rotina_1977

June 30, 2015

In [1]:
```
from IPython.display import display #from IPython.core.display import HTML

import pandas as pd
pd.set_option('display.mpl_style', 'default') #Make the graphs a bit prettier

#Variable to avoid log prints when generating pdf file
impressao = False #True = to not print logs | False = to print logs
```

0.1 Funções gerais

In [2]:
```
def consulta_refext(row, name_file, name_col_ref, name_col_filt, name_col_search):
    """
    Traz valor de referência externa (em arquivo csv) baseado em valor de referência do arquivo
    O primeiro argumento passado é a "linha".
    O segundo argumento é o nome do arquivo csv que será consultado (indicar o nome com a exten
    O terceiro argumento é o nome da coluna no dataframe (.csv) consultado que servirá de refên
    O quarto argumento é o nome da coluna de filtro do dataframe atual
    O quinto argumento é o nome da coluna no dataframe (.csv) consultado que contém o valor a s
        Uso:
        od1977['coluna a receber o valor'] = od1977.apply(lambda row: consulta_refext(row, 'fil
    """
    if row [name_col_filt]==0:
        return row[name_col_filt]
    data_frame = pd.read_csv(name_file,sep=';')
    return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])
```

In [3]:
```
def verifica_DUMMY(data_frame, nome_variavel):
    """
    Verifica se uma variável, dummy, contém algum valor diferente de 0 ou de 1.
        Uso:
        verifica_DUMMY(nome_do_dataframe, 'coluna a ser verificada')
    """
    contador_de_erros = 0
    for index, value in data_frame.iterrows():
        if int(value[nome_variavel]) != 1 and int(value[nome_variavel]) != 0:
            if not impressao:
                print("Erro encontrado no registro " + str(index+1) + ".")
                print("    Valor encontrado: " + str(value[nome_variavel]))
            contador_de_erros += 1
    print("Total de erros encontrados: " + str(contador_de_erros))
```

In [4]:
```
def verifica_RANGE(df, variavel, valor_menor, valor_maior):
    """
```

```
            Verifica se uma variável, do tipo número inteiro, contém algum valor menor que "valor_menor
                Uso:
                verifica_RANGE(nome_do_dataframe, 'coluna a ser verificada', 'valor_menor', 'valor_maio
            """
            df_filtrado = df[(df[variavel]<valor_menor) | (df[variavel]>valor_maior)]
            #Printing a summary of the values that not fit in the Range
            result = df_filtrado[variavel].value_counts()
            print(result)
            #If 'impressao = False', the output contains the values of dataframe that do not fit in the
            if not impressao:
                df_filtrado
```

In [5]:
```
def gera_ID_DOM(row):
    """
    Gera o ID_DOM baseado no 'ANO', na 'ZONA_DOM' e no 'NO_DOM'
    O argumento passado é a "linha".
        Uso:
        od1977['ID_DOM'] = od1977.apply(gera_ID_DOM, axis=1)
    """
    ano = int(row['ANO'])
    zona = int(row['ZONA_DOM'])
    no_dom = int(row['NO_DOM'])
    return int(str(ano)+str('%03d'%(zona)) + str('%04d'%(no_dom)))
```

In [6]:
```
def gera_ID_FAM(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
        Uso:
        od1977['ID_FAM'] = od1977.apply(gera_ID_FAM, axis=1)
    """
    id_dom = int(row['ID_DOM'])
    no_fam = int(row['NO_FAM'])
    return int(str(id_dom) + str('%02d'%(no_fam)))
```

In [7]:
```
def gera_ID_PESS(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
        Uso:
        od1977['ID_PESS'] = od1977.apply(gera_ID_PESS, axis=1)
    """
    id_fam = int(row['ID_FAM'])
    no_pess = int(row['NO_PESS'])
    return int(str(id_fam) + str('%02d'%(no_pess)))
```

In [8]:
```
def gera_ID_VIAG(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
        Uso:
        od1977['ID_VIAG'] = od1977.apply(gera_ID_VIAG, axis=1)
    """
    id_pess = int(row['ID_PESS'])
    no_viag = int(row['NO_VIAG'])
    return int(str(id_pess) + str('%02d'%(no_viag)))
```

```
In [9]: def calcula_DIST_VIAG(row):
            """
            Calcula a distância euclidiana dadas as coordenadas (x,y) de origem e coordenadas (x,y) de
            O argumento passado é a "linha".
                Uso:
                od1977['DIST_VIAG'] = od1977.apply(calcula_DIST_VIAG, axis=1)
            """
            co_orig_x = float(row['CO_ORIG_X'])
            co_orig_y = float(row['CO_ORIG_Y'])
            co_dest_x = float(row['CO_DEST_X'])
            co_dest_y = float(row['CO_DEST_Y'])
            return math.sqrt(math.pow((co_orig_x - co_dest_x), 2) + math.pow((co_orig_y - co_dest_y), 2)
```

```
In [10]: #Reading csv file and store its contend in an intern dataframe
         od1977 = pd.read_csv('OD_1977.csv', sep=';', decimal=',')

In [11]: #Renaming the column UCOD to UCOD_DOM
         od1977.rename(columns={'UCOD':'UCOD_DOM'}, inplace=True)

In [12]: #Creating the column UCOD_ESC (it will go to the end of dataframe)
         od1977['UCOD_ESC']=None

In [13]: #Creating the column UCOD_TRAB1 (it will go to the end of dataframe)
         od1977['UCOD_TRAB1']=None

In [14]: #Creating the column UCOD_TRAB2 (it will go to the end of dataframe)
         od1977['UCOD_TRAB2']=None

In [15]: #Creating the column UCOD_ORIG (it will go to the end of dataframe)
         od1977['UCOD_ORIG']=None

In [16]: #Creating the column UCOD_DEST (it will go to the end of dataframe)
         od1977['UCOD_DEST']=None

In [17]: od1977 = od1977[:15000]

In [18]: #Reordering the columns, precisely, these that were just created (at the end of dataframe) nea
         od1977 = od1977[['ANO',
          'CD_ENTRE',
          'DIA_SEM',
          'UCOD_DOM',
          'ZONA_DOM',
          'SUBZONA_DOM',
          'MUN_DOM',
          'CO_DOM_X',
          'CO_DOM_Y',
          'ID_DOM',
          'F_DOM',
          'FE_DOM',
          'NO_DOM',
          'TIPO_DOM',
          'TOT_FAM',
          'ID_FAM',
          'F_FAM',
```

```
'FE_FAM',
'NO_FAM',
'COND_MORA',
'QT_AUTO',
'QT_BICI',
'QT_MOTO',
'CD_RENFAM',
'REN_FAM',
'ID_PESS',
'F_PESS',
'FE_PESS',
'NO_PESS',
'SIT_FAM',
'IDADE',
'SEXO',
'ESTUDA',
'GRAU_INSTR',
'OCUP',
'SETOR_ATIV',
'CD_RENIND',
'REN_IND',
'UCOD_ESC',
'ZONA_ESC',
'SUBZONA_ESC',
'MUN_ESC',
'CO_ESC_X',
'CO_ESC_Y',
'UCOD_TRAB1',
'ZONA_TRAB1',
'SUBZONA_TRAB1',
'MUN_TRAB1',
'CO_TRAB1_X',
'CO_TRAB1_Y',
'UCOD_TRAB2',
'ZONA_TRAB2',
'SUBZONA_TRAB2',
'MUN_TRAB2',
'CO_TRAB2_X',
'CO_TRAB2_Y',
'ID_VIAG',
'F_VIAG',
'FE_VIAG',
'NO_VIAG',
'TOT_VIAG',
'UCOD_ORIG',
'ZONA_ORIG',
'SUBZONA_ORIG',
'MUN_ORIG',
'CO_ORIG_X',
'CO_ORIG_Y',
'UCOD_DEST',
'ZONA_DEST',
'SUBZONA_DEST',
'MUN_DEST',
```

```
          'CO_DEST_X',
          'CO_DEST_Y',
          'DIST_VIAG',
          'MOTIVO_ORIG',
          'MOTIVO_DEST',
          'MODO1',
          'MODO2',
          'MODO3',
          'MODO4',
          'MODO_PRIN',
          'TIPO_VIAG',
          'H_SAIDA',
          'MIN_SAIDA',
          'ANDA_ORIG',
          'H_CHEG',
          'MIN_CHEG',
          'ANDA_DEST',
          'DURACAO',
          'TIPO_EST_AUTO',
          'VALOR_EST_AUTO']]
```

In [19]:
```
#Storing the variables list in the "cols" variable
cols = od1977.columns.tolist()
if not impressao:
    #printing "cols" variable to check if the reorder operation was effective
    display(cols)
```

```
['ANO',
 'CD_ENTRE',
 'DIA_SEM',
 'UCOD_DOM',
 'ZONA_DOM',
 'SUBZONA_DOM',
 'MUN_DOM',
 'CO_DOM_X',
 'CO_DOM_Y',
 'ID_DOM',
 'F_DOM',
 'FE_DOM',
 'NO_DOM',
 'TIPO_DOM',
 'TOT_FAM',
 'ID_FAM',
 'F_FAM',
 'FE_FAM',
 'NO_FAM',
 'COND_MORA',
 'QT_AUTO',
 'QT_BICI',
 'QT_MOTO',
 'CD_RENFAM',
 'REN_FAM',
 'ID_PESS',
 'F_PESS',
 'FE_PESS',
```

```
'NO_PESS',
'SIT_FAM',
'IDADE',
'SEXO',
'ESTUDA',
'GRAU_INSTR',
'OCUP',
'SETOR_ATIV',
'CD_RENIND',
'REN_IND',
'UCOD_ESC',
'ZONA_ESC',
'SUBZONA_ESC',
'MUN_ESC',
'CO_ESC_X',
'CO_ESC_Y',
'UCOD_TRAB1',
'ZONA_TRAB1',
'SUBZONA_TRAB1',
'MUN_TRAB1',
'CO_TRAB1_X',
'CO_TRAB1_Y',
'UCOD_TRAB2',
'ZONA_TRAB2',
'SUBZONA_TRAB2',
'MUN_TRAB2',
'CO_TRAB2_X',
'CO_TRAB2_Y',
'ID_VIAG',
'F_VIAG',
'FE_VIAG',
'NO_VIAG',
'TOT_VIAG',
'UCOD_ORIG',
'ZONA_ORIG',
'SUBZONA_ORIG',
'MUN_ORIG',
'CO_ORIG_X',
'CO_ORIG_Y',
'UCOD_DEST',
'ZONA_DEST',
'SUBZONA_DEST',
'MUN_DEST',
'CO_DEST_X',
'CO_DEST_Y',
'DIST_VIAG',
'MOTIVO_ORIG',
'MOTIVO_DEST',
'MODO1',
'MODO2',
'MODO3',
'MODO4',
'MODO_PRIN',
'TIPO_VIAG',
```

```
 'H_SAIDA',
 'MIN_SAIDA',
 'ANDA_ORIG',
 'H_CHEG',
 'MIN_CHEG',
 'ANDA_DEST',
 'DURACAO',
 'TIPO_EST_AUTO',
 'VALOR_EST_AUTO']
```

```
In [20]: if not impressao:
             #Describing data (whole dataframe)- count, mean, std, min and max
             display(od1977.describe())
```

	ANO	CD_ENTRE	DIA_SEM	UCOD_DOM	ZONA_DOM	SUBZONA_DOM	MUN_DOM	\
count	0	0	0	0	15000.000000	15000.000000	15000	
mean	NaN	NaN	NaN	NaN	9.747600	311.355267	1	
std	NaN	NaN	NaN	NaN	5.207786	184.137085	0	
min	NaN	NaN	NaN	NaN	1.000000	1.000000	1	
25%	NaN	NaN	NaN	NaN	5.000000	276.000000	1	
50%	NaN	NaN	NaN	NaN	10.000000	290.000000	1	
75%	NaN	NaN	NaN	NaN	14.000000	519.000000	1	
max	NaN	NaN	NaN	NaN	18.000000	633.000000	1	

