1 = dfhousing.groupby('Zipcode')['1PersonHousehold', '2PersonHousehold',
'3PersonHousehold', '4PersonHousehold',
'5PersonHousehold'].mean().sort_values('Zipcode', ascending = False)
```

```
G2 =
dfhousing[['Zipcode','TotalAffordableUnits']].groupby(['Zipcode']).sum().sort_value
s('Zipcode', ascending = False)
G1b = G1.astype(int) # Report calculated results as integers for cleaner results
G2b = G2.astype(int) # Report calculated results as integers for cleaner results

# Indexed calculated results for future mergers
G1b['Zipcode'] = G1b.index
G2b['Zipcode'] = G2b.index

# # Averaging racial percentages, summing total populations, & total housing units
across zipcodes
G3 = dfhoods.groupby('Zipcode')['NonHispanicWhitePercentage',
'AfricanAmericanPercentage', 'HispanicPercentage', 'AsianPercentage',
'OtherPercentage'].mean().sort_values('Zipcode', ascending = False)
G4 =
dfhoods[['Zipcode','TotalPopulation']].groupby(['Zipcode']).sum().sort_values('Zipc
ode', ascending = False)
G5 =
dfhoods[['Zipcode','TotalHousingUnits']].groupby(['Zipcode']).sum().sort_values('Zi
pcode', ascending = False)
G3a = G3 * 100 # Calculating results to percent amount results
G3b = G3a.round(1)  # Report calculated percentages for cleaner results

# Indexed calculated results for future mergers
G3['Zipcode'] = G3.index
G4['Zipcode'] = G4.index
G5['Zipcode'] = G5.index
#-------------------------------------------------------------------------------

# Merge ALL column-edited files to each other for file 1 output
dfmerge1 = dfhousing2.merge(dfinternet2, how='outer',
on=['Zipcode','SiteName','SiteType','Longitude','Latitude'])
dfmerge1.count() # dfhousing2 & dfinternet2 merged

dfmerge2 = dfschools2.merge(dfhealth2, how='outer',
on=['Zipcode','SiteName','SiteType','Longitude','Latitude'])
dfmerge2.count() # dfschools2 & dfhealth2 merged

dfmergebldgs = dfmerge1.merge(dfmerge2, how='outer',
on=['Zipcode','SiteName','SiteType','Longitude','Latitude'])
dfmergebldgs.count() # dfsmerge1 & dfmerge2 merged
dfmergebldgs.head(10) # Double check if inputed correctly. They did!!!

# Merge ALL column-edited files to each other for file 2 output
gmerge1 = G1b.merge(G2b, how= "inner", on='Zipcode') # G1 & G2 inputs merged
gmerge2 = gmerge1.merge(G3b, how= "inner", on='Zipcode') # gmerge1 & G3 inputs
merged

gmerge3 = gmerge2.merge(G4, how= "inner", on='Zipcode') # gmerge2 & G4 inputs
merged
gmergedemo = gmerge3.merge(G5, how= "inner", on='Zipcode') # gmerge3 & G5 merged
gmergedemo.count() # All columns merged
gmergedemo.head(10) # Double check if inputed correctly. They did!!!

# Create a new column of affordable housing rate
gmergedemo['AffordableHousingRate'] = ((gmergedemo['TotalAffordableUnits'] /
gmergedemo['TotalHousingUnits']) * 100).round(1)
# Report calculated percentages for cleaner results
```

```
gmergedemo.head(10) # Double check if results rounded correctly. They did!!!

#-------------------------------------------------------------------------------

# Double check if there are duplicates in dfmergebldgs
dfmergebldgs[dfmergebldgs.duplicated(['SiteName', 'Longitude', 'Latitude'],
keep='first')]
# 10 rows were found to be duplicated more than once in cols Site Name, Longitude,
& Latitude

# Remove these duplicates in dfmergebldgs
dfmergebldgs = dfmergebldgs.drop_duplicates(subset=['SiteName', 'Longitude',
'Latitude'], keep='first')
dfmergebldgs.count() # Check if duplicates were dropped. They were!!!

# Double check if there are duplicates in gmergedemo
gmergedemo[gmergedemo.duplicated(['Zipcode'], keep='first')]
# No rows were found to be duplicated more than once
#-------------------------------------------------------------------------------

# Create Austin_Public_Sites csv & Austin_Affordable_Housing_Demographics output
files
Austin_Public_Sites = dfmergebldgs
Austin_Public_Sites.to_csv(outpath + fileout1, index=False) # No number index with
columns included
Austin_Affordable_Housing_Demographics = gmergedemo
Austin_Affordable_Housing_Demographics.to_csv(outpath+fileout2, index=False)
```