bazalewski_capstone_cleaning

April 28, 2022

[1]:
```
import pandas as pd
pd.set_option('display.max_rows', 1000)
```

1 Sales Data

[2]:
```
#import sales csv
sales = pd.read_csv('new_business.csv')

#make names uppercase
sales['cleaned_name']=sales['Client Name (first L)'].str.upper().str.strip().
 ↪str.replace(',','').str.replace(' ','').str.replace('.','')
```

```
C:\Users\julie\anaconda3\lib\site-packages\ipykernel_launcher.py:5:
FutureWarning: The default value of regex will change from True to False in a
future version. In addition, single character regular expressions will*not* be
treated as literal strings when regex=True.
  """
```

[3]:
```
#import contacts csv
contacts = pd.read_csv('contacts_confidential.csv')

#remove those without zip codes
contacts_w_zip = contacts[contacts['Merged Zip'] != ' '].reset_index()

contacts_w_zip['cleaned_name'] = contacts_w_zip['Titleized First L'].str.
 ↪upper().str.strip().str.replace(',','').str.replace(' ','').str.replace('.
 ↪','')
```

```
C:\Users\julie\anaconda3\lib\site-packages\ipykernel_launcher.py:7:
FutureWarning: The default value of regex will change from True to False in a
future version. In addition, single character regular expressions will*not* be
treated as literal strings when regex=True.
  import sys
```

[4]:
```
#need to add another column for the cases when there is a couple name in the␣
 ↪sales data ("AND"). Add both directions
```

```
contacts_w_zip['combined name'] = contacts_w_zip['Last Name'].str.strip().
 ↪str[0] + contacts_w_zip['First Name1'].str.strip() + 'AND' +␣
 ↪contacts_w_zip['First Name2'].str.strip()
contacts_w_zip['combined name 2'] = contacts_w_zip['Last Name'].str.strip().
 ↪str[0] + contacts_w_zip['First Name2'].str.strip() +  'AND' +␣
 ↪contacts_w_zip['First Name1'].str.strip()
```

[5]:
```
#determine if there are still any duplicate entries
df = contacts_w_zip.groupby(['Titleized First L','Last Name','Merged Zip']).
 ↪agg([('total', 'count')]).reset_index()
filter_df = df[['Titleized First L', 'Last Name', 'First Name1', 'Merged Zip']]
filter_df[filter_df[('First Name1','total')] > 1].reset_index()
```

[5]:
```
Empty DataFrame
Columns: [(index, ), (Titleized First L, ), (Last Name, ), (First Name1, total),
(Merged Zip, )]
Index: []
```

[6]:
```
#determine if there are still any duplicate entries
df = contacts_w_zip.groupby(['Titleized First L']).agg([('total', 'count')]).
 ↪reset_index()
filter_df = df[['Titleized First L', 'Last Name']]
#filter_df
filter_df[filter_df[('Last Name','total')]> 1].reset_index()
```

[6]:
```
Empty DataFrame
Columns: [(index, ), (Titleized First L, ), (Last Name, total)]
Index: []
```

1.1 Nicknames

[7]:
```
#import nicknames csv (https://github.com/carltonnorthern/
 ↪nickname-and-diminutive-names-lookup)
nicknames = pd.read_csv('names.csv')
nicknames = nicknames.apply(lambda x: x.astype(str).str.upper())
```

[8]:
```
nickname_df = contacts_w_zip.merge(nicknames, left_on='First Name1',␣
 ↪right_on='name').
 ↪drop(['name','nickname7','nickname8','nickname9','nickname10','nickname11','nickname13','ni
```

[9]:
```
nickname_df['nickname1'] = nickname_df['Last Name'].str[0] +␣
 ↪nickname_df['nickname1']
nickname_df['nickname2'] = nickname_df['Last Name'].str[0] +␣
 ↪nickname_df['nickname2']
nickname_df['nickname3'] = nickname_df['Last Name'].str[0] +␣
 ↪nickname_df['nickname3']
```

```
nickname_df['nickname4'] = nickname_df['Last Name'].str[0] +␣
↪nickname_df['nickname4']
```

1.2 Join Sales and Contact Data Frames

[10]:
```
df_join = sales.merge(contacts_w_zip, on='cleaned_name')
```

[11]:
```
df_combined_names1_join = sales.merge(contacts_w_zip[contacts_w_zip['combined␣
↪name'].str.contains('AND')], left_on='cleaned_name', right_on='combined␣
↪name')
df_combined_names1_join = df_combined_names1_join.
↪drop(['cleaned_name_y'],axis=1)
```

[12]:
```
df_combined_names2_join = sales.merge(contacts_w_zip[contacts_w_zip['combined␣
↪name'].str.contains('AND')], left_on='cleaned_name', right_on='combined name␣
↪2')
df_combined_names2_join = df_combined_names2_join.
↪drop(['cleaned_name_y'],axis=1)
```