	CO_DOM_X	CO_DOM_Y	ID_DOM	...	TIPO_VIAG	\
count	0	0	1.500000e+04	...	15000.000000	
mean	NaN	NaN	1.002741e+12	...	2.246933	
std	NaN	NaN	5.203091e+11	...	0.827850	
min	NaN	NaN	1.200010e+11	...	1.000000	
25%	NaN	NaN	5.200071e+11	...	2.000000	
50%	NaN	NaN	1.030021e+12	...	2.000000	
75%	NaN	NaN	1.440003e+12	...	3.000000	
max	NaN	NaN	1.830004e+12	...	3.000000	

	H_SAIDA	MIN_SAIDA	ANDA_ORIG	H_CHEG	MIN_CHEG	\
count	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	
mean	11.100400	15.790067	1.884267	11.275600	19.985267	
std	6.718382	17.492787	3.514100	6.826767	18.291798	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	7.000000	0.000000	0.000000	7.000000	0.000000	
50%	12.000000	10.000000	0.000000	12.000000	15.000000	
75%	17.000000	30.000000	2.000000	17.000000	35.000000	
max	23.000000	59.000000	50.000000	24.000000	59.000000	

	ANDA_DEST	DURACAO	TIPO_EST_AUTO	VALOR_EST_AUTO
count	15000.000000	15000.000000	15000.000000	15000.000000
mean	1.876667	23.201667	0.711000	4.012733
std	3.702031	25.271876	1.759494	49.587124
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	5.000000	0.000000	0.000000
50%	0.000000	15.000000	0.000000	0.000000
75%	2.000000	30.000000	0.000000	0.000000
max	59.000000	240.000000	7.000000	3500.000000

```
[8 rows x 86 columns]
```

0.2 Passo 1: UCOD_DOM

Na coluna "UCOD_DOM", linha i, ler o valor da linha i da coluna "ZONA_DOM", daí, buscar o mesmo valor na coluna "Zona 1977" do arquivo UCOD-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "UCOD_DOM"

[Teste: no banco completo, checar se o min == 1 e o max == 67]

```
In [21]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_DOM" code
         od1977['UCOD_DOM'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 1977
```

```
In [22]: if not impressao:
             #Describing data ("UCOD_DOM" column) - count, mean, std, min and max
             display(od1977['UCOD_DOM'].describe())
```

```
count    15000.000000
mean         3.896667
std          2.220852
min          1.000000
25%          2.000000
50%          5.000000
75%          6.000000
max          7.000000
Name: UCOD_DOM, dtype: float64
```

```
In [23]: if not impressao:
             #Count for check "UCOD_DOM"
             display(od1977['UCOD_DOM'].value_counts())
```

```
5    4094
2    3481
1    3136
7    2148
6    2141
dtype: int64
```

```
In [24]: #Verifying value interval for check - conditions: "UCOD_DOM < 1" and "UCOD_DOM > 67"
         verifica_RANGE(od1977, 'UCOD_DOM', 1, 67)
         #od1977_ex[(od1977['UCOD_DOM']<1) | (od1977['UCOD_DOM']>67)]
```

```
Series([], dtype: int64)
```

0.3 Passo 2: "ANO"

Preencher a coluna "ANO" com valor 1 em todas células ####Categorias: |valor|ano_correspondente| |——-|——| |1|1977| |2|1987| |3|1997| |4|2007|

```
In [25]: #Assigning value '1' to all cels of the "ANO" column
         od1977["ANO"]=1
```

```
In [26]: if not impressao:
             #Describing data ("ANO" column) - count, mean, std, min and max
             display(od1977['ANO'].describe())
```

```
count    15000
mean         1
std          0
min          1
25%          1
50%          1
75%          1
max          1
Name: ANO, dtype: float64
```

0.4 Passo 3: "CD_ENTRE"

Substituir valores da coluna "CD_ENTRE" Todas viagens são consideradas "completas", segundo informações do Metrô

- sem viagem: se TOT_VIAG == 0
- com viagem: se TOT_VIAG != 0

Valor	Descrição
0	Completa sem viagem
1	Completa com viagem

Categorias novas [Teste: Checar se existe algum número diferente de 0 ou 1. Se encontrar, retornar erro indicando em qual linha.]

```
In [27]: if not impressao:
             #Counting for check "CD_ENTRE"
             display(od1977['CD_ENTRE'].value_counts())
```

```
Series([], dtype: int64)
```

```
In [28]: #Defining 'CD_ENTRE' based on 'TOT_VIAG' values
         od1977[od1977['TOT_VIAG']==0]['CD_ENTRE']=0
         od1977[od1977['TOT_VIAG']!=0]['CD_ENTRE']=1
```

```
In [29]: if not impressao:
             #Counting "CD_ENTRE" in order to compare the values before and after the replacement
             display(od1977['CD_ENTRE'].value_counts())
```

```
Series([], dtype: int64)
```

```
In [30]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1977, 'CD_ENTRE')
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-30-e651327ac353> in <module>()
  1 #Verifying if there was left some value other than 0 or 1
----> 2 verifica_DUMMY(od1977, 'CD_ENTRE')


<ipython-input-3-bdf264bbc98a> in verifica_DUMMY(data_frame, nome_variavel)
  7     contador_de_erros = 0
  8     for index, value in data_frame.iterrows():
----> 9         if int(value[nome_variavel]) != 1 and int(value[nome_variavel]) != 0:
 10             if not impressao:
 11                 print("Erro encontrado no registro " + str(index+1) + ".")


ValueError: cannot convert float NaN to integer
```

0.5 Passo 4: "DIA_SEM"

Não existe essa informação no banco de dados de 1977, logo, este campo será preenchido com 0.

Valor	Descrição
0	Não disponível
2	Segunda-Feira
3	Terça-Feira
4	Quarta-Feira
5	Quinta-Feira
6	Sexta-Feira

Categorias:

```
In [31]: #Assigning value '0' to all cels of the "DIA_SEM" column
         od1977['DIA_SEM']=0

In [32]: if not impressao:
             #Counting "DIA_SEM" in order to check the values after the procedure
             display(od1977['DIA_SEM'].value_counts())

0    15000
dtype: int64
```

0.6 Passo 5: "ZONA_DOM"

Checar se existe algum erro

Categorias:

1 a 243

[Teste: Checar se existe algum número < 1 ou > 243. Se encontrar, retornar erro indicando em qual linha.]

```
In [33]: #Verifying value interval for check - conditions: "ZONA_DOM < 1" and "ZONA_DOM > 243"
         #od1977[(od1977['ZONA_DOM']<1) | (od1977['ZONA_DOM']>243)]
         verifica_RANGE(od1977, 'ZONA_DOM', 1, 243)
```

```
Series([], dtype: int64)
```

0.7 Passo 6: "SUBZONA_DOM"

Checar se existe algum erro

Categorias:

1 a 633

[Teste: Checar se existe algum número < 1 ou > 633. Se encontrar, retornar erro indicando em qual linha.]

```
In [34]: #Verifying value interval for check - conditions: "SUBZONA_DOM < 1" and "SUBZONA_DOM > 633"
         #od1977[(od1977['SUBZONA_DOM']<1) | (od1977['SUBZONA_DOM']>633)]
         verifica_RANGE(od1977, 'SUBZONA_DOM', 1, 633)
```

```
Series([], dtype: int64)
```

0.8 Passo 7: "MUN_DOM"

Checar se existe algum erro

Categorias

1 a 27

[Teste: Checar se existe algum número < 1 ou > 27. Se encontrar, retornar erro indicando em qual linha.]

```
In [35]: #Verifying value interval for check - conditions: "MUN_DOM < 1" and "MUN_DOM > 27"
         #od1977[(od1977['MUN_DOM']<1) | (od1977['MUN_DOM']>27)]
         verifica_RANGE(od1977, 'MUN_DOM', 1, 27)
```

```
Series([], dtype: int64)
```

0.9 Passo 8: "CO_DOM_X"

Na coluna "CO_DOM_X", linha i, ler o valor da linha i da coluna "SUBZONA_DOM", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"

```
In [38]: #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_DOM" code
         od1977['CO_DOM_X'] = od1977.apply(lambda row: consulta_refext(row, 'coord_subzonas_1977.csv',
```

```
---------------------------------------------------------------------------

OSError                                   Traceback (most recent call last)

<ipython-input-38-941bb3407119> in <module>()
      1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_DOM" code
----> 2 od1977['CO_DOM_X'] = od1977.apply(lambda row: consulta_refext(row, 'coord_subzonas_1977.csv',


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


<ipython-input-38-941bb3407119> in <lambda>(row)
      1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_DOM" code
----> 2 od1977['CO_DOM_X'] = od1977.apply(lambda row: consulta_refext(row, 'coord_subzonas_1977.csv',


<ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name
```

```
/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
244
245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
247
248     if (nrows is not None) and (chunksize is not None):


/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
560             self.options['has_index_names'] = kwds['has_index_names']
561
--> 562         self._make_engine(self.engine)
563
564     def _get_options_with_defaults(self, engine):


/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
697     def _make_engine(self, engine='c'):
698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
700         else:
701             if engine == 'python':


/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
1064         kwds['allow_leading_cols'] = self.index_col is not False
1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
1067
1068         # XXX


pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


OSError: ("File b'coord_subzonas_1977.csv' does not exist", 'occurred at index 0')
```

0.10 Passo 9: "CO_DOM_Y"

Na coluna "CO_DOM_Y", linha i, ler o valor da linha i da coluna "SUBZONA_DOM", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"

```
In [39]: #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_DOM" code
         od1977['CO_DOM_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', 'S
```

```
--------------------------------------------------------------------------
```

```
    OSError                                   Traceback (most recent call last)

    <ipython-input-39-402be06611e0> in <module>()
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_DOM" code
----> 2 od1977['CO_DOM_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv',


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-39-402be06611e0> in <lambda>(row)
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_DOM" code
----> 2 od1977['CO_DOM_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv',


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
```

```
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
    697     def _make_engine(self, engine='c'):
    698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
    700         else:
    701             if engine == 'python':


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1064         kwds['allow_leading_cols'] = self.index_col is not False
   1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
   1067
   1068         # XXX


    pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


    pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


    OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 0')
```

0.11 Passo 10: "ID_DOM"

construir o "ID_DOM"

[Na coluna "ID_DOM", linha i, ler o valor da linha i da coluna "ZONA_DOM", e concatenar esse valor (com 3 dígitos) com o número do domicílio , que é o valor da linha i da coluna "NO_DOM" (com 4 dígitos). Resultado será um ID_DOM, que pode se repetir nas linhas, de 7 dígitos. Isso deve ser concatenado com o "Ano". Resultado = 8 dígitos]

Outra possibilidade, concatenar com a UCOD ao invés da zona...

```
In [41]: # Generating "NO_DOM" as a subindex of each "ZONA_DOM"
         # For each 'ZONA_DOM' the 'NO_DOM' will be updated every time the 'F_DOM' is equal to 1.
         # Otherwise - if 'F_DOM' is equal to 0; then the 'NO_DOM' will be equal to the 'NO_DOM' from t

         def gera_NO_DOM(row):
             # Use this function with:
             #     dataframe.apply(gera_NO_DOM, axis=1)
             #
             # Return 1 if the 'NO_DOM' was applied and 0 if it was not.
             #
             # row.name is the index of the specific row.

             return_value = 1
             if row.name == 0 or od1977.loc[row.name, 'ZONA_DOM'] != od1977.loc[row.name - 1, 'ZONA_DOM
```

```
            # If first row of dataframe, then NO_DOM = 1. or
            # If first row of a ZONA_DOM, then NO_DOM = 1 also.
            # This considers that the dataframe is (also) ordered by ZONA_DOM.
            # It is a strong requirement.
            od1977.loc[row.name, 'NO_DOM'] = 1
        elif row['F_DOM'] == 1:
            od1977.loc[row.name, 'NO_DOM'] = od1977.loc[row.name - 1, 'NO_DOM'] + 1
        elif row['F_DOM'] == 0:
            od1977.loc[row.name, 'NO_DOM'] = od1977.loc[row.name - 1, 'NO_DOM']
        else:
            print("Something went wrong on the ID" + str(row.name))
            return 0
        return 1
```

```
In [42]: od1977.apply(gera_NO_DOM, axis=1).sum()
```

```
Out[42]: 15000
```

```
In [43]: #Generating "ID_DOM" from the concatenation of "ANO", "ZONA_DOM" and "NO_DOM" variables
         od1977['ID_DOM'] = od1977.apply(gera_ID_DOM, axis=1)
```

0.12 Passo 11: "F_DOM"

Checar se existe algum erro na coluna "F_DOM"

Valor	Descrição
0	Demais registros
1	Primeiro Registro do Domicílio

Categorias [Teste: Checar se existe algum número diferente de 0 ou 1. Se encontrar, retornar erro indicando em qual linha.]

```
In [44]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1977, 'F_DOM')
```

```
Total de erros encontrados: 0
```

0.13 "FE_DOM" e "NO_DOM"

Nada há que se fazer em relação aos dados das colunas "FE_DOM" e "NO_DOM"

0.14 Passo 12: "TIPO_DOM"

Checar se existe algum erro

Valor	Descrição
0	Não respondeu
1	Particular
2	Coletivo

Categorias anteriores / novas [Teste: Checar se existe algum número < 0 ou > 2. Se encontrar, retornar erro indicando em qual linha.]

```
In [45]: if not impressao:
             #Counting for check "TIPO_DOM"
             display(od1977['TIPO_DOM'].value_counts())
```

```
1    13590
2     1274
0      136
dtype: int64
```

```
In [46]: #Verifying value interval for check - conditions: "TIPO_DOM < 0" and "TIPO_DOM > 2"
         #od1977[(od1977['TIPO_DOM']<0) | (od1977['TIPO_DOM']>2)]
         verifica_RANGE(od1977, 'TIPO_DOM', 0, 2)
```

```
Series([], dtype: int64)
```

0.15 "TOT_FAM"

Nada há que se fazer em relação aos dados da coluna "TOT_FAM"

0.16 Passo 13: "ID_FAM"

Construir o "ID_FAM"

Na coluna "ID_FAM", linha i, ler o valor da linha i da coluna "ID_DOM", e concatenar esse valor (com 8 dígitos) com o número da família, que é o valor da linha i da coluna "NO_FAM" (com 2 dígitos).

Resultado será um ID_FAM, que pode se repetir nas linhas, de 10 dígitos.

```
In [47]: #Generating "ID_FAM" from the concatenation of "ID_DOM" and "NO_FAM" variables
         od1977['ID_FAM'] = od1977.apply(gera_ID_FAM, axis=1)
```

0.17 Passo 14: "F_FAM"

Checar se existe algum erro na coluna "F_FAM"

Valor	Descrição
0	Demais registros
1	Primeiro Registro da Família

Categorias [Teste: Checar se existe algum número diferente de 0 ou 1. Se encontrar, retornar erro indicando em qual linha.]

```
In [48]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1977, 'F_FAM')
```

```
Total de erros encontrados: 0
```

0.18 "FE_FAM" e "NO_FAM"

Nada há que se fazer em relação aos dados das colunas "FE_FAM" e "NO_FAM"

0.19 Passo 15: "COND_MORA"

Substituir valores da coluna "COND_MORA"

- Substituir todos valores **1** por **2**
- Substituir todos valores **3** por **1**
- Substituir todos valores **4** por **3**
- Substituir todos valores **5** por **3**
- Substituir todos valores **0** por **4**

Valor	Descrição
1	Própria paga
2	Própria em pagamento
3	Alugada
4	Cedida
5	Outro

Categorias anteriores

Valor	Descrição
1	Alugada
2	Própria
3	Outros
4	Não respondeu

Categorias novas [Teste: Checar se existe algum número < 1 ou > 4. Se encontrar, retornar erro indicando em qual linha.]

```
In [49]: if not impressao:
             #Counting for check "COND_MORA"
             display(od1977['COND_MORA'].value_counts())
```

```
3    9614
1    3738
2     836
4     660
0     145
5       7
dtype: int64
```

```
In [50]: #Replacing the values 1 for 2
         od1977.loc[od1977['COND_MORA']==1,'COND_MORA'] = 2
         #Replacing the values 3 for 1
         od1977.loc[od1977['COND_MORA']==3,'COND_MORA'] = 1
         #Replacing the values 4 for 3
         od1977.loc[od1977['COND_MORA']==4,'COND_MORA'] = 3
         #Replacing the values 5 for 3
         od1977.loc[od1977['COND_MORA']==5,'COND_MORA'] = 3
         #Replacing the values 0 for 4
         od1977.loc[od1977['COND_MORA']==0,'COND_MORA'] = 4
```

```
In [51]: if not impressao:
             #Counting "COND_MORA" in order to compare the values before and after the replacement
             display(od1977['COND_MORA'].value_counts())
```

```
1    9614
2    4574
3     667
4     145
dtype: int64
```

```
In [52]: #Verifying value interval for check - conditions: "COND_MORA < 1" and "COND_MORA > 4"
         #od1977[(od1977['COND_MORA']<1) | (od1977['COND_MORA']>4)]
         verifica_RANGE(od1977, 'COND_MORA', 1, 4)
```

```
Series([], dtype: int64)
```

0.20 "QT_AUTO"

Nada há que se fazer em relação aos dados da coluna "QT_AUTO"

0.21 "QT_BICI" e QT_MOTO"

Não existe essa informação no banco de dados de 1977, logo, estes campos serão preenchidos com 0.

```
In [53]: #Assigning value '0' to all cels of the "QT_BICI" column
         od1977['QT_BICI']=0
```

```
In [54]: if not impressao:
             #Counting "QT_BICI" in order to check the values after the procedure
             display(od1977['QT_BICI'].value_counts())
```

```
0    15000
dtype: int64
```

```
In [55]: #Assigning value '0' to all cels of the "QT_BICI" column
         od1977['QT_MOTO']=0
```

```
In [56]: if not impressao:
             #Counting "QT_MOTO" in order to check the values after the procedure
             display(od1977['QT_MOTO'].value_counts())
```

```
0    15000
dtype: int64
```

0.22 Passo 16: "CD_RENFAM"

Substituir valores da coluna "CD_RENFAM"

- Excluir único registro na categoria 3
- Substituir todos valores **1** por **2**
- Quando categoria original for 0 (respondeu) checar no campo REN_FAM se o valor é nulo. Se for nulo, manter na categoria 0 (Renda Familiar Declarada como Zero), senão, mover para a categoria 1 (Renda Familiar Declarada e Maior que Zero).

Valor	Descrição
0	Respondeu
1	Não Sabe
2	Não Respondeu
3	Não se Aplica

Categorias anteriores

Valor	Descrição
0	Renda Familiar Declarada como Zero
1	Renda Familiar Declarada e Maior que Zero
2	Renda Atribuída

Categorias novas [Teste: Checar se existe algum número < 0 ou > 2. Se encontrar, retornar erro indicando em qual linha.]

```
In [57]: if not impressao:
             #Counting for check "CD_RENFAM"
             display(od1977['CD_RENFAM'].value_counts())
```

```
0    14324
```

```
2      395
1      245
3       36
dtype: int64
```

```
In [58]: #Excluding the only row whose value is 3
         #######

In [59]: #Replacing the values 1 for 2
         od1977.loc[od1977['CD_RENFAM']==1,'CD_RENFAM'] = 2

In [60]: #Spliting the category 0 "Respondeu" into 0 "Renda Familiar Declarada como Zero" and 1 "Renda
         #####

In [61]: if not impressao:
             #Counting "CD_RENFAM" in order to compare the values before and after the replacement
             display(od1977['CD_RENFAM'].value_counts())
```

```
0    14324
2      640
3       36
dtype: int64
```

```
In [62]: #Verifying value interval for check - conditions: "CD_RENFAM < 0" and "CD_RENFAM > 2"
         #od1977[(od1977['CD_RENFAM']<0) | (od1977['CD_RENFAM']>2)]
         verifica_RANGE(od1977, 'CD_RENFAM', 0, 2)
```

```
3    36
dtype: int64
```

0.23 "REN_FAM"

Nada há que se fazer em relação aos dados da colunas "REN_FAM"

0.24 Passo 17: "ID_PESS"

Construir o "ID_PESS"

Na coluna "ID_PESS", linha i, ler o valor da linha i da coluna "ID_FAM", e concatenar esse valor (10 dígitos) com o número da pessoa, que é o valor da linha i da coluna "NO_PESS" (com 2 dígitos).

Resultado será um ID_PESS, que pode se repetir nas linhas, de 12 dígitos.

```
In [63]: #Generating "ID_PESS" from the concatenation of "ID_FAM" and "NO_PESS" variables
         od1977['ID_PESS'] = od1977.apply(gera_ID_PESS, axis=1)
```

0.25 Passo 18: "F_PESS"

Checar se existe algum erro na coluna "F_PESS"

Valor	Descrição
0	Demais registros
1	Primeiro Registro da Pessoa

Categorias [Teste: Checar se existe algum número diferente de 0 ou 1. Se encontrar, retornar erro indicando em qual linha.]

```
In [64]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1977, 'F_PESS')

Total de erros encontrados: 0
```

0.26 "FE_PESS" e "NO_PESS"

Nada há que se fazer em relação aos dados das colunas "FE_PESS" e "NO_PESS"

0.27 Passo 19: "SIT_FAM"

- Substituir todos valores **5** por **4**
- Substituir todos valores **6** por **5**
- Substituir todos valores **7** por **6**

Valor	Descrição
1	Chefe
2	Cônjuge
3	Filho(a)
4	Parente
5	Agregado
6	Empregado Residente
7	Visitante não Residente

Categorias anteriores

Valor	Descrição
1	Pessoa Responsável
2	Cônjuge/Companheiro(a)
3	Filho(a)/Enteado(a)
4	Outro Parente / Agregado
5	Empregado Residente

Valor	Descrição
6	Outros (visitante não residente / parente do empregado)

Categorias novas: [Teste: Checar se existe algum número < 1 ou > 6. Se encontrar, retornar erro indicando em qual linha.]

```
In [65]: if not impressao:
             #Counting for check "SIT_FAM"
             display(od1977['SIT_FAM'].value_counts())
```

```
1    5458
3    5034
2    2295
4    1395
5     465
6     166
0     140
7      47
dtype: int64
```

```
In [66]: #Replacing the values 5 for 4
         od1977.loc[od1977['SIT_FAM']==5,'SIT_FAM'] = 4
         #Replacing the values 6 for 5
         od1977.loc[od1977['SIT_FAM']==6,'SIT_FAM'] = 5
         #Replacing the values 7 for 6
         od1977.loc[od1977['SIT_FAM']==7,'SIT_FAM'] = 6
```

```
In [67]: if not impressao:
             #Counting "CD_RENFAM" in order to compare the values before and after the replacement
             display(od1977['SIT_FAM'].value_counts())
```

```
1    5458
3    5034
2    2295
4    1860
5     166
0     140
6      47
dtype: int64
```

```
In [68]: #Verifying value interval for check - conditions: "SIT_FAM < 1" and "SIT_FAM > 6"
         #od1977[(od1977['SIT_FAM']<1) | (od1977['SIT_FAM']>6)]
         verifica_RANGE(od1977, 'SIT_FAM', 1, 6)
```

```
0    140
dtype: int64
```

0.28 "IDADE"

Nada há que se fazer em relação aos dados da coluna "IDADE"

0.29 Passo 20: "SEXO"

Substituir valores da coluna "SEXO"

- Substituir todos valores **2** por **0**

Valor	Descrição
1	Masculino
2	Feminino

Categorias anteriores

Valor	Descrição
0	Feminino
1	Masculino

Categorias novas [Teste: Checar se existe algum número diferente de 0 ou 1. Se encontrar, retornar erro indicando em qual linha.]

```
In [69]: if not impressao:
             #Counting for check "SEXO"
             display(od1977['SEXO'].value_counts())
```

```
1    8222
2    6638
0     140
dtype: int64
```

```
In [70]: #Replacing the values 2 for 0
         od1977.loc[od1977['SEXO']==2,'SEXO'] = 0
```

```
In [71]: if not impressao:
             #Counting "SEXO" in order to compare the values before and after the replacement
             display(od1977['SEXO'].value_counts())
```

```
1    8222
0    6778
dtype: int64
```

```
In [72]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1977, 'SEXO')
```

```
Total de erros encontrados: 0
```

0.30 Passo 21: "ESTUDA"

CRIAR!