[13]:
```
joined_df = pd.
↪concat([df_join,df_combined_names1_join,df_combined_names2_join]).
↪reset_index()
joined_df = joined_df.drop(['cleaned_name_x'],axis=1)
```

[14]:
```
#check for duplicates
joined_df = joined_df.drop_duplicates()
```

[15]:
```
df_nicknames1_joined = sales.merge(nickname_df, left_on='cleaned_name',␣
↪right_on='nickname1')
df_nicknames2_joined = sales.merge(nickname_df, left_on='cleaned_name',␣
↪right_on='nickname2')
df_nicknames3_joined = sales.merge(nickname_df, left_on='cleaned_name',␣
↪right_on='nickname3')
df_nicknames4_joined = sales.merge(nickname_df, left_on='cleaned_name',␣
↪right_on='nickname4')
```

[16]:
```
joined_df = joined_df.drop(['level_0'],axis=1)
joined_df = pd.concat([joined_df,df_nicknames1_joined]).reset_index()
joined_df = joined_df.drop(['level_0'],axis=1)
joined_df = pd.concat([joined_df,df_nicknames2_joined]).reset_index()
joined_df = joined_df.drop(['level_0'],axis=1)
joined_df = pd.concat([joined_df,df_nicknames3_joined]).reset_index()
joined_df = joined_df.drop(['level_0'],axis=1)
joined_df = pd.concat([joined_df,df_nicknames4_joined]).reset_index()
```

[17]:
```
joined_df_final = joined_df[['Initial Contact Month', 'Initial Contact Year',␣
↪'Referral Source', 'Practice Area', 'Fee', 'Cleaned Cities', 'State',␣
↪'Merged Zip']]
```

[18]:
```
joined_df_final['Fee'] = joined_df_final['Fee'].str.strip().str.replace(',','').
↪str.replace('$','').str.replace('-','').str.replace(' ','NaN')
joined_df_final['Fee'] = pd.to_numeric(joined_df_final['Fee'], errors='coerce')
```

```
C:\Users\julie\anaconda3\lib\site-packages\ipykernel_launcher.py:1:
FutureWarning: The default value of regex will change from True to False in a
future version. In addition, single character regular expressions will*not* be
treated as literal strings when regex=True.
  """Entry point for launching an IPython kernel.
C:\Users\julie\anaconda3\lib\site-packages\ipykernel_launcher.py:1:
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
  """Entry point for launching an IPython kernel.
C:\Users\julie\anaconda3\lib\site-packages\ipykernel_launcher.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
```

[19]:
```
joined_df_final.columns = ['Contact Month', 'Contact Year', 'Referral',␣
↪'Practice Area', 'Fee', 'City', 'State', 'Zip']
```

[20]:
```
joined_df_final.to_csv('cleaned_sales.csv')
```

2 Calls Data

[21]:
```
#import calls csv
calls = pd.read_csv('call_list.csv')
```

[22]:
```
joined_calls_df = joined_df.merge(calls, left_on='Formatted Phone',␣
↪right_on='Phone Number', how="right")
```

[23]:
```
joined_calls_df_final = joined_calls_df[['Fee','Merged␣
↪Zip','City','State_y','Source','Duration (seconds)',
```

```
                                        'Start Time', 'Keywords', 'Campaign',␣
↪'Active Page', 'Ad Group',
                                        'Referrer', 'Device Type', 'Browser']]
joined_calls_df_final.columns         = ['Fee', 'Zip', 'City', 'State',␣
↪'Source', 'Duration',
                                        'Start Time', 'Keywords','Campaign',␣
↪'Page', 'Ad Group',
                                        'Referrer', 'Device Type', 'Browser']
```

[24]:
```
joined_calls_df_final.to_csv('cleaned_calls.csv')
```

3 Census Data

[ ]:
```
census_DP02 = pd.read_csv('ACSDP5Y2019.
↪DP02_data_with_overlays_2021-11-10T063909.csv', low_memory=False)
census_DP03 = pd.read_csv('ACSDP5Y2019.
↪DP03_data_with_overlays_2021-11-04T190527.csv', low_memory=False)
census_DP04 = pd.read_csv('ACSDP5Y2019.
↪DP04_data_with_overlays_2021-11-08T005314.csv', low_memory=False)
census_DP05 = pd.read_csv('ACSDP5Y2019.
↪DP05_data_with_overlays_2021-11-04T120142.csv', low_memory=False)
```