Se zona da escola for zero (==0) então não estuda (0), senão, não estuda (1)

Valor	Descrição
0	Não estuda
1	Estuda

Categorias novas [Teste: Checar se existe algum número diferente de 0 ou 1. Se encontrar, retornar erro indicando em qual linha.]

```
In [73]: if not impressao:
             #Counting for check "ESTUDA"
             display(od1977['ESTUDA'].value_counts())
```

```
Series([], dtype: int64)
```

```
In [74]: #Replacing the values 2 for 0
         ### implementar rotina do estuda
```

```
In [75]: if not impressao:
             #Counting "ESTUDA" in order to compare the values before and after the replacement
             display(od1977['ESTUDA'].value_counts())
```

```
Series([], dtype: int64)
```

```
In [76]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1977, 'ESTUDA')
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-76-dd78a539b8bb> in <module>()
          1 #Verifying if there was left some value other than 0 or 1
    ----> 2 verifica_DUMMY(od1977, 'ESTUDA')


        <ipython-input-3-bdf264bbc98a> in verifica_DUMMY(data_frame, nome_variavel)
          7     contador_de_erros = 0
          8     for index, value in data_frame.iterrows():
    ----> 9         if int(value[nome_variavel]) != 1 and int(value[nome_variavel]) != 0:
         10             if not impressao:
         11                 print("Erro encontrado no registro " + str(index+1) + ".")


        ValueError: cannot convert float NaN to integer
```

0.31 Passo 22: "GRAU_INSTR"

Substituir valores da coluna "GRAU_INSTR"

- Substituir todos valores **2** por **1**
- Substituir todos valores **3** por **1**
- Substituir todos valores **4** por **1**
- Substituir todos valores **5** por **2**
- Substituir todos valores **6** por **2**
- Substituir todos valores **7** por **3**
- Substituir todos valores **8** por **3**
- Substituir todos valores **9** por **4**

Valor	Descrição
1	Sem Instrução
2	Primário Incompleto
3	Primário Completo
4	Ginasial Incompleto
5	Ginasial Completo
6	Colegial Incompleto
7	Colegial Completo
8	Universitário Incompleto
9	Universitário Completo

Categorias anteriores:

Valor	Descrição
0	Não declarou
1	Não-Alfabetizado/Fundamental Incompleto
2	Fundamental Completo/Médio Incompleto
3	Médio Completo/Superior Incompleto
4	Superior completo

Categorias novas [Teste: Checar se existe algum número < 1 ou > 4. Se encontrar, retornar erro indicando em qual linha.]

```
In [77]: if not impressao:
             #Counting for check "GRAU_INSTR"
             display(od1977['GRAU_INSTR'].value_counts())
```

```
3    2527
4    2348
2    2056
7    1772
5    1432
1    1310
6    1171
```

```
9    1124
8    1102
0     158
dtype: int64
```

```
In [78]: #Replacing the values 2 for 1
         od1977.loc[od1977['GRAU_INSTR']==2,'GRAU_INSTR'] = 1
         #Replacing the values 3 for 1
         od1977.loc[od1977['GRAU_INSTR']==3,'GRAU_INSTR'] = 1
         #Replacing the values 4 for 1
         od1977.loc[od1977['GRAU_INSTR']==4,'GRAU_INSTR'] = 1
         #Replacing the values 5 for 2
         od1977.loc[od1977['GRAU_INSTR']==5,'GRAU_INSTR'] = 2
         #Replacing the values 6 for 2
         od1977.loc[od1977['GRAU_INSTR']==6,'GRAU_INSTR'] = 2
         #Replacing the values 7 for 3
         od1977.loc[od1977['GRAU_INSTR']==7,'GRAU_INSTR'] = 3
         #Replacing the values 8 for 3
         od1977.loc[od1977['GRAU_INSTR']==8,'GRAU_INSTR'] = 3
         #Replacing the values 9 for 4
         od1977.loc[od1977['GRAU_INSTR']==9,'GRAU_INSTR'] = 4
```

```
In [79]: if not impressao:
             #Counting "GRAU_INSTR" in order to compare the values before and after the replacement
             display(od1977['GRAU_INSTR'].value_counts())
```

```
1    8241
3    2874
2    2603
4    1124
0     158
dtype: int64
```

```
In [80]: #Verifying value interval for check - conditions: "GRAU_INSTR < 1" and "GRAU_INSTR > 4"
         #od1977[(od1977['GRAU_INSTR']<1) | (od1977['GRAU_INSTR']>4)]
         verifica_RANGE(od1977, 'GRAU_INSTR', 1, 4)
```

```
0    158
dtype: int64
```

0.32 Passo 23: "OCUP"

Substituir valores da coluna "OCUP"

- Substituir todos valores **1** por **7**
- Substituir todos valores **2** por **6**
- Substituir todos valores **4** por **5**
- Substituir todos valores **5** por **4**
- Substituir todos valores **6** por **2**
- Substituir todos valores **7** em diante por **1**

Valor	Descrição
1	Estudante
2	Prendas Domésticas
3	Aposentado
4	Sem Ocupação (nunca trabalhou)
5	Desempregado
6	Em licença
7 em diante	diversas profissões

Categorias anteriores:

Valor	Descrição
1	Tem trabalho
2	Em licença médica
3	Aposentado / pensionista
4	Desempregado
5	Sem ocupação
6	Dona de casa
7	Estudante

Categorias novas [Teste: Checar se existe algum número < 0 ou > 7. Se encontrar, retornar erro indicando em qual linha.]

```
In [81]: if not impressao:
             #Counting for check "OCUP"
             display(od1977['OCUP'].value_counts())
```

```
2     2073
1     2072
13    1639
4      889
14     829
12     802
11     669
3      561
0      509
31     375
18     365
51     344
15     319
16     303
17     283
5      280
53     274
43     253
```

```
32     234
44     214
33     179
34     167
22     166
36     161
23     152
55     132
24     105
52     104
42      86
54      79
21      76
6       71
46      64
35      57
41      38
45      30
25      27
56      12
59       4
26       2
57       1
dtype: int64
```

```
In [82]: #Replacing the values 1 for 7
         od1977.loc[od1977['OCUP']==1,'OCUP'] = 7
         #Replacing the values 2 for 9
         od1977.loc[od1977['OCUP']==2,'OCUP'] = 9
         #Replacing the values 4 for 8
         od1977.loc[od1977['OCUP']==4,'OCUP'] = 8
         #Replacing the values 5 for 4
         od1977.loc[od1977['OCUP']==5,'OCUP'] = 4
         #Replacing the values 8 for 5
         od1977.loc[od1977['OCUP']==8,'OCUP'] = 5
         #Replacing the values 6 for 2
         od1977.loc[od1977['OCUP']==6,'OCUP'] = 2
         #Replacing the values 9 for 6
         od1977.loc[od1977['OCUP']==9,'OCUP'] = 6
         #Replacing the values > 6 for 7
         od1977.loc[od1977['OCUP']>7,'OCUP'] = 1
```

```
In [83]: if not impressao:
             #Counting "OCUP" in order to compare the values before and after the replacement
             display(od1977['OCUP'].value_counts())
```

```
1    8545
6    2073
7    2072
5     889
3     561
0     509
4     280
2      71
```

```
dtype: int64
```

```
In [84]: #Verifying value interval for check - conditions: "OCUP < 1" and "OCUP > 7"
         #od1977[(od1977['OCUP']<1) | (od1977['OCUP']>7)]
         verifica_RANGE(od1977, 'OCUP', 1, 7)
```

```
0    509
dtype: int64
```

0.33 Passo 24: "SETOR_ATIV"

Substituir valores da coluna "SETOR_ATIV"

Na coluna "SETOR_ATIV", linha i, ler o valor da linha i da coluna "SETOR_ATIV", daí, buscar o mesmo valor na coluna "COD" do arquivo setor_ativ-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "COD_UNIF"

Categorias anteriores

ver arquivo .csv

Valor	Descrição
0	Não respondeu
1	Agrícola
2	Construção Civil
3	Indústria
4	Comércio
5	Administração Pública
6	Serviços de Transporte
7	Serviços
8	Serviços Autônomos
9	Outros
10	Não se aplica

Categorias novas [Teste: Checar se existe algum número < 1 ou > 10. Se encontrar, retornar erro indicando em qual linha.]

```
In [85]: if not impressao:
             #Counting for check "SETOR_ATIV"
             display(od1977['SETOR_ATIV'].value_counts())
```

```
10    5695
4     2591
3     1956
7     1929
8     1095
```

```
5      729
0      509
6      214
2      198
9       54
1       30
dtype: int64
```

```
In [86]: #Getting from the csv file the "CD_UNIF" (unified code for activity sector) correspondent to t
         od1977['SETOR_ATIV'] = od1977.apply(lambda row: consulta_refext(row, 'setor_ativ-1977.csv', 'C
```

```
In [87]: if not impressao:
             #Counting "SETOR_ATIV" in order to compare the values before and after the replacement
             display(od1977['SETOR_ATIV'].value_counts())
```

```
10    5695
4     2591
3     1956
7     1929
8     1095
5      729
0      509
6      214
2      198
9       54
1       30
dtype: int64
```

```
In [88]: #Verifying value interval for check - conditions: "SETOR_ATIV < 0" and "SETOR_ATIV > 10"
         #od1977[(od1977['SETOR_ATIV']<0) | (od1977['SETOR_ATIV']>10)]
         verifica_RANGE(od1977, 'SETOR_ATIV', 0, 10)
```

```
Series([], dtype: int64)
```

0.34 Passo 25: "CD_RENIND"

Checar se existe algum erro na coluna "CD_RENIND"

Valor	Descrição
1	Tem renda
2	Não tem renda
3	Não declarou

Categorias [Teste: Checar se existe algum número < 1 ou > 3. Se encontrar, retornar erro indicando em qual linha.]

```
In [89]: if not impressao:
             #Counting for check "CD_RENIND"
             display(od1977['CD_RENIND'].value_counts())
```

```
0    9951
1    4769
2     280
dtype: int64
```

```
In [90]: #Verifying value interval for check - conditions: "CD_RENIND < 1" and "CD_RENIND > 3"
         #od1977[(od1977['CD_RENIND']<1) | (od1977['CD_RENIND']>3)]
         verifica_RANGE(od1977, 'CD_RENIND', 1, 3)
```

```
0    9951
dtype: int64
```

0.35 "REN_IND"

Nada há que se fazer em relação aos dados da coluna "REN_IND"

0.36 Passo 26: "UCOD_ESC"

Na coluna "UCOD_ESC", linha i, ler o valor da linha i da coluna "ZONA_ESC", daí, buscar o mesmo valor na coluna "Zona 1977" do arquivo UCOD-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "UCOD_ESC"

[Teste: no banco completo, checar se o min == 1 e o max == 67]

```
In [91]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ESC" code
         od1977['UCOD_ESC'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 1977
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-91-e718fcf85c06> in <module>()
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ESC" code
----> 2 od1977['UCOD_ESC'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 19


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
3687                     if reduce is None:
3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
3690             else:
3691                 return self._apply_broadcast(f, axis)


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
3777             try:
3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
3780                     keys.append(v.name)
3781             except Exception as e:
```

```
    <ipython-input-91-e718fcf85c06> in <lambda>(row)
      1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ESC" code
----> 2 od1977['UCOD_ESC'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 19


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     13         return row[name_col_filt]
     14     data_frame = pd.read_csv(name_file,sep=';')
---> 15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/core/series.py in wrapper(self)
     75             return converter(self.iloc[0])
     76         raise TypeError(
---> 77             "cannot convert the series to {0}".format(str(converter)))
     78     return wrapper
     79


    TypeError: ("cannot convert the series to <class 'int'>", 'occurred at index 5459')
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_ESC" column) - count, mean, std, min and max
            display(od1977['UCOD_ESC'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_ESC"
            display(od1977['UCOD_ESC'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "UCOD_ESC < 1" and "UCOD_ESC > 67"
        #The 'error' returns must be related to "UCOD_ESC" == 0, that is, trips that are not school pur
        #od1977[(od1977['UCOD_ESC']<1) | (od1977['UCOD_ESC']>67)]
        verifica_RANGE(od1977, 'UCOD_ESC', 1, 67)
```

0.37 Passo 27: "ZONA_ESC"

Checar se existe algum erro

Categorias:

1 a 243

[Teste: Checar se existe algum número < 1 ou > 243. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "ZONA_ESC < 1" and "ZONA_ESC > 243"
        #The 'error' returns must be related to "ZONA_ESC" == 0, that is, trips that are not school pur
        #od1977[(od1977['ZONA_ESC']<1) | (od1977['ZONA_ESC']>254)]
        verifica_RANGE(od1977, 'ZONA_ESC', 1, 243)
```

0.38 Passo 28: "SUBZONA_ESC"

Checar se existe algum erro

Categorias:

1 a 633

[Teste: Checar se existe algum número < 1 ou > 633. Se encontrar, retornar erro indicando em qual linha.]

```
In [92]: #Verifying value interval for check - conditions: "SUBZONA_ESC < 1" and "SUBZONA_ESC > 633"
         #od1977[(od1977['SUBZONA_ESC']<1) | (od1977['SUBZONA_ESC']>633)]
         verifica_RANGE(od1977, 'SUBZONA_ESC', 1, 633)
```

```
0      11072
999       21
dtype: int64
```

0.39 Passo 29: "MUN_ESC"

Checar se existe algum erro

Categorias

1 a 27

[Teste: Checar se existe algum número < 1 ou > 27. Se encontrar, retornar erro indicando em qual linha.]

```
In [93]: #Verifying value interval for check - conditions: "MUN_ESC < 1" and "MUN_ESC > 27"
         #The 'error' returns must be related to "MUN_ESC" == 0, that is, trips that are not school pur
         #od1977[(od1977['MUN_ESC']<1) | (od1977['MUN_ESC']>27)]
         verifica_RANGE(od1977, 'MUN_ESC', 1, 27)
```

```
0    11093
dtype: int64
```

0.40 Passo 30: "CO_ESC_X"

Na coluna "CO_ESC_X", linha i, ler o valor da linha i da coluna "SUBZONA_ESC", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"

```
In [94]: #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_ESC" code
         od1977['CO_ESC_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', '
```

```
---------------------------------------------------------------------------

OSError                                   Traceback (most recent call last)

<ipython-input-94-6c89f56be538> in <module>()
  1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_ESC" code
```

```
----> 2 od1977['CO_ESC_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv',


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-94-6c89f56be538> in <lambda>(row)
      1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_ESC" code
----> 2 od1977['CO_ESC_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv',


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):
```

```
    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
    697     def _make_engine(self, engine='c'):
    698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
    700         else:
    701             if engine == 'python':


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1064         kwds['allow_leading_cols'] = self.index_col is not False
   1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
   1067
   1068         # XXX


    pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


    pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


    OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 24')
```

0.41 Passo 31: "CO_ESC_Y"

Na coluna "CO_ESC_Y", linha i, ler o valor da linha i da coluna "SUBZONA_ESC", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"

```
In [95]: #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_ESC" code
         od1977['CO_ESC_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', 'S


    ---------------------------------------------------------------------------

    OSError                                   Traceback (most recent call last)

    <ipython-input-95-deef12fbf0b9> in <module>()
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_ESC" code
----> 2 od1977['CO_ESC_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv',


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)
```

```
    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-95-deef12fbf0b9> in <lambda>(row)
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_ESC" code
----> 2 od1977['CO_ESC_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv',


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
    697     def _make_engine(self, engine='c'):
    698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
    700         else:
    701             if engine == 'python':


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1064         kwds['allow_leading_cols'] = self.index_col is not False
```

```
   1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
   1067
   1068         # XXX


pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 24')
```

0.42 Passo 32: "UCOD_TRAB1"

Na coluna "UCOD_TRAB1", linha i, ler o valor da linha i da coluna "ZONA_TRAB1", daí, buscar o mesmo valor na coluna "Zona 1977" do arquivo UCOD-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "UCOD_TRAB1"

[Teste: no banco completo, checar se o min == 1 e o max == 67]

```
In [96]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB1" code
         od1977['UCOD_TRAB1'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 19


---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-96-c9316b02efdf> in <module>()
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB1" code
----> 2 od1977['UCOD_TRAB1'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
3687                     if reduce is None:
3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
3690             else:
3691                 return self._apply_broadcast(f, axis)


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
3777             try:
3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
3780                     keys.append(v.name)
3781             except Exception as e:


<ipython-input-96-c9316b02efdf> in <lambda>(row)
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB1" code
```

```
----> 2 od1977['UCOD_TRAB1'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona

    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     13         return row[name_col_filt]
     14     data_frame = pd.read_csv(name_file,sep=';')
---> 15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])

    /usr/local/lib/python3.4/dist-packages/pandas/core/series.py in wrapper(self)
     75             return converter(self.iloc[0])
     76         raise TypeError(
---> 77             "cannot convert the series to {0}".format(str(converter)))
     78     return wrapper
     79

    TypeError: ("cannot convert the series to <class 'int'>", 'occurred at index 459')
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_TRAB1" column) - count, mean, std, min and max
            display(od1977['UCOD_TRAB1'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_TRAB1"
            display(od1977['UCOD_TRAB1'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "UCOD_TRAB1 < 1" and "UCOD_TRAB1 > 67"
        #The 'error' returns must be related to "UCOD_TRAB1" == 0, that is, trips that are not work pur
        #od1977[(od1977['UCOD_TRAB1']<1) | (od1977['UCOD_TRAB1']>67)]
        verifica_RANGE(od1977, 'UCOD_TRAB1', 1, 67)
```

0.43 Passo 33: "ZONA_TRAB1"

Checar se existe algum erro

Categorias:

1 a 243

[Teste: Checar se existe algum número < 1 ou > 243. Se encontrar, retornar erro indicando em qual linha.]

```
In [97]: #Verifying value interval for check - conditions: "ZONA_TRAB1 < 1" and "ZONA_TRAB1 > 243"
         #The 'error' returns must be related to "ZONA_TRAB1"==0, that is, trips that are not work purp
         #od1977[(od1977['ZONA_TRAB1']<1) | (od1977['ZONA_TRAB1']>243)]
         verifica_RANGE(od1977, 'ZONA_TRAB1', 1, 243)
```

```
0      6289
999      59
dtype: int64
```

0.44 Passo 34: "SUBZONA_TRAB1"

Checar se existe algum erro

Categorias:

1 a 633

[Teste: Checar se existe algum número < 1 ou > 633. Se encontrar, retornar erro indicando em qual linha.]

```
In [98]: #Verifying value interval for check - conditions: "SUBZONA_TRAB1 < 1" and "SUBZONA_TRAB1 > 633
         #od1977[(od1977['SUBZONA_TRAB1']<1) | (od1977['SUBZONA_TRAB1']>633)]
         verifica_RANGE(od1977, 'SUBZONA_TRAB1', 1, 633)
```

```
0      6289
999      59
dtype: int64
```

0.45 Passo 35: "MUN_TRAB1"

Checar se existe algum erro

Categorias

1 a 27

[Teste: Checar se existe algum número < 1 ou > 27. Se encontrar, retornar erro indicando em qual linha.]

```
In [99]: #Verifying value interval for check - conditions: "MUN_TRAB1 < 1" ou de "MUN_TRAB1 > 27"
         #The 'error' returns must be related to "MUN_TRAB1" == 0, that is, trips that are not work pur
         #od1977[(od1977['MUN_TRAB1']<1) | (od1977['MUN_TRAB1']>27)]
         verifica_RANGE(od1977, 'MUN_TRAB1', 1, 27)
```

```
0    6348
dtype: int64
```

0.46 Passo 36: "CO_TRAB1_X"

Na coluna "CO_TRAB1_X", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB1", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"

```
In [100]: #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_TRAB1" code
          od1977['CO_TRAB1_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv'
```

```
---------------------------------------------------------------------------

OSError                                   Traceback (most recent call last)

<ipython-input-100-02680f5a7a98> in <module>()
  1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_TRAB1" code
```

```
----> 2 od1977['CO_TRAB1_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-100-02680f5a7a98> in <lambda>(row)
      1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_TRAB1" code
----> 2 od1977['CO_TRAB1_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):
```

```
    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
    697     def _make_engine(self, engine='c'):
    698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
    700         else:
    701             if engine == 'python':


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1064         kwds['allow_leading_cols'] = self.index_col is not False
   1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
   1067
   1068         # XXX


    pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


    pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


    OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 0')
```

0.47 Passo 37: "CO_TRAB1_Y"

Na coluna "CO_TRAB1_Y", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB1", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"

```
In [101]: #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_TRAB1" code
          od1977['CO_TRAB1_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv'


    ---------------------------------------------------------------------------

    OSError                                   Traceback (most recent call last)

    <ipython-input-101-f84f58067178> in <module>()
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_TRAB1" code
----> 2 od1977['CO_TRAB1_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)
```

```
    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-101-f84f58067178> in <lambda>(row)
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_TRAB1" code
----> 2 od1977['CO_TRAB1_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
    697     def _make_engine(self, engine='c'):
    698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
    700         else:
    701             if engine == 'python':


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1064         kwds['allow_leading_cols'] = self.index_col is not False
```

```
   1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
   1067
   1068         # XXX


pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 0')
```

0.48 Passo 38: "UCOD_TRAB2"

Na coluna "UCOD_TRAB2", linha i, ler o valor da linha i da coluna "ZONA_TRAB2", daí, buscar o mesmo valor na coluna "Zona 1977" do arquivo UCOD-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "UCOD_TRAB2"

[Teste: no banco completo, checar se o min == 1 e o max == 67]

```
In [102]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB2" code
          od1977['UCOD_TRAB2'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 19


---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-102-56eccb0cd35e> in <module>()
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB2" code
----> 2 od1977['UCOD_TRAB2'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
3687                     if reduce is None:
3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
3690             else:
3691                 return self._apply_broadcast(f, axis)


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
3777             try:
3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
3780                     keys.append(v.name)
3781             except Exception as e:


<ipython-input-102-56eccb0cd35e> in <lambda>(row)
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB2" code
```

```
----> 2 od1977['UCOD_TRAB2'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona

    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     13         return row[name_col_filt]
     14     data_frame = pd.read_csv(name_file,sep=';')
---> 15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])

    /usr/local/lib/python3.4/dist-packages/pandas/core/series.py in wrapper(self)
     75             return converter(self.iloc[0])
     76         raise TypeError(
---> 77             "cannot convert the series to {0}".format(str(converter)))
     78     return wrapper
     79

    TypeError: ("cannot convert the series to <class 'int'>", 'occurred at index 2281')
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_TRAB2" column) - count, mean, std, min and max
            display(od1977['UCOD_TRAB2'].describe())
```

```
In [103]: if not impressao:
              #Count for check "UCOD_TRAB2"
              display(od1977['UCOD_TRAB2'].value_counts())
```

```
Series([], dtype: int64)
```

```
In [104]: #Verifying value interval for check - conditions: "UCOD_TRAB2 < 1" and "UCOD_TRAB2 > 67"
          #The 'error' returns must be related to "UCOD_TRAB2" == 0, that is, trips that are not work p
          #od1977[(od1977['UCOD_TRAB2']<1) | (od1977['UCOD_TRAB2']>67)]
          verifica_RANGE(od1977, 'UCOD_TRAB2', 1, 67)
```

```
Series([], dtype: int64)
```

0.49 Passo 39: "ZONA_TRAB2"

Checar se existe algum erro

Categorias:

1 a 243

[Teste: Checar se existe algum número < 1 ou > 243. Se encontrar, retornar erro indicando em qual linha.]

```
In [105]: #Verifying value interval for check - conditions: "ZONA_TRAB2 < 1" and "ZONA_TRAB2 > 243
          #The 'error' returns must be related to "ZONA_TRAB2"==0, that is, trips that are not work pur
          #od1977[(od1977['ZONA_TRAB2']<1) | (od1977['ZONA_TRAB2']>243)]
          verifica_RANGE(od1977, 'ZONA_TRAB2', 1, 243)
```

```
0      14678
999        6
dtype: int64
```

0.50 Passo 40: "SUBZONA_TRAB2"

Checar se existe algum erro

Categorias:

1 a 633

[Teste: Checar se existe algum número < 1 ou > 633. Se encontrar, retornar erro indicando em qual linha.]

```
In [106]: #Verifying value interval for check - conditions: "SUBZONA_TRAB2 < 1" and "SUBZONA_TRAB2 > 63
          #od1977[(od1977['SUBZONA_TRAB2']<1) | (od1977['SUBZONA_TRAB2']>633)]
          verifica_RANGE(od1977, 'SUBZONA_TRAB2', 1, 633)
```

```
0      14678
999        6
dtype: int64
```

0.51 Passo 41: "MUN_TRAB2"

Checar se existe algum erro

Categorias

1 a 27

[Teste: Checar se existe algum número < 1 ou > 27. Se encontrar, retornar erro indicando em qual linha.]

```
In [107]: #Verifying value interval for check - conditions: "MUN_TRAB2 < 1" ou de "MUN_TRAB2 > 27"
          #The 'error' returns must be related to "MUN_TRAB2" == 0, that is, trips that are not work pu
          #od1977[(od1977['MUN_TRAB2']<1) | (od1977['MUN_TRAB2']>27)]
          verifica_RANGE(od1977, 'MUN_TRAB2', 1, 27)
```

```
0    14684
dtype: int64
```

0.52 Passo 42: "CO_TRAB2_X"

Na coluna "CO_TRAB2_X", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB2", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"

```
In [108]: #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_TRAB2" code
          od1977['CO_TRAB2_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv'
```

```
---------------------------------------------------------------------------

OSError                                   Traceback (most recent call last)

<ipython-input-108-839efe0a6a0e> in <module>()
  1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_TRAB2" code
```

```
----> 2 od1977['CO_TRAB2_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
   3687                     if reduce is None:
   3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
   3690             else:
   3691                 return self._apply_broadcast(f, axis)


    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-108-839efe0a6a0e> in <lambda>(row)
      1 #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_TRAB2" code
----> 2 od1977['CO_TRAB2_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):
```

```
/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
697     def _make_engine(self, engine='c'):
698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
700         else:
701             if engine == 'python':


/usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
1064         kwds['allow_leading_cols'] = self.index_col is not False
1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
1067
1068         # XXX


pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 477')
```

0.53 Passo 43: "CO_TRAB2_Y"

Na coluna "CO_TRAB2_Y", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB2", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"

```
In [109]: #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_TRAB2" code
          od1977['CO_TRAB2_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv'
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)

<ipython-input-109-8248ca59deec> in <module>()
  1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_TRAB2" code
----> 2 od1977['CO_TRAB2_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
3687                     if reduce is None:
3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
3690             else:
3691                 return self._apply_broadcast(f, axis)
```

```
    /usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
   3777             try:
   3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
   3780                     keys.append(v.name)
   3781             except Exception as e:


    <ipython-input-109-8248ca59deec> in <lambda>(row)
      1 #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_TRAB2" code
----> 2 od1977['CO_TRAB2_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv


    <ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
     12     if row [name_col_filt]==0:
     13         return row[name_col_filt]
---> 14     data_frame = pd.read_csv(name_file,sep=';')
     15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep,
    468                     skip_blank_lines=skip_blank_lines)
    469
--> 470         return _read(filepath_or_buffer, kwds)
    471
    472     parser_f.__name__ = name


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    244
    245     # Create the parser.
--> 246     parser = TextFileReader(filepath_or_buffer, **kwds)
    247
    248     if (nrows is not None) and (chunksize is not None):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    560             self.options['has_index_names'] = kwds['has_index_names']
    561
--> 562         self._make_engine(self.engine)
    563
    564     def _get_options_with_defaults(self, engine):


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in _make_engine(self, engine)
    697     def _make_engine(self, engine='c'):
    698         if engine == 'c':
--> 699             self._engine = CParserWrapper(self.f, **self.options)
    700         else:
    701             if engine == 'python':


    /usr/local/lib/python3.4/dist-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1064         kwds['allow_leading_cols'] = self.index_col is not False
```

```
   1065
-> 1066         self._reader = _parser.TextReader(src, **kwds)
   1067
   1068         # XXX


pandas/parser.pyx in pandas.parser.TextReader.__cinit__ (pandas/parser.c:3163)()


pandas/parser.pyx in pandas.parser.TextReader._setup_parser_source (pandas/parser.c:5779)()


OSError: ("File b'COORD-SUBZONA-1977.csv' does not exist", 'occurred at index 477')
```

0.54 Passo 44: "ID_VIAG"

Construir o "ID_VIAG"

Na coluna "ID_VIAG", linha i, ler o valor da linha i da coluna "ID_PESS", e concatenar esse valor (12 dígitos) com o número da pessoa, que é o valor da linha i da coluna "NO_VIAG" (com 2 dígitos).

Resultado será um ID_VIAG, que pode se repetir nas linhas, 14 dígitos.

```
In [110]: #Generating "ID_VIAG" from the concatenation of "ID_PESS" and "NO_VIAG" variables
          od1977['ID_VIAG'] = od1977.apply(gera_ID_VIAG, axis=1)
```

0.55 Passo 45: "F_VIAG"

Excluir a coluna "F_VIAG", porque as viagens são numeradas, então já se saber pelo NO_VIAG qual é a primeira do indivíduo.

```
In [111]: od1977 = od1977.drop('F_VIAG', 1)
```

```
In [112]: #Storing the variables list in the "cols" variable
          cols = od1977.columns.tolist()
          if not impressao:
              #printing "cols" variable to check if the reorder operation was effective
              display(cols)
```

```
['ANO',
 'CD_ENTRE',
 'DIA_SEM',
 'UCOD_DOM',
 'ZONA_DOM',
 'SUBZONA_DOM',
 'MUN_DOM',
 'CO_DOM_X',
 'CO_DOM_Y',
 'ID_DOM',
 'F_DOM',
 'FE_DOM',
 'NO_DOM',
 'TIPO_DOM',
```

```
'TOT_FAM',
'ID_FAM',
'F_FAM',
'FE_FAM',
'NO_FAM',
'COND_MORA',
'QT_AUTO',
'QT_BICI',
'QT_MOTO',
'CD_RENFAM',
'REN_FAM',
'ID_PESS',
'F_PESS',
'FE_PESS',
'NO_PESS',
'SIT_FAM',
'IDADE',
'SEXO',
'ESTUDA',
'GRAU_INSTR',
'OCUP',
'SETOR_ATIV',
'CD_RENIND',
'REN_IND',
'UCOD_ESC',
'ZONA_ESC',
'SUBZONA_ESC',
'MUN_ESC',
'CO_ESC_X',
'CO_ESC_Y',
'UCOD_TRAB1',
'ZONA_TRAB1',
'SUBZONA_TRAB1',
'MUN_TRAB1',
'CO_TRAB1_X',
'CO_TRAB1_Y',
'UCOD_TRAB2',
'ZONA_TRAB2',
'SUBZONA_TRAB2',
'MUN_TRAB2',
'CO_TRAB2_X',
'CO_TRAB2_Y',
'ID_VIAG',
'FE_VIAG',
'NO_VIAG',
'TOT_VIAG',
'UCOD_ORIG',
'ZONA_ORIG',
'SUBZONA_ORIG',
'MUN_ORIG',
'CO_ORIG_X',
'CO_ORIG_Y',
'UCOD_DEST',
'ZONA_DEST',
```

```
 'SUBZONA_DEST',
 'MUN_DEST',
 'CO_DEST_X',
 'CO_DEST_Y',
 'DIST_VIAG',
 'MOTIVO_ORIG',
 'MOTIVO_DEST',
 'MODO1',
 'MODO2',
 'MODO3',
 'MODO4',
 'MODO_PRIN',
 'TIPO_VIAG',
 'H_SAIDA',
 'MIN_SAIDA',
 'ANDA_ORIG',
 'H_CHEG',
 'MIN_CHEG',
 'ANDA_DEST',
 'DURACAO',
 'TIPO_EST_AUTO',
 'VALOR_EST_AUTO']
```

0.56 "FE_VIAG" E "NO_VIAG"

Nada há que se fazer em relação aos dados das colunas "FE_VIAG" e "NO_VIAG"

0.57 "TOT_VIAG"

CALCULAR

```
In [113]: if not impressao:
              #Describing data ("TOT_VIAG" column) - count, mean, std, min and max
              display(od1977['TOT_VIAG'].describe())
          display(od1977[:30][['ID_PESS','NO_VIAG','TOT_VIAG']])
```

```
count     0
mean    NaN
std     NaN
min     NaN
25%     NaN
50%     NaN
75%     NaN
max     NaN
Name: TOT_VIAG, dtype: float64
```

```
         ID_PESS  NO_VIAG  TOT_VIAG
0   100100010101        1       NaN
1   100100010101        2       NaN
```

```
2   100100010101        3       NaN
3   100100010101        4       NaN
4   100100010102        0       NaN
5   100100020101        1       NaN
6   100100020101        2       NaN
7   100100020101        3       NaN
8   100100020101        4       NaN
9   100100020102        1       NaN
10  100100020102        2       NaN
11  100100020102        3       NaN
12  100100020102        4       NaN
13  100100030101        1       NaN
14  100100030101        2       NaN
15  100100030101        3       NaN
16  100100030101        4       NaN
17  100100030102        1       NaN
18  100100030102        2       NaN
19  100100030102        3       NaN
20  100100030102        4       NaN
21  100100040101        1       NaN
22  100100040101        2       NaN
23  100100040102        0       NaN
24  100100040103        1       NaN
25  100100040103        2       NaN
26  100100040103        3       NaN
27  100100040104        1       NaN
28  100100040104        2       NaN
29  100100050000        0       NaN
```

```
In [114]: def atrib_tot_viag(row):
              od1977.loc[od1977['ID_PESS']==row['ID_PESS'],'TOT_VIAG'] = row['NO_VIAG']
              #print('id_pessoa: ' + str(row['ID_PESS']) + ' | no_viag:' + str(row['NO_VIAG']))
              #print(row)

          od1977.loc[:,['ID_PESS','NO_VIAG']].groupby(['ID_PESS'],sort=False).agg({'NO_VIAG':max,'ID_PES
          #od1977[od1977['ID_PESS']==20002030404][['ID_PESS','NO_VIAG']]
```

```
Out[114]: ID_PESS
          100100010101    None
          100100010102    None
          100100020101    None
          100100020102    None
          100100030101    None
          100100030102    None
          100100040101    None
          100100040102    None
          100100040103    None
          100100040104    None
          100100050000    None
          100100060101    None
          100100060102    None
          100100060201    None
          100100060301    None
          100100060302    None
```

```
100100060401    None
100100060402    None
100100060501    None
100100060601    None
100100060602    None
100100060603    None
100100060604    None
100100060701    None
100100060702    None
100100070101    None
100100070102    None
100100070201    None
100100070202    None
100100070301    None
                ...
101800340102    None
101800340103    None
101800340104    None
101800340105    None
101800350101    None
101800350102    None
101800350103    None
101800350104    None
101800350105    None
101800350106    None
101800360101    None
101800360102    None
101800360103    None
101800370101    None
101800370102    None
101800380101    None
101800380102    None
101800380103    None
101800380104    None
101800380105    None
101800390101    None
101800390102    None
101800390103    None
101800390104    None
101800400101    None
101800400102    None
101800410101    None
101800410102    None
101800410103    None
101800410104    None
dtype: object
```

```
In [115]: display(od1977.loc[:90,['ID_PESS','NO_VIAG','TOT_VIAG']])
          #Verificar as viagens do ID_PESS = 100100070403
          #display(od1977.loc[od1977['ID_PESS']==100100070403,['ID_PESS','NO_VIAG','TOT_VIAG','ID_DOM',
```

	ID_PESS	NO_VIAG	TOT_VIAG
0	100100010101	1	4
1	100100010101	2	4
2	100100010101	3	4

3	100100010101	4	4
4	100100010102	0	0
5	100100020101	1	4
6	100100020101	2	4
7	100100020101	3	4
8	100100020101	4	4
9	100100020102	1	4
10	100100020102	2	4
11	100100020102	3	4
12	100100020102	4	4
13	100100030101	1	4
14	100100030101	2	4
15	100100030101	3	4
16	100100030101	4	4
17	100100030102	1	4
18	100100030102	2	4
19	100100030102	3	4
20	100100030102	4	4
21	100100040101	1	2
22	100100040101	2	2
23	100100040102	0	0
24	100100040103	1	3
25	100100040103	2	3
26	100100040103	3	3
27	100100040104	1	2
28	100100040104	2	2
29	100100050000	0	0
..	...	...	...
61	100100070102	0	0
62	100100070201	1	2
63	100100070201	2	2
64	100100070202	1	2
65	100100070202	2	2
66	100100070301	1	2
67	100100070301	2	2
68	100100070302	1	2
69	100100070302	2	2
70	100100070401	1	2
71	100100070401	2	2
72	100100070402	1	2
73	100100070402	2	2
74	100100070403	1	3
75	100100070403	2	3
76	100100070403	3	3
77	100100070403	3	3
78	100100080101	1	2
79	100100080101	2	2
80	100100090101	0	0
81	100100090102	1	3
82	100100090102	2	3
83	100100090102	3	3
84	100100100000	0	0
85	100100110101	1	3
86	100100110101	2	3

```
87  100100110101        3         3
88  100100110102        1         2
89  100100110102        2         2
90  100100110103        1         4

[91 rows x 3 columns]
```

```
In [116]: if not impressao:
              #Count for check "TOT_VIAG"
              display(od1977['TOT_VIAG'].value_counts())
```

```
2     4569
4     2947
0     2389
3     1278
6      952
5      936
1      706
8      315
7      310
9      143
10     138
11     105
12      81
13      65
14      28
20      20
16      15
37       3
dtype: int64
```

```
In [117]: #Agora uma função que irá verificar se para todo "ID_PESS" o "TOT_VIAG" é igual ao 'NO_VIAG' 
          def verifica_no_viag_tot_viag(row):
              if row['NO_VIAG'] != row['TOT_VIAG']:
                  print(row)
          od1977.loc[:,['ID_PESS','NO_VIAG','TOT_VIAG']].groupby('ID_PESS').agg({'NO_VIAG':'max','ID_PE
```

```
Out[117]: ID_PESS
          100100010101    None
          100100010102    None
          100100020101    None
          100100020102    None
          100100030101    None
          100100030102    None
          100100040101    None
          100100040102    None
          100100040103    None
          100100040104    None
          100100050000    None
          100100060101    None
          100100060102    None
          100100060201    None
          100100060301    None
```

```
100100060302    None
100100060401    None
100100060402    None
100100060501    None
100100060601    None
100100060602    None
100100060603    None
100100060604    None
100100060701    None
100100060702    None
100100070101    None
100100070102    None
100100070201    None
100100070202    None
100100070301    None
                ...
101800340102    None
101800340103    None
101800340104    None
101800340105    None
101800350101    None
101800350102    None
101800350103    None
101800350104    None
101800350105    None
101800350106    None
101800360101    None
101800360102    None
101800360103    None
101800370101    None
101800370102    None
101800380101    None
101800380102    None
101800380103    None
101800380104    None
101800380105    None
101800390101    None
101800390102    None
101800390103    None
101800390104    None
101800400101    None
101800400102    None
101800410101    None
101800410102    None
101800410103    None
101800410104    None
dtype: object
```

```
In [118]: if not impressao:
              #Describing data ("TOT_VIAG" column) - count, mean, std, min and max
              display(od1977['TOT_VIAG'].describe())
```

```
count    15000.000000
mean         3.155333
std          2.626721
```

```
min          0.000000
25%          2.000000
50%          2.000000
75%          4.000000
max         37.000000
Name: TOT_VIAG, dtype: float64
```

0.58 Passo 46: "UCOD_ORIG"

Na coluna "UCOD_ORIG", linha i, ler o valor da linha i da coluna "ZONA_ORIG", daí, buscar o mesmo valor na coluna "Zona 1977" do arquivo UCOD-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "UCOD_ORIG"

[Teste: no banco completo, checar se o min == 1 e o max == 67]

```
In [119]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ORIG" code
          od1977['UCOD_ORIG'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 19
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-119-5baee46cf6e3> in <module>()
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ORIG" code
----> 2 od1977['UCOD_ORIG'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 1


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in apply(self, func, axis, broadcas
3687                     if reduce is None:
3688                         reduce = True
-> 3689                     return self._apply_standard(f, axis, reduce=reduce)
3690             else:
3691                 return self._apply_broadcast(f, axis)


/usr/local/lib/python3.4/dist-packages/pandas/core/frame.py in _apply_standard(self, func, axis,
3777             try:
3778                 for i, v in enumerate(series_gen):
-> 3779                     results[i] = func(v)
3780                     keys.append(v.name)
3781             except Exception as e:


<ipython-input-119-5baee46cf6e3> in <lambda>(row)
  1 #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ORIG" code
----> 2 od1977['UCOD_ORIG'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 1


<ipython-input-2-e86902145311> in consulta_refext(row, name_file, name_col_ref, name_col_filt, na
 13         return row[name_col_filt]
 14     data_frame = pd.read_csv(name_file,sep=';')
---> 15     return int(data_frame[data_frame[name_col_ref]==row[name_col_filt]][name_col_search])
```

```
/usr/local/lib/python3.4/dist-packages/pandas/core/series.py in wrapper(self)
 75             return converter(self.iloc[0])
 76         raise TypeError(
---> 77             "cannot convert the series to {0}".format(str(converter)))
 78     return wrapper
 79

TypeError: ("cannot convert the series to <class 'int'>", 'occurred at index 13517')
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_ORIG" column) - count, mean, std, min and max
            display(od1977['UCOD_ORIG'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_ORIG"
            display(od1977['UCOD_ORIG'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "UCOD_ORIG < 1" and "UCOD_ORIG > 67"
        #The 'error' returns must be related to "UCOD_ORIG" == 0, that is, trips that were not made
        #od1977[(od1977['UCOD_ORIG']<1) | (od1977['UCOD_ORIG']>67)]
        verifica_RANGE(od1977, 'UCOD_ORIG', 1, 67)
```

0.59 Passo 47: "ZONA_ORIG"

Checar se existe algum erro

Categorias:

1 a 243

[Teste: Checar se existe algum número < 1 ou > 243. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "ZONA_ORIG < 1" and "ZONA_ORIG > 243"
        #The 'error' returns must be related to "ZONA_ORIG"==0, that is, trips that were not made
        #od1977[(od1977['ZONA_ORIG']<1) | (od1977['ZONA_ORIG']>243)]
        verifica_RANGE(od1977, 'ZONA_ORIG', 1, 243)
```

0.60 Passo 48: "SUBZONA_ORIG"

Checar se existe algum erro

Categorias:

1 a 633

[Teste: Checar se existe algum número < 1 ou > 633. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "SUBZONA_ORIG < 1" and "SUBZONA_ORIG > 633"
        #od1977[(od1977['SUBZONA_ORIG']<1) | (od1977['SUBZONA_ORIG']>633)]
        verifica_RANGE(od1977, 'SUBZONA_ORIG', 1, 633)
```

0.61 Passo 49: "MUN_ORIG"

Checar se existe algum erro

Categorias

1 a 27

[Teste: Checar se existe algum número < 1 ou > 27. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "MUN_ORIG < 1" ou de "MUN_ORIG > 27"
        #The 'error' returns must be related to "MUN_ORIG" == 0, that is, trips that were not made
        #od1977[(od1977['MUN_ORIG']<1) | (od1977['MUN_ORIG']>27)]
        verifica_RANGE(od1977, 'MUN_ORIG', 1, 27)
```

0.62 Passo 50: "CO_ORIG_X"

Na coluna "CO_ORIG_X", linha i, ler o valor da linha i da coluna "SUBZONA_ORIG", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"

```
In [ ]: #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_ORIG" code
        od1977['CO_ORIG_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', 'S
```

0.63 Passo 51: "CO_ORIG_Y"

a coluna "CO_ORIG_Y", linha i, ler o valor da linha i da coluna "SUBZONA_ORIG", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"

```
In [ ]: #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_ORIG" code
        od1977['CO_ORIG_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', 'S
```

0.64 Passo 52: "UCOD_DEST"

Na coluna "UCOD_DEST", linha i, ler o valor da linha i da coluna "ZONA_DEST", daí, buscar o mesmo valor na coluna "Zona 1977" do arquivo UCOD-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "UCOD_DEST"

[Teste: no banco completo, checar se o min == 1 e o max == 67]

```
In [ ]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_DEST" code
        od1977['UCOD_DEST'] = od1977.apply(lambda row: consulta_refext(row, 'UCOD-1977.csv', 'Zona 1977
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_DEST" column) - count, mean, std, min and max
            display(od1977['UCOD_DEST'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_DEST"
            display(od1977['UCOD_DEST'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "UCOD_DEST < 1" and "UCOD_DEST > 67"
        #The 'error' returns must be related to "UCOD_DEST" == 0, that is, trips that were not made
        #od1977[(od1977['UCOD_DEST']<1) | (od1977['UCOD_DEST']>67)]
        verifica_RANGE(od1977, 'UCOD_DEST', 1, 67)
```

0.65 Passo 53: "ZONA_DEST"

Checar se existe algum erro

Categorias:

1 a 243

[Teste: Checar se existe algum número < 1 ou > 243. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "ZONA_DEST < 1" and "ZONA_DEST > 243"
        #The 'error' returns must be related to "ZONA_DEST"==0, that is, trips that are not school purp
        #od1977[(od1977['ZONA_DEST']<1) | (od1977['ZONA_DEST']>243)]
        verifica_RANGE(od1977, 'ZONA_DEST', 1, 243)
```

0.66 Passo 54: "SUBZONA_DEST"

Checar se existe algum erro

Categorias:

1 a 633

[Teste: Checar se existe algum número < 1 ou > 633. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "SUBZONA_DEST < 1" and "SUBZONA_DEST > 633"
        #od1977[(od1977['SUBZONA_DEST']<1) | (od1977['SUBZONA_DEST']>633)]
        verifica_RANGE(od1977, 'SUBZONA_DEST', 1, 633)
```

0.67 Passo 55: "MUN_DEST"

Checar se existe algum erro

Categorias

1 a 27

[Teste: Checar se existe algum número < 1 ou > 27. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "MUN_DEST < 1" ou de "MUN_DEST > 27"
        #The 'error' returns must be related to "MUN_DEST" == 0, that is, trips that were not made
        #od1977[(od1977['MUN_DEST']<1) | (od1977['MUN_DEST']>27)]
        verifica_RANGE(od1977, 'MUN_DEST', 1, 27)
```

0.68 Passo 56: "CO_DEST_X"

Na coluna "CO_DEST_X", linha i, ler o valor da linha i da coluna "SUBZONA_DEST", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"

```
In [ ]: #Getting from the csv file the "CO_X" code correspondent to the "SUBZONA_DEST" code
        od1977['CO_DEST_X'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', 'S
```

0.69 Passo 57: "CO_DEST_Y"

Na coluna "CO_DEST_Y", linha i, ler o valor da linha i da coluna "SUBZONA_DEST", daí, buscar o mesmo valor na coluna "SUBZONA" do arquivo COORD-SUBZONA-1977.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"

```
In [ ]: #Getting from the csv file the "CO_Y" code correspondent to the "SUBZONA_DEST" code
        od1977['CO_DEST_Y'] = od1977.apply(lambda row: consulta_refext(row, 'COORD-SUBZONA-1977.csv', 'S
```

0.70 Passo 58: "DIST_VIAG"

Calcula-se a distância euclidiana (a partir da CO_ORIG_X;CO_ORIG_Y e CO_DEST_X;CO_DEST_Y)

Falta um teste de verificação!!!

```
In [ ]: #Calculating "DIST_VIAG" (euclidian distance) from the origin cordinates (CO_ORIG_X;CO_ORIG_Y)a
        od1977['DIST_VIAG'] = od1977.apply(lambda row: calcula_DIST_VIAG(row), axis=1)
```

0.71 Passo 59: "MOTIVO_ORIG"

- Substituir todos valores **6** por **11**
- Substituir todos valores **7** por **6**
- Substituir todos valores **8** por **7**
- Substituir todos valores **10** por **8**
- Substituir todos valores **11** por **9**

Valor	Descrição
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Escola/Educação
5	Compras
6	Negócios
7	Médico/Dentista/Saúde
8	Recreação/Visitas
9	Servir Passageiro
10	Residência

Categorias anteriores

Valor	Descrição
0	Não respondeu/não fez viagem
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Educação
5	Compras
6	Saúde
7	Lazer
8	Residência
9	Outros

Categorias novas [Teste: Checar se existe algum número < 0 ou > 9. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting "MOTIVO_ORIG" in order to compare the values before and after the replacement
            display(od1977['MOTIVO_ORIG'].value_counts())
```

```
In [ ]: #Replacing the values 6 for 11
        od1977.loc[od1977['MOTIVO_ORIG']==6,'MOTIVO_ORIG'] = 11
        #Replacing the values 7 for 6
        od1977.loc[od1977['MOTIVO_ORIG']==7,'MOTIVO_ORIG'] = 6
        #Replacing the values 8 for 7
        od1977.loc[od1977['MOTIVO_ORIG']==8,'MOTIVO_ORIG'] = 7
        #Replacing the values 10 for 8
        od1977.loc[od1977['MOTIVO_ORIG']==10,'MOTIVO_ORIG'] = 8
        #Replacing the values 11 for 9
        od1977.loc[od1977['MOTIVO_ORIG']==11,'MOTIVO_ORIG'] = 9
```

```
In [ ]: if not impressao:
            #Counting "MOTIVO_ORIG" in order to compare the values before and after the replacement
            display(od1977['MOTIVO_ORIG'].value_counts())

In [ ]: #Verifying value interval for check - conditions: "MOTIVO_ORIG < 0" and "MOTIVO_ORIG > 9"
        #od1977[(od1977['MOTIVO_ORIG']<0) | (od1977['MOTIVO_ORIG']>9)]
        verifica_RANGE(od1977, 'MOTIVO_ORIG', 0, 9)
```

0.72 Passo 60: "MOTIVO_DEST"

- Substituir todos valores **6** por **11**
- Substituir todos valores **7** por **6**
- Substituir todos valores **8** por **7**
- Substituir todos valores **10** por **8**
- Substituir todos valores **11** por **9**

Valor	Descrição
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Escola/Educação
5	Compras
6	Negócios
7	Médico/Dentista/Saúde
8	Recreação/Visitas
9	Servir Passageiro
10	Residência

Categorias anteriores

Valor	Descrição
0	Não respondeu/não fez viagem
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Educação
5	Compras
6	Saúde
7	Lazer
8	Residência

Valor	Descrição
9	Outros

Categorias novas [Teste: Checar se existe algum número < 0 ou > 9. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting "MOTIVO_DEST in order to compare the values before and after the replacement
            display(od1977['MOTIVO_DEST'].value_counts())
```

```
In [ ]: #Replacing the values 6 for 11
        od1977.loc[od1977['MOTIVO_DEST']==6,'MOTIVO_DEST'] = 11
        #Replacing the values 7 for 6
        od1977.loc[od1977['MOTIVO_DEST']==7,'MOTIVO_DEST'] = 6
        #Replacing the values 8 for 7
        od1977.loc[od1977['MOTIVO_DEST']==8,'MOTIVO_DEST'] = 7
        #Replacing the values 10 for 8
        od1977.loc[od1977['MOTIVO_DEST']==10,'MOTIVO_DEST'] = 8
        #Replacing the values 11 for 9
        od1977.loc[od1977['MOTIVO_DEST']==11,'MOTIVO_DEST'] = 9
```

```
In [ ]: if not impressao:
            #Counting "MOTIVO_DEST" in order to compare the values before and after the replacement
            display(od1977['MOTIVO_DEST'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MOTIVO_DEST < 0" and "MOTIVO_DEST > 9"
        #od1977[(od1977['MOTIVO_DEST']<0) | (od1977['MOTIVO_DEST']>9)]
        verifica_RANGE(od1977, 'MOTIVO_DEST', 0, 9)
```

0.73 Passo 61: "MODO1"

Não há o que fazer com os valores da coluna "MODO1"

Valor	Descrição
1	Ônibus trólebus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua
7	Metrô
8	Trem
9	Motocicleta
10	Bicicleta
11	A Pé
12	Outros

Categorias anteriores

Valor	Descrição
0	Não respondeu/não fez viagem
1	Ônibus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua / Van / Microônibus
7	Metrô
8	Trem
9	Moto
10	Bicicleta
11	A Pé
12	Outros

Categorias novas [Teste: Checar se existe algum número < 0 ou > 12. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting for check "MODO1"
            display(od1977['MODO1'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MODO1 < 0" and "MODO1 > 12"
        #od1977[(od1977['MODO1']<0) | (od21977['MODO1']>12)]
        verifica_RANGE(od1977, 'MODO1', 0, 12)
```

0.74 Passo 62: "MODO2"

Não há o que fazer com os valores da coluna "MODO2"

Valor	Descrição
1	Ônibus trólebus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua
7	Metrô
8	Trem

Valor	Descrição
9	Motocicleta
10	Bicicleta
11	A Pé
12	Outros

Categorias anteriores

Valor	Descrição
0	Não respondeu/não fez viagem
1	Ônibus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua / Van / Microônibus
7	Metrô
8	Trem
9	Moto
10	Bicicleta
11	A Pé
12	Outros

Categorias novas [Teste: Checar se existe algum número < 0 ou > 12. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting for check "MODO2"
            display(od1977['MODO2'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MODO2 < 0" and "MODO2 > 12"
        #od1977[(od1977['MODO2']<0) | (od1977['MODO2']>12)]
        verifica_RANGE(od1977, 'MODO2', 0, 12)
```

0.75 Passo 63: "MODO3"

Não há o que fazer com os valores da coluna "MODO3"

Valor	Descrição
1	Ônibus trólebus
2	Ônibus Escolar / Empresa

Valor	Descrição
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua
7	Metrô
8	Trem
9	Motocicleta
10	Bicicleta
11	A Pé
12	Outros

Categorias anteriores

Valor	Descrição
0	Não respondeu/não fez viagem
1	Ônibus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua / Van / Microônibus
7	Metrô
8	Trem
9	Moto
10	Bicicleta
11	A Pé
12	Outros

Categorias novas [Teste: Checar se existe algum número < 0 ou > 12. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting for check "MODO3"
            display(od1977['MODO3'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MODO3 < 0" and "MODO3 > 12"
        #od1977[(od1977['MODO3']<0) | (od1977['MODO3']>12)]
        verifica_RANGE(od1977, 'MODO3', 0, 12)
```

0.76 Passo 64: "MODO4"

Nada há que se fazer em relação à coluna "TIPO_EST_AUTO" - não há dados de 1977, coluna permanecerá vazia

```
In [ ]: if not impressao:
            #Counting for check "MODO4"
            display(od1977['MODO4'].value_counts())
```

0.77 Passo 65: "MODO_PRIN"

Não há o que fazer com os valores da coluna "MODO_PRIN"

Valor	Descrição
1	Ônibus trólebus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua
7	Metrô
8	Trem
9	Motocicleta
10	Bicicleta
11	A Pé
12	Outros

Categorias anteriores

Valor	Descrição
0	Não respondeu/não fez viagem
1	Ônibus
2	Ônibus Escolar / Empresa
3	Dirigindo Automóvel
4	Passageiro de Automóvel
5	Táxi
6	Lotação / Perua / Van / Microônibus
7	Metrô
8	Trem
9	Moto
10	Bicicleta

Valor	Descrição
11	A Pé
12	Outros

Categorias novas [Teste: Checar se existe algum número < 0 ou > 12. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting for check "MODO_PRIN"
            display(od1977['MODO_PRIN'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MODO_PRIN < 0" and "MODO_PRIN > 12"
        #od1977[(od1977['MODO_PRIN']<0) | (od1977['MODO_PRIN']>12)]
        verifica_RANGE(od1977, 'MODO_PRIN', 0, 12)
```

0.78 "TIPO_VIAG"; "H_SAIDA"; "MIN_SAIDA"; "ANDA_ORIG"; "H_CHEG"; "MIN_CHEG"; "ANDA_DEST" e "DURACAO"

Nada há que se fazer em relação aos dados das colunas "TIPO_VIAG"; "H_SAIDA"; "MIN_SAIDA"; "ANDA_ORIG"; "H_CHEG"; "MIN_CHEG"; "ANDA_DEST" e "DURACAO"

0.79 "TIPO_EST_AUTO"

Substituir valores da coluna "TIPO_EST_AUTO"

- Substituir todos valores **1** por **5**
- Substituir todos valores **2** por **2**
- Substituir todos valores **3** por **2**
- Substituir todos valores **4** por **3**
- Substituir todos valores **5** por **5**
- Substituir todos valores **6** por **4**
- Substituir todos valores **7** por **1**

Valor	Descrição
1	Zona Azul / Parqímetro
2	Estacionamento Avulso
3	Estacionamento Mensal
4	Estacionamento Próprio
5	Meio-Fio / Logradouro
6	Estacionamento Patrocinado
7	Não estacionou

Categorias anteriores

Valor	Descrição
0	Não Respondeu
1	Não Estacionou
2	Estacionamento Particular (Avulso / Mensal)
3	Estacionamento Próprio
4	Estacionamento Patrocinado
5	Rua (meio fio / zona azul / zona marrom / parquímetro)

Categorias novas [Teste: Checar se existe algum número < 0 ou > 5. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: if not impressao:
            #Counting for check "TIPO_EST_AUTO"
            display(od1977['TIPO_EST_AUTO'].value_counts())
```

```
In [ ]: #Replacing the values 1 for 5
        od1977.loc[od1977['TIPO_EST_AUTO']==1,'TIPO_EST_AUTO'] = 5
        #Replacing the values 3 for 2
        od1977.loc[od1977['TIPO_EST_AUTO']==3,'TIPO_EST_AUTO'] = 2
        #Replacing the values 4 for 3
        od1977.loc[od1977['TIPO_EST_AUTO']==4,'TIPO_EST_AUTO'] = 3
        #Replacing the values 6 for 4
        od1977.loc[od1977['TIPO_EST_AUTO']==6,'TIPO_EST_AUTO'] = 4
        #Replacing the values 7 for 1
        od1977.loc[od1977['TIPO_EST_AUTO']==7,'TIPO_EST_AUTO'] = 1
```

```
In [ ]: if not impressao:
            #Counting "TIPO_EST_AUTO in order to compare the values before and after the replacement
            display(od1977['TIPO_EST_AUTO'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "TIPO_EST_AUTO < 0" and "TIPO_EST_AUTO > 5"
        #od1977[(od1977['TIPO_EST_AUTO']<0) | (od1977['TIPO_EST_AUTO']>5)]
        verifica_RANGE(od1977, 'TIPO_EST_AUTO', 0, 5)
```

0.80 "VALOR_EST_AUTO"

Nada há que se fazer em relação à coluna "VALOR_EST_AUTO".

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```