[ ]:
```
census_DP02 = census_DP02[['NAME', 'DP02_0001E', 'DP02_0002PE', 'DP02_0003PE',
       'DP02_0006PE', 'DP02_0010PE', 'DP02_0016E', 'DP02_0017E', 'DP02_0026PE',
       'DP02_0027PE', 'DP02_0030PE', 'DP02_0032PE', 'DP02_0033PE',
       'DP02_0036PE', 'DP02_0062PE', 'DP02_0064PE', 'DP02_0065PE',
       'DP02_0066PE', 'DP02_0072PE']]
census_DP03 = census_DP03[['NAME', 'DP03_0001E', 'DP03_0002PE', 'DP03_0009PE',
       'DP03_0047PE', 'DP03_0048PE', 'DP03_0049PE', 'DP03_0062E', 'DP03_0063E',
       'DP03_0088E']]
census_DP04 = census_DP04[['NAME', 'DP04_0041PE', 'DP04_0042PE', 'DP04_0043PE',
       'DP04_0089E', 'DP04_0101E', 'DP04_0110E', 'DP04_0111PE', 'DP04_0112PE',
       'DP04_0113PE', 'DP04_0114PE', 'DP04_0115PE', 'DP04_0134E', 'DP04_0136E',
       'DP04_0137PE', 'DP04_0138PE', 'DP04_0139PE', 'DP04_0140PE',
       'DP04_0141PE', 'DP04_0142PE']]
census_DP05 = census_DP05[['NAME', 'DP05_0001E', 'DP05_0002PE', 'DP05_0003PE',
       'DP05_0018E', 'DP05_0019PE', 'DP05_0023PE', 'DP05_0024PE',
       'DP05_0037PE', 'DP05_0038PE', 'DP05_0044PE', 'DP05_0071PE']]
```

[ ]:
```
census_DP02.columns = ['ZCTA', 'Total Households', 'Percent Married Couple␣
↪Family',
                       'Percent Married Couple Family with Children', 'Percent␣
↪Male Householder',
                      'Percent Female Householder', 'Average Household Size',␣
↪'Average Family Size',
```

```
                      'Percent Males Never Married', 'Percent Males Married',␣
 ↪'Percent Males Divorced',
                       'Percent Females Never Married','Percent Females␣
 ↪Married','Percent Females Divorced',
                      'Percent High School Grad', 'Percent Assoc Deg', 'Percent␣
 ↪Bachelors Deg',
                       'Percent Graduate Deg', 'Percent Disabled']
census_DP03.columns = ['ZCTA', 'Total Pop 16 and Up', 'Percent in Labor Force',␣
 ↪'Unemployment Rate',
                       'Percent Private Sector', 'Percent Govt Workers',␣
 ↪'Percent Self Employed',
                       'Median Income', 'Mean Income', 'Per Capita Income']
census_DP04.columns = ['ZCTA', 'Percent 2 Bedroom Homes', 'Percent 3 Bedroom␣
 ↪Homes', 'Percent 4 Bedroom Homes',
                       'Median House Value', 'Median Mortgage', 'Tot Housing␣
 ↪Units with Mortgage',
                       'Mortgage Less than 20 Percent of Income', 'Mortgage␣
 ↪Between 20 and 25 Percent of Income',
                      'Mortgage Between 25 and 30 Percent of Income', 'Mortgage␣
 ↪Between 30 and 35 Percent of Income',
                      'Mortgage More than 35 Percent of Income', 'Total Units␣
 ↪Paying Rent',
                       'Rent Less than 15 Percent of Income', 'Rent Between 15␣
 ↪and 20 Percent of Income',
                       'Rent Between 20 and 25 Percent of Income', 'Rent␣
 ↪Between 20 and 25 Percent of Income',
                       'Rent Between 25 and 30 Percent of Income', 'Rent␣
 ↪Between 30 and 35 Percent of Income',
                       'Rent More than 35 Percent of Income']
census_DP05.columns = ['ZCTA', 'Total Pop', 'Percent Male', 'Percent Female',␣
 ↪'Median Age', 'Percent Under 18',
                       'Percent 62 and Over', 'Percent 65 and Over', 'Percent␣
 ↪White', 'Percent Black', 'Percent Asian',
                      'Percent Hispanic']
```

[ ]:
```
#join tables
census_df = census_DP02.merge(census_DP03, on='ZCTA').merge(census_DP04,␣
 ↪on='ZCTA').merge(census_DP05, on='ZCTA')
```

[ ]:
```
#remove NA (some tables do not include Puerto Rico data)
census_df = census_df.dropna().drop(0, axis=0).reset_index(drop=True)
```

[ ]:
```
#remove 'ZCTA5' from each ZCTA
census_df['ZCTA'] = census_df['ZCTA'].str.replace('ZCTA5','')
```

[ ]:
```
census_df
```

[ ]:
```
census_df.to_csv('cleaned_census.csv')
```

[ ]:

[ ]:

[ ]:

[ ]: