rotina_2007

June 30, 2015

In [31]:
```
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 [32]:
```
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 arquiv
    O primeiro argumento passado é a "linha".
    O segundo argumento é o nome do arquivo csv que será consultado (indicar o nome com a exte
    O terceiro argumento é o nome da coluna no dataframe (.csv) consultado que servirá de refê
    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
        Uso:
        od2007_ex['coluna a receber o valor'] = od2007_ex.apply(lambda row: consulta_refext(ro
    """
    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 [33]:
```
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 [34]:
```
def verifica_RANGE(df, variavel, valor_menor, valor_maior):
    """
```

```
        Verifica se uma variável, do tipo número inteiro, contém algum valor menor que "valor_meno
            Uso:
            verifica_RANGE(nome_do_dataframe, 'coluna a ser verificada', 'valor_menor', 'valor_mai
        """
        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 th
        if not impressao:
            df_filtrado
```

In [35]:
```
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:
        od2007_ex['ID_DOM'] = od2007_ex.apply(lambda row: gera_ID_DOM(row), 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 [36]:
```
def gera_ID_FAM(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
        Uso:
        od2007_ex['ID_FAM'] = od2007_ex.apply(lambda row: gera_ID_FAM(row), axis=1)
    """
    id_dom = int(row['ID_DOM'])
    no_fam = int(row['NO_FAM'])
    return int(str(id_dom) + str('%02d'%(no_fam)))
```

In [37]:
```
def gera_ID_PESS(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
        Uso:
        od2007_ex['ID_PESS'] = od2007_ex.apply(lambda row: gera_ID_PESS(row), axis=1)
    """
    id_fam = int(row['ID_FAM'])
    no_pess = int(row['NO_PESS'])
    return int(str(id_fam) + str('%02d'%(no_pess)))
```

In [38]:
```
def gera_ID_VIAG(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
        Uso:
        od2007_ex['ID_VIAG'] = od2007_ex.apply(lambda row: gera_ID_VIAG(row), axis=1)
    """
    id_pess = int(row['ID_PESS'])
    no_viag = int(row['NO_VIAG'])
    return int(str(id_pess) + str('%02d'%(no_viag)))
```

In [39]:
```
#Reading csv file and store its contend in an intern dataframe
od2007 = pd.read_csv('OD_2007_v2d.csv', sep=';', decimal=',')
```

In [40]:
```
#Reading csv file and store its contend in an intern dataframe
addcol_2007 = pd.read_csv('OD_2007_addcol.csv', sep=';', decimal=',')
```

In [41]:
```
#Replacing a column from a dataframe to another
od2007['OCUP'] = addcol_2007['CD_ATIV']
```

In [42]:
```
#Renaming the column UCOD to UCOD_DOM
od2007.rename(columns={'UCOD':'UCOD_DOM'}, inplace=True)
```

In [43]:
```
#Creating the column UCOD_ESC (it will go to the end of dataframe)
od2007['UCOD_ESC']=None
```

In [44]:
```
#Creating the column UCOD_TRAB1 (it will go to the end of dataframe)
od2007['UCOD_TRAB1']=None
```

In [45]:
```
#Creating the column UCOD_TRAB2 (it will go to the end of dataframe)
od2007['UCOD_TRAB2']=None
```

In [46]:
```
#Creating the column UCOD_ORIG (it will go to the end of dataframe)
od2007['UCOD_ORIG']=None
```

In [47]:
```
#Creating the column UCOD_DEST (it will go to the end of dataframe)
od2007['UCOD_DEST']=None
```

In [48]:
```
od2007 = od2007[:5000]
```

In [49]:
```
#Reordering the columns, precisely, these that were just created (at the end of dataframe) nea
od2007 = od2007[['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 [50]:
```
#Storing the variables list in the "cols" variable
cols = od2007.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 [51]: if not impressao:
             #Describing data (whole dataframe)- count, mean, std, min and max
             display(od2007.describe())
```

```
       ANO     CD_ENTRE      DIA_SEM  UCOD_DOM     ZONA_DOM  SUBZONA_DOM  \
count    0  5000.000000  5000.000000         0  5000.000000            0
mean   NaN     5.955000     3.505800       NaN     6.196600          NaN
std    NaN     0.207325     2.022074       NaN     2.768299          NaN
min    NaN     5.000000     0.000000       NaN     1.000000          NaN
25%    NaN     6.000000     2.000000       NaN     4.000000          NaN
50%    NaN     6.000000     4.000000       NaN     6.000000          NaN
75%    NaN     6.000000     5.000000       NaN     8.000000          NaN
max    NaN     6.000000     6.000000       NaN    11.000000          NaN

       MUN_DOM       CO_DOM_X        CO_DOM_Y         ID_DOM       ...        \
count     5000    5000.000000     5000.000000    5000.000000       ...
mean        36  333194.919400  7396076.016600   63683.969000       ...
std          0    1054.619144     1254.622705   27216.197166       ...
min         36  331372.000000  7393989.000000   10001.000000       ...
25%         36  332308.000000  7394890.000000   41447.000000       ...
50%         36  332999.500000  7396106.000000   67044.000000       ...
75%         36  333780.000000  7397231.000000   81210.000000       ...
max         36  336071.000000  7398071.000000  111201.000000       ...

         TIPO_VIAG      H_SAIDA    MIN_SAIDA    ANDA_ORIG       H_CHEG  \
count  5000.000000  5000.000000  5000.000000  5000.000000  5000.000000
mean      1.944200    11.159000    15.863200     1.708400    11.415000
std       1.152281     6.536778    18.340681     3.703473     6.613919
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       1.000000     7.000000     0.000000     0.000000     7.000000
50%       2.000000    12.000000     0.000000     0.000000    12.000000
75%       3.000000    17.000000    30.000000     1.000000    17.000000
max       4.000000    23.000000    59.000000    40.000000    23.000000

         MIN_CHEG    ANDA_DEST      DURACAO  TIPO_EST_AUTO  VALOR_EST_AUTO
count  5000.00000  5000.000000  5000.000000    5000.000000     5000.000000
mean     18.36280     1.680600    24.195600       0.329800        0.961800
std      18.10171     3.592654    25.497462       1.212488       12.430287
min       0.00000     0.000000     0.000000       0.000000        0.000000
25%       0.00000     0.000000     7.000000       0.000000        0.000000
50%      15.00000     0.000000    15.000000       0.000000        0.000000
75%      30.00000     1.000000    30.000000       0.000000        0.000000
max      58.00000    40.000000   210.000000       8.000000      500.000000

[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 2007" do arquivo UCOD-2007.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 [52]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_DOM" code
         od2007['UCOD_DOM'] = od2007.apply(lambda row: consulta_refext(row, 'UCOD-2007.csv', 'Zona 2007
```

```
In [53]: if not impressao:
             #Describing data ("UCOD_DOM" column) - count, mean, std, min and max
             display(od2007['UCOD_DOM'].describe())
```

```
count    5000.000000
mean        2.559400
std         1.673748
min         1.000000
25%         1.000000
50%         1.000000
75%         4.000000
max         5.000000
Name: UCOD_DOM, dtype: float64
```

```
In [54]: if not impressao:
             #Count for check "UCOD_DOM"
             display(od2007['UCOD_DOM'].value_counts())
```

```
1    2632
4    1675
5     693
dtype: int64
```

```
In [55]: #Verifying value interval for check - conditions: "UCOD_DOM < 1" and "UCOD_DOM > 67"
         verifica_RANGE(od2007, 'UCOD_DOM', 1, 67)
         #od2007_ex[(od2007['UCOD_DOM']<1) | (od2007['UCOD_DOM']>67)]
```

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

0.3 Passo 2: "ANO"

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

```
In [56]: #Assigning value '4' to all cels of the "ANO" column
         od2007["ANO"]=4
```

```
In [57]: if not impressao:
             #Describing data ("ANO" column) - count, mean, std, min and max
             display(od2007['ANO'].describe())
```

```
count    5000
mean        4
std         0
min         4
25%         4
50%         4
75%         4
max         4
Name: ANO, dtype: float64
```

0.4 Passo 3: "CD_ENTRE"

- Substituir todos valores **5** por **0**.
- Substituir todos valores **6** por **1**.

Valor	Descricao
0	Completa sem viagem
1	Completa com Viagem

Categorias:

```
In [62]: if not impressao:
             #Counting for check "CD_ENTRE"
             display(od2007['CD_ENTRE'].value_counts())
```

```
6    4775
5     225
dtype: int64
```

```
In [63]: #Replacing the values 5 for 0
         od2007.loc[od2007['CD_ENTRE']==5,'CD_ENTRE'] = 0
         #Replacing the values 6 for 1
         od2007.loc[od2007['CD_ENTRE']==6,'CD_ENTRE'] = 1
```

```
In [64]: if not impressao:
             #Counting "CD_ENTRE" in order to compare the values before and after the replacement
             display(od2007['CD_ENTRE'].value_counts())
```

```
1    4775
0     225
dtype: int64
```

```
In [65]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od2007, 'CD_ENTRE')
```

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

0.5 Passo 4: "DIA_SEM"

Checar se existe algum erro na coluna ####Categorias: Valor|Descrição —|— 0|Não respondeu 2|Segunda-Feira 3|Terça-Feira 4|Quarta-Feira 5|Quinta-Feira 6|Sexta-Feira

[Teste: Checar se existe algum número < 2 ou > 6. Se encontrar, retornar erro indicando em qual linha.]

```
In [68]: if not impressao:
             #Counting for check "DIA_SEM"
             display(od2007['DIA_SEM'].value_counts())
```

```
6    1246
2    1078
4     886
0     705
5     549
3     536
dtype: int64
```

```
In [69]: #Verifying value interval for check - conditions: "DIA_SEM < 1" and "DIA_SEM > 67"
         verifica_RANGE(od2007, 'DIA_SEM', 2, 6)
         #od2007[(od2007['DIA_SEM']<2) | (od2007['DIA_SEM']>6)]['DIA_SEM'].value_counts()
```

```
0    705
dtype: int64
```

0.6 Passo 5: "ZONA_DOM"

Checar se existe algum erro

Categorias:

1 a 460

[Teste: Checar se existe algum número < 1 ou > 460. Se encontrar, retornar erro indicando em qual linha.]

```
In [70]: #Verifying value interval for check - conditions: "ZONA_DOM < 1" and "ZONA_DOM > 460"
         #od2007_ex[(od2007['ZONA_DOM']<1) | (od2007['ZONA_DOM']>460)]
         verifica_RANGE(od2007, 'ZONA_DOM', 1, 460)
```

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

0.7 Passo 6: "SUBZONA_DOM"

A coluna "SUBZONA_DOM" ficará vazia (missing values) em 2007 porque este ano já contém os dados de coordenadas.

Para os anos em que esse input não existe originalmente, esse campo é critério para inserção de coordenadas.

0.8 Passo 7: "MUN_DOM"

Checar se existe algum erro

Categorias

1 a 39

[Teste: Checar se existe algum número < 1 ou > 39. Se encontrar, retornar erro indicando em qual linha.]

```
In [71]: #Verifying value interval for check - conditions: "MUN_DOM < 1" and "MUN_DOM > 39"
         #od2007[(od2007['MUN_DOM']<1) | (od2007['MUN_DOM']>39)]
         verifica_RANGE(od2007, 'MUN_DOM', 1, 39)
```

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

0.9 Passo 8: "CO_DOM_X"

Em 2007 já existe a informação de "CO_DOM_X"

Para os demais anos:

[Na coluna "CO_DOM_X", linha i, ler o valor da linha i da coluna "SUBZONA_DOM", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.10 Passo 9: "CO_DOM_Y"

Em 2007 já existe a informação de "CO_DOM_Y"

Para os demais anos:

[Na coluna "CO_DOM_Y", linha i, ler o valor da linha i da coluna "SUBZONA_DOM", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

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 concateado com o "Ano". Resultado = 8 dígitos]

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

```
In [72]: #Generating "ID_DOM" from the concatenation of "ANO", "ZONA_DOM" and "NO_DOM" variables
         od2007['ID_DOM'] = od2007.apply(lambda row: gera_ID_DOM(row), 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 [73]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od2007, '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"

Substituir valores da coluna "TIPO_DOM"

- Substituir todos valores **1** por **0**.
- Substituir e todos valores **2** por **1**.
- Substituir e todos valores **3** por **1**.

Valor	Descrição
1	Particular
2	Coletivo
3	Favela

Categorias anteriores

Valor	Descrição
0	Particular
1	Coletivo

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

```
In [74]: if not impressao:
             #Counting for check "TIPO_DOM"
             display(od2007['TIPO_DOM'].value_counts())
```

```
1    4444
```

```
2     556
dtype: int64
```

```
In [75]: #Replacing the values 1 for 0
         od2007.loc[od2007['TIPO_DOM']==1,'TIPO_DOM'] = 0
         #Replacing the values 2 for 1
         od2007.loc[od2007['TIPO_DOM']==2,'TIPO_DOM'] = 1
         #Replacing the values 3 for 1
         od2007.loc[od2007['TIPO_DOM']==3,'TIPO_DOM'] = 1
```

```
In [76]: if not impressao:
             #Counting "TIPO_DOM" in order to compare the values before and after the replacement
             display(od2007['TIPO_DOM'].value_counts())
```

```
0    4444
1     556
dtype: int64
```

```
In [77]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od2007, 'TIPO_DOM')
```

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

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 [78]: #Generating "ID_FAM" from the concatenation of "ID_DOM" and "NO_FAM" variables
         od2007['ID_FAM'] = od2007.apply(lambda row: gera_ID_FAM(row), 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 [79]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od2007, '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 **4** por **3**
- Substituir todos valores **5** por **4**

Valor	Descrição
1	Alugada
2	Própria
3	Cedida
4	Outros
5	Não respondeu

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 [80]: if not impressao:
             #Counting for check "COND_MORA"
             display(od2007['COND_MORA'].value_counts())
```

```
1    2489
2    2113
```

```
3     320
4      42
5      36
dtype: int64
```

```
In [81]: #Replacing the values 4 for 3
         od2007.loc[od2007['COND_MORA']==4,'COND_MORA'] = 3
         #Replacing the values 5 for 4
         od2007.loc[od2007['COND_MORA']==5,'COND_MORA'] = 4
```

```
In [82]: if not impressao:
             #Counting "COND_MORA" in order to compare the values before and after the replacement
             display(od2007['COND_MORA'].value_counts())
```

```
1    2489
2    2113
3     362
4      36
dtype: int64
```

```
In [83]: #Verifying value interval for check - conditions: "COND_MORA < 1" and "COND_MORA > 4"
         #od2007[(od2007['COND_MORA']<1) | (od2007['COND_MORA']>4)]
         verifica_RANGE(od2007, 'COND_MORA', 1, 4)
```

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

0.20 "QT_AUTO", "QT_BICI" e QT_MOTO"

Nada há que se fazer em relação aos dados das colunas "QT_AUTO", "QT_BICI" e QT_MOTO"

0.21 Passo 16: "CD_RENFAM"

Substituir valores da coluna "CD_RENFAM"

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

Valor	Descrição
1	Renda Familiar Declarada e Maior que Zero
2	Renda Familiar Declarada como Zero
3	Renda Atribuída pelo Critério Brasil
4	Renda Atribuída pela Média da Zona

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 [84]: if not impressao:
             #Counting for check "CD_RENFAM"
             display(od2007['CD_RENFAM'].value_counts())
```

```
1    3304
3    1644
2      38
4      14
dtype: int64
```

```
In [85]: #Replacing the values 2 for 0
         od2007.loc[od2007['CD_RENFAM']==2,'CD_RENFAM'] = 0
         #Replacing the values 3 for 2
         od2007.loc[od2007['CD_RENFAM']==3,'CD_RENFAM'] = 2
         #Replacing the values 4 for 2
         od2007.loc[od2007['CD_RENFAM']==4,'CD_RENFAM'] = 2
```

```
In [86]: if not impressao:
             #Counting "CD_RENFAM" in order to compare the values before and after the replacement
             display(od2007['CD_RENFAM'].value_counts())
```

```
1    3304
2    1658
0      38
dtype: int64
```

```
In [87]: #Verifying value interval for check - conditions: "CD_RENFAM < 0" and "CD_RENFAM > 2"
         #od2007[(od2007['CD_RENFAM']<0) | (od2007['CD_RENFAM']>2)]
         verifica_RANGE(od2007, 'CD_RENFAM', 0, 2)
```

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

0.22 "REN_FAM"

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

0.23 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 [88]: #Generating "ID_PESS" from the concatenation of "ID_FAM" and "NO_PESS" variables
         od2007['ID_PESS'] = od2007.apply(lambda row: gera_ID_PESS(row), axis=1)
```

0.24 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 [89]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od2007, 'F_PESS')

Total de erros encontrados: 0
```

0.25 "FE_PESS" e "NO_PESS"

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

0.26 Passo 19: "SIT_FAM"

Substituir valores da coluna "SIT_FAM"

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

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

Valor	Descrição
5	Agregado
6	Empregado Residente
7	Parente do empregado

Categorias anteriores

Valor	Descrição
1	Pessoa Responsável
2	Cônjuge/Companheiro(a)
3	Filho(a)/Enteado(a)
4	Outro Parente / Agregado
5	Empregado Residente
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 [90]: if not impressao:
             #Counting for check "SIT_FAM"
             display(od2007['SIT_FAM'].value_counts())
```

```
1    2052
3    1397
2     907
4     476
5     131
6      33
7       4
dtype: int64
```

```
In [91]: #Replacing the values 5 for 4
         od2007.loc[od2007['SIT_FAM']==5,'SIT_FAM'] = 4
         #Replacing the values 6 for 5
         od2007.loc[od2007['SIT_FAM']==6,'SIT_FAM'] = 5
         #Replacing the values 7 for 6
         od2007.loc[od2007['SIT_FAM']==7,'SIT_FAM'] = 6
```

```
In [92]: if not impressao:
             #Counting "SIT_FAM" in order to compare the values before and after the replacement
             display(od2007['SIT_FAM'].value_counts())
```

```
1    2052
3    1397
2     907
4     607
```

```
5      33
6       4
dtype: int64
```

```
In [93]: #Verifying value interval for check - conditions: "SIT_FAM < 1" and "SIT_FAM > 6"
         #od2007[(od2007['SIT_FAM']<0) | (od2007['SIT_FAM']>6)]
         verifica_RANGE(od2007, 'SIT_FAM', 0, 6)
```

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

0.27 "IDADE"

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

0.28 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 [94]: if not impressao:
             #Counting for check "SEXO"
             display(od2007['SEXO'].value_counts())
```

```
2    2580
1    2420
dtype: int64
```

```
In [95]: #Replacing the values 2 for 0
         od2007.loc[od2007['SEXO']==2,'SEXO'] = 0
```

```
In [96]: if not impressao:
             #Counting "SEXO" in order to compare the values before and after the replacement
             display(od2007['SEXO'].value_counts())
```

```
0    2580
1    2420
dtype: int64
```

```
In [97]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od2007, 'SEXO')
```

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

0.29 Passo 21: "ESTUDA"

Substituir valores da coluna "ESTUDA"

- Substituir todos valores **1** por **0**
- Substituir todos valores **2, 3, 4, 5, 6 e 7** por 1

Categorias anteriores Valor|Descrição —-|—- 1|Não 2|Creche/Pré-Escola 3|1º Grau/Fundamental 4|2º Grau/Médio 5|Superior/Universitário 6|Outros ####Categorias novas Valor|Descrição —-|—- 0|Não estuda 1|Estuda

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

```
In [98]: if not impressao:
             #Counting for check "ESTUDA"
             display(od2007['ESTUDA'].value_counts())
```

```
1    3721
3     493
5     258
2     240
4     170
6     118
dtype: int64
```

```
In [99]: #Replacing the values 1 for 0
         od2007.loc[od2007['ESTUDA']==1,'ESTUDA'] = 0
         #Replacing the values 2 for 1
         od2007.loc[od2007['ESTUDA']==2,'ESTUDA'] = 1
         #Replacing the values 3 for 1
         od2007.loc[od2007['ESTUDA']==3,'ESTUDA'] = 1
         #Replacing the values 4 for 1
         od2007.loc[od2007['ESTUDA']==4,'ESTUDA'] = 1
         #Replacing the values 5 for 1
         od2007.loc[od2007['ESTUDA']==5,'ESTUDA'] = 1
         #Replacing the values 6 for 1
         od2007.loc[od2007['ESTUDA']==6,'ESTUDA'] = 1
```

```
In [100]: if not impressao:
              #Counting "ESTUDA" in order to compare the values before and after the replacement
              display(od2007['ESTUDA'].value_counts())
```

```
0    3721
1    1279
dtype: int64
```

```
In [101]: #Verifying if there was left some value other than 0 or 1
          verifica_DUMMY(od2007, 'ESTUDA')
```

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

0.30 Passo 22: "GRAU_INSTR"

Substituir valores da coluna "GRAU_INSTR"

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

Valor	Descrição
1	Não-alfabetizado/Primário Incompleto
2	Primário Completo/Ginásio Incompleto
3	Ginásio Completo/Colegial Incompleto
4	Colegial Completo/Superior Incompleto
5	Superior Completo

Categorias anteriores:

Valor	Descrição
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 [102]: if not impressao:
              #Counting for check "GRAU_INSTR"
              display(od2007['GRAU_INSTR'].value_counts())
```

```
4    1622
2     988
3     817
1     805
5     768
dtype: int64
```

```
In [103]: #Replacing the values 2 for 1
          od2007.loc[od2007['GRAU_INSTR']==2,'GRAU_INSTR'] = 1
          #Replacing the values 3 for 2
          od2007.loc[od2007['GRAU_INSTR']==3,'GRAU_INSTR'] = 2
          #Replacing the values 4 for 3
          od2007.loc[od2007['GRAU_INSTR']==4,'GRAU_INSTR'] = 3
          #Replacing the values 5 for 4
          od2007.loc[od2007['GRAU_INSTR']==5,'GRAU_INSTR'] = 4
```

```
In [104]: if not impressao:
              #Counting "GRAU_INSTR" in order to compare the values before and after the replacement
              display(od2007['GRAU_INSTR'].value_counts())
```

```
1    1793
3    1622
2     817
4     768
dtype: int64
```

```
In [105]: #Verifying value interval for check - conditions: "GRAU_INSTR < 1" and "GRAU_INSTR > 4"
          #od2007[(od2007['GRAU_INSTR']<1) | (od2007['GRAU_INSTR']>4)]
          verifica_RANGE(od2007, 'GRAU_INSTR', 1, 4)
```

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

0.31 Passo 23: "OCUP"

Substituir valores da coluna "OCUP"

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

Valor	Descrição
1	Tem trabalho
2	Faz bico
3	Em licença médica

Valor	Descrição
4	Aposentado / pensionista
5	Sem trabalho
6	Nunca trabalhou
7	Dona de casa
8	Estudante

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 [106]: if not impressao:
              #Counting for check "OCUP"
              display(od2007['OCUP'].value_counts())
```

```
1    2980
8     652
4     475
6     283
7     237
5     216
2     136
3      21
dtype: int64
```

```
In [107]: #Replacing the values 2 for 1
          od2007.loc[od2007['OCUP']==2,'OCUP'] = 1
          #Replacing the values 3 for 2
          od2007.loc[od2007['OCUP']==3,'OCUP'] = 2
          #Replacing the values 4 for 3
          od2007.loc[od2007['OCUP']==4,'OCUP'] = 3
          #Replacing the values 5 for 4
          od2007.loc[od2007['OCUP']==5,'OCUP'] = 4
          #Replacing the values 6 for 5
          od2007.loc[od2007['OCUP']==6,'OCUP'] = 5
```

```
        #Replacing the values 7 for 6
        od2007.loc[od2007['OCUP']==7,'OCUP'] = 6
        #Replacing the values 8 for 7
        od2007.loc[od2007['OCUP']==8,'OCUP'] = 7
```

```
In [108]: if not impressao:
              #Counting "OCUP" in order to compare the values before and after the replacement
              display(od2007['OCUP'].value_counts())
```

```
1    3116
7     652
3     475
5     283
6     237
4     216
2      21
dtype: int64
```

```
In [109]: #Verifying value interval for check - conditions: "OCUP < 1" and "OCUP > 7"
          #od2007[(od2007['OCUP']<1) | (od2007['OCUP']>7)]
          verifica_RANGE(od2007, 'OCUP', 1, 7)
```

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

0.32 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-2007.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "COD_UNIF"

Categorias anteriores

ver arquivo .csv

Valor	Descrição
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 [110]: if not impressao:
              #Counting for check "SETOR_ATIV"
              display(od2007['SETOR_ATIV'].value_counts())
```

```
0     1832
14     823
4      813
12     442
3      241
9      174
10     142
11     133
13     121
7       95
8       74
2       43
5       39
6       25
1        3
dtype: int64
```

```
In [111]: #Getting from the csv file the "CD_UNIF" (unified code for activity sector) correspondent to
          od2007['SETOR_ATIV'] = od2007.apply(lambda row: consulta_refext(row, 'setor_ativ-2007.csv', '
```

```
In [112]: if not impressao:
              #Counting "SETOR_ATIV" in order to compare the values before and after the replacement
              display(od2007['SETOR_ATIV'].value_counts())
```

```
0    1832
7    1060
9     823
4     813
3     241
5     121
6      64
2      43
1       3
dtype: int64
```

0.33 Passo 25: "CD_RENIND"

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

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

Categorias anteriores

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

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

```
In [113]: #Verifying value interval for check - conditions: "CD_RENIND < 1" and "CD_RENIND > 3"
          #od2007[(od2007['CD_RENIND']<1) | (od2007['CD_RENIND']>3)]
          verifica_RANGE(od2007, 'CD_RENIND', 1, 3)
```

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

0.34 "REN_IND"

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

0.35 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 2007" do arquivo UCOD-2007.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 [114]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ESC" code
          od2007['UCOD_ESC'] = od2007.apply(lambda row: consulta_refext(row, 'UCOD-2007.csv', 'Zona 200
```

```
In [115]: if not impressao:
              #Describing data ("UCOD_ESC" column) - count, mean, std, min and max
              display(od2007['UCOD_ESC'].describe())
```

```
count    5000.000000
mean        2.083200
std         6.545598
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max        65.000000
Name: UCOD_ESC, dtype: float64
```

```
In [116]: if not impressao:
              #Count for check "UCOD_ESC"
              display(od2007['UCOD_ESC'].value_counts())
```

```
0     3695
4      380
5      226
2      133
1      125
3      111
19      50
7       46
23      36
8       25
13      21
6       15
43      15
24      13
9        9
35       8
15       8
20       8
58       7
21       6
39       6
53       6
14       6
10       5
11       5
60       4
26       4
50       4
25       4
30       4
65       4
42       3
33       2
44       2
41       2
61       2
dtype: int64
```

```
In [117]: #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 p
          #od2007[(od2007['UCOD_ESC']<1) | (od2007['UCOD_ESC']>67)]
          verifica_RANGE(od2007, 'UCOD_ESC', 1, 67)
```

```
0    3695
dtype: int64
```

0.36 Passo 27: "ZONA_ESC"

Checar se existe algum erro

Categorias:

1 a 460

[Teste: Checar se existe algum número < 1 ou > 460. Se encontrar, retornar erro indicando em qual linha.]

```
In [118]: #Verifying value interval for check - conditions: "ZONA_ESC < 1" and "ZONA_ESC > 460"
          #The 'error' returns must be related to "ZONA_ESC" == 0, that is, trips that are not school p
          #od2007[(od2007['ZONA_ESC']<1) | (od2007['ZONA_ESC']>460)]
          verifica_RANGE(od2007, 'ZONA_ESC', 1, 460)
```

```
0    3695
dtype: int64
```

0.37 Passo 28: "SUBZONA_ESC"

A coluna "SUBZONA_ESC" ficará vazia (missing values) em 2007 porque este ano já contém os dados de coordenadas.

Para os anos em que esse input não existe originalmente, esse campo é critério para inserção de coordenadas.

0.38 Passo 29: "MUN_ESC"

Checar se existe algum erro

Categorias

1 a 39

[Teste: Checar se existe algum número < 1 ou > 39. Se encontrar, retornar erro indicando em qual linha.]

```
In [119]: #Verifying value interval for check - conditions: "MUN_ESC < 1" and "MUN_ESC > 39"
          #The 'error' returns must be related to "MUN_ESC" == 0, that is, trips that are not school pu
          #od2007[(od2007['MUN_ESC']<1) | (od2007['MUN_ESC']>39)]
          verifica_RANGE(od2007, 'MUN_ESC', 1, 39)
```

```
0     3693
99       2
dtype: int64
```

0.39 Passo 30: "CO_ESC_X"

Em 2007 já existe a informação de "CO_ESC_X"

Para os demais anos:

[Na coluna "CO_ESC_X", linha i, ler o valor da linha i da coluna "SUBZONA_ESC", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.40 Passo 31: "CO_ESC_Y"

Em 2007 já existe a informação de "CO_ESC_Y"

Para os demais anos:

[Na coluna "CO_ESC_Y", linha i, ler o valor da linha i da coluna "SUBZONA_ESC", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.41 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 2007" do arquivo UCOD-2007.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 [120]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB1" code
          od2007['UCOD_TRAB1'] = od2007.apply(lambda row: consulta_refext(row, 'UCOD-2007.csv', 'Zona 2(
```

```
In [121]: if not impressao:
              #Describing data ("UCOD_TRAB1" column) - count, mean, std, min and max
              display(od2007['UCOD_TRAB1'].describe())
```

```
count    5000.00000
mean        4.61520
std         9.27465
min         0.00000
25%         0.00000
50%         1.00000
75%         4.00000
max        65.00000
Name: UCOD_TRAB1, dtype: float64
```

```
In [122]: if not impressao:
              #Count for check "UCOD_TRAB1"
              display(od2007['UCOD_TRAB1'].value_counts())
```

```
0     1844
1     1005
4      730
5      311
2      183
3      120
13     105
19      73
14      70
8       57
10      49
12      42
23      37
24      32
15      28
```

```
6       27
11      27
41      19
7       19
43      19
9       17
39      15
17      12
30      11
32      10
44       9
38       8
21       8
58       8
57       7
46       7
26       7
37       7
29       6
61       6
16       6
34       6
48       5
40       5
63       5
25       4
60       4
42       4
54       4
59       3
50       3
62       3
49       2
53       2
22       2
28       2
27       2
65       2
55       1
dtype: int64
```

```
In [123]: #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 school
          #od2007[(od2007['UCOD_TRAB1']<1) | (od2007['UCOD_TRAB1']>67)]
          verifica_RANGE(od2007, 'UCOD_TRAB1', 1, 67)
```

```
0    1844
dtype: int64
```

0.42 Passo 33: "ZONA_TRAB1"

Checar se existe algum erro

Categorias:

1 a 460

[Teste: Checar se existe algum número < 1 ou > 460. Se encontrar, retornar erro indicando em qual linha.]

```
In [124]: #Verifying value interval for check - conditions: "ZONA_TRAB1 < 1" and "ZONA_TRAB1 > 460"
          #The 'error' returns must be related to "ZONA_TRAB1"==0, that is, trips that are not school p
          #od2007[(od2007['ZONA_TRAB1']<1) | (od2007['ZONA_TRAB1']>460)]
          verifica_RANGE(od2007, 'ZONA_TRAB1', 1, 460)
```

```
0    1844
dtype: int64
```

0.43 Passo 34: "SUBZONA_TRAB1"

A coluna "SUBZONA_TRAB1" ficará vazia (missing values) em 2007 porque este ano já contém os dados de coordenadas.

Para os anos em que esse input não existe originalmente, esse campo é critério para inserção de coordenadas.

0.44 Passo 35: "MUN_TRAB1"

Checar se existe algum erro

Categorias

1 a 39

[Teste: Checar se existe algum número < 1 ou > 39. Se encontrar, retornar erro indicando em qual linha.]

```
In [125]: #Verifying value interval for check - conditions: "MUN_TRAB1 < 1" ou de "MUN_TRAB1 > 39"
          #The 'error' returns must be related to "MUN_TRAB1" == 0, that is, trips that are not school
          #od2007[(od2007['MUN_TRAB1']<1) | (od2007['MUN_TRAB1']>39)]
          verifica_RANGE(od2007, 'MUN_TRAB1', 1, 39)
```

```
0     1832
99      12
dtype: int64
```

0.45 Passo 36: "CO_TRAB1_X"

Em 2007 já existe a informação de "CO_TRAB1_X"

Para os demais anos:

[Na coluna "CO_TRAB1_X", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB1", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.46 Passo 37: "CO_TRAB1_Y"

Em 2007 já existe a informação de "CO_TRAB1_Y"

Para os demais anos:

[Na coluna "CO_TRAB1_Y", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB1", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.47 Passo 38: "UCOD_TRAB2"

Na coluna "UCOD_TRAB2", linha i, ler o valor da linha i da coluna "ZONA_TRAB1", daí, buscar o mesmo valor na coluna "Zona 2007" do arquivo UCOD-2007.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 [126]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB2" code
          od2007['UCOD_TRAB2'] = od2007.apply(lambda row: consulta_refext(row, 'UCOD-2007.csv', 'Zona 2(
```

```
In [127]: if not impressao:
              #Describing data ("UCOD_TRAB2" column) - count, mean, std, min and max
              display(od2007['UCOD_TRAB2'].describe())
```

```
count    5000.000000
mean        0.276400
std         2.330212
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max        50.000000
Name: UCOD_TRAB2, dtype: float64
```

```
In [128]: if not impressao:
              #Count for check "UCOD_TRAB2"
              display(od2007['UCOD_TRAB2'].value_counts())
```

```
0     4875
1       24
4       18
19      18
13      12
21       8
3        6
12       6
2        5
38       4
30       4
10       4
15       4
5        4
8        3
```

```
16       2
11       2
50       1
dtype: int64
```

```
In [129]: #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 school
          #od2007[(od2007['UCOD_TRAB2']<1) | (od2007['UCOD_TRAB2']>67)]
          verifica_RANGE(od2007, 'UCOD_TRAB2', 1, 67)
```

```
0    4875
dtype: int64
```

0.48 Passo 39: "ZONA_TRAB2"

Checar se existe algum erro

Categorias:

1 a 460

[Teste: Checar se existe algum número < 1 ou > 460. Se encontrar, retornar erro indicando em qual linha.]

```
In [130]: #Verifying value interval for check - conditions: "ZONA_TRAB2 < 1" and "ZONA_TRAB2 > 460"
          #The 'error' returns must be related to "ZONA_TRAB2"==0, that is, trips that are not school p
          #od2007[(od2007['ZONA_TRAB2']<1) | (od2007['ZONA_TRAB2']>460)]
          verifica_RANGE(od2007, 'ZONA_TRAB2', 1, 460)
```

```
0    4875
dtype: int64
```

0.49 Passo 40: "SUBZONA_TRAB2"

A coluna "SUBZONA_TRAB2" ficará vazia (missing values) em 2007 porque este ano já contém os dados de coordenadas.

Para os anos em que esse input não existe originalmente, esse campo é critério para inserção de coordenadas.

0.50 Passo 41: "MUN_TRAB2"

Checar se existe algum erro

Categorias

1 a 39

[Teste: Checar se existe algum número < 1 ou > 39. Se encontrar, retornar erro indicando em qual linha.]

```
In [131]: #Verifying value interval for check - conditions: "MUN_TRAB2 < 1" ou de "MUN_TRAB2 > 39"
          #The 'error' returns must be related to "MUN_TRAB2" == 0, that is, trips that are not school
          #od2007[(od2007['MUN_TRAB2']<1) | (od2007['MUN_TRAB2']>39)]
          verifica_RANGE(od2007, 'MUN_TRAB2', 1, 39)
```

```
0    4875
dtype: int64
```

0.51 Passo 42: "CO_TRAB2_X"

Em 2007 já existe a informação de "CO_TRAB2_X"

Para os demais anos:

[Na coluna "CO_TRAB2_X", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB2", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.52 Passo 43: "CO_TRAB2_Y"

Em 2007 já existe a informação de "CO_TRAB2_Y"

Para os demais anos:

[Na coluna "CO_TRAB2_Y", linha i, ler o valor da linha i da coluna "SUBZONA_TRAB2", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.53 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 [132]: #Generating "ID_VIAG" from the concatenation of "ID_PESS" and "NO_VIAG" variables
          od2007['ID_VIAG'] = od2007.apply(lambda row: gera_ID_VIAG(row), axis=1)
```

0.54 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 [133]: od2007 = od2007.drop('F_VIAG', 1)
```

```
In [134]: #Storing the variables list in the "cols" variable
          cols = od2007.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']
```

0.55 "FE_VIAG" e "NO_VIAG"

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

0.56 "TOT_VIAG"

Nada há que se fazer em relação aos dados das colunas "TOT_VIAG"

0.57 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 2007" do arquivo UCOD-2007.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 [135]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ORIG" code
          od2007['UCOD_ORIG'] = od2007.apply(lambda row: consulta_refext(row, 'UCOD-2007.csv', 'Zona 20
```

```
In [136]: if not impressao:
              #Describing data ("UCOD_ORIG" column) - count, mean, std, min and max
              display(od2007['UCOD_ORIG'].describe())
```

```
count    5000.000000
mean        4.641600
std         8.208814
min         0.000000
25%         1.000000
50%         3.000000
75%         4.000000
max        65.000000
Name: UCOD_ORIG, dtype: float64
```

```
In [137]: if not impressao:
              #Count for check "UCOD_ORIG"
              display(od2007['UCOD_ORIG'].value_counts())
```

```
1     1585
4     1212
0      705
5      584
2      157
3      131
19      71
13      65
8       54
14      31
23      31
10      30
7       29
6       25
12      25
24      22
15      18
11      17
9       17
30      13
43      13
21      12
60      10
29      10
41       9
39       9
17       8
26       7
16       6
63       6
```

```
58       6
44       5
38       5
34       4
42       4
35       4
37       4
61       4
22       4
50       4
46       4
20       3
59       3
25       3
33       3
27       3
32       2
31       2
55       2
62       2
28       2
54       2
40       2
18       2
53       2
57       2
65       2
36       1
48       1
49       1
dtype: int64
```

```
In [138]: #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 are not school
          #od2007[(od2007['UCOD_ORIG']<1) | (od2007['UCOD_ORIG']>67)]
          verifica_RANGE(od2007, 'UCOD_ORIG', 1, 67)
```

```
0    705
dtype: int64
```

0.58 Passo 47: "ZONA_ORIG"

Checar se existe algum erro

Categorias:

1 a 460

[Teste: Checar se existe algum número < 1 ou > 460. Se encontrar, retornar erro indicando em qual linha.]

```
In [139]: #Verifying value interval for check - conditions: "ZONA_ORIG < 1" and "ZONA_ORIG > 460"
          #The 'error' returns must be related to "ZONA_ORIG"==0, that is, trips that are not school pu
```

```
#od2007[(od2007['ZONA_ORIG']<1) | (od2007['ZONA_ORIG']>460)]
verifica_RANGE(od2007, 'ZONA_ORIG', 1, 460)
```

```
0    705
dtype: int64
```

0.59 Passo 48: "SUBZONA_ORIG"

A coluna "SUBZONA_ORIG" ficará vazia (missing values) em 2007 porque este ano já contém os dados de coordenadas.

Para os anos em que esse input não existe originalmente, esse campo é critério para inserção de coordenadas.

0.60 Passo 49: "MUN_ORIG"

Checar se existe algum erro

Categorias

1 a 39

[Teste: Checar se existe algum número < 1 ou > 39. Se encontrar, retornar erro indicando em qual linha.]

```
In [140]: #Verifying value interval for check - conditions: "MUN_ORIG < 1" ou de "MUN_ORIG > 39"
          #The 'error' returns must be related to "MUN_ORIG" == 0, that is, trips that are not school p
          #od2007[(od2007['MUN_ORIG']<1) | (od2007['MUN_ORIG']>39)]
          verifica_RANGE(od2007, 'MUN_ORIG', 1, 39)
```

```
0    705
dtype: int64
```

0.61 Passo 50: "CO_ORIG_X"

Em 2007 já existe a informação de "CO_ORIG_X"

Para os demais anos:

[Na coluna "CO_ORIG_X", linha i, ler o valor da linha i da coluna "SUBZONA_ORIG", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.62 Passo 51: "CO_ORIG_Y"

Em 2007 já existe a informação de "CO_ORIG_Y"

Para os demais anos:

[Na coluna "CO_ORIG_Y", linha i, ler o valor da linha i da coluna "SUBZONA_ORIG", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.63 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 2007" do arquivo UCOD-2007.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 [141]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_DEST" code
          od2007['UCOD_DEST'] = od2007.apply(lambda row: consulta_refext(row, 'UCOD-2007.csv', 'Zona 20
```

```
In [142]: if not impressao:
              #Describing data ("UCOD_DEST" column) - count, mean, std, min and max
              display(od2007['UCOD_DEST'].describe())
```

```
count    5000.000000
mean        4.736800
std         8.478626
min         0.000000
25%         1.000000
50%         3.000000
75%         4.000000
max        66.000000
Name: UCOD_DEST, dtype: float64
```

```
In [143]: if not impressao:
              #Count for check "UCOD_DEST"
              display(od2007['UCOD_DEST'].value_counts())
```

```
1     1578
4     1202
0      705
5      584
2      160
3      132
19      73
13      66
8       56
10      30
7       30
14      30
23      29
6       25
12      25
24      21
15      19
9       18
11      17
30      14
60      13
21      12
41      12
43      12
29       9
39       9
17       8
```

```
63       8
26       7
58       6
      ...
16       5
46       4
61       4
35       4
20       4
34       4
37       4
22       4
50       4
42       4
27       3
54       3
25       3
59       3
33       3
36       2
31       2
28       2
32       2
65       2
40       2
62       2
53       2
48       1
66       1
45       1
49       1
57       1
55       1
18       1
dtype: int64
```

```
In [144]: #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 are not school
          #od2007[(od2007['UCOD_DEST']<1) | (od2007['UCOD_DEST']>67)]
          verifica_RANGE(od2007, 'UCOD_DEST', 1, 67)
```

```
0    705
dtype: int64
```

0.64 Passo 53: "ZONA_DEST"

Checar se existe algum erro

Categorias:

1 a 460

[Teste: Checar se existe algum número < 1 ou > 460. Se encontrar, retornar erro indicando em qual linha.]

```
In [145]: #Verifying value interval for check - conditions: "ZONA_DEST < 1" and "ZONA_DEST > 460"
          #The 'error' returns must be related to "ZONA_DEST"==0, that is, trips that are not school pu
          #od2007[(od2007['ZONA_DEST']<1) | (od2007['ZONA_DEST']>460)]
          verifica_RANGE(od2007, 'ZONA_DEST', 1, 460)
```

```
0    705
dtype: int64
```

0.65 Passo 54: "SUBZONA_DEST"

A coluna "SUBZONA_DEST" ficará vazia (missing values) em 2007 porque este ano já contém os dados de coordenadas.

Para os anos em que esse input não existe originalmente, esse campo é critério para inserção de coordenadas.

0.66 Passo 55: "MUN_DEST"

Checar se existe algum erro

Categorias

1 a 39

[Teste: Checar se existe algum número < 1 ou > 39. Se encontrar, retornar erro indicando em qual linha.]

```
In [146]: #Verifying value interval for check - conditions: "MUN_DEST < 1" ou de "MUN_DEST > 39"
          #The 'error' returns must be related to "MUN_DEST" == 0, that is, trips that are not school p
          #od2007[(od2007['MUN_DEST']<1) | (od2007['MUN_DEST']>39)]
          verifica_RANGE(od2007, 'MUN_DEST', 1, 39)
```

```
0    705
dtype: int64
```

0.67 Passo 56: "CO_DEST_X"

Em 2007 já existe a informação de "CO_DEST_X"

Para os demais anos:

[Na coluna "CO_DEST_X", linha i, ler o valor da linha i da coluna "SUBZONA_DEST", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_X"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.68 Passo 57: "CO_DEST_Y"

Em 2007 já existe a informação de "CO_DEST_Y"

Para os demais anos:

[Na coluna "CO_DEST_Y", linha i, ler o valor da linha i da coluna "SUBZONA_DEST", daí, buscar o mesmo valor na coluna "Subzonas_XXXX" do arquivo CO-SUBZONAS-XXXX.csv. Ao achar, retornar o valor da mesma linha, só que da coluna "CO_Y"]

[Obs.: ainda preciso construir esses csv, ou seja, determinar os centroides das subzonas a partir do MapInfo]

0.69 Passo 58: "DIST_VIAG"

Em 2007 já existe a informação de "DIST_VIAG"

Nos demais anos é preciso calcular a distância euclidiana (a partir da CO_ORIG_X;CO_ORIG_Y e CO_DEST_X;CO_DEST_Y)

0.70 Passo 59: "MOTIVO_ORIG"

Substituir valores da coluna "MOTIVO_ORIG"

- Substituir todos valores **10** por **9**

Valor	Descrição
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Educação
5	Compras
6	Saúde
7	Lazer
8	Residência
9	Procurar Emprego
10	Assuntos Pessoais

Categorias anteriores

Valor	Descrição
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Educação

Valor	Descrição
5	Compras
6	Saúde
7	Lazer
8	Residência
9	Outros

Categorias novas [Teste: Checar se existe algum número < 1 ou > 9. Se encontrar, retornar erro indicando em qual linha.]

```
In [147]: if not impressao:
              #Counting for check "MOTIVO_ORIG"
              display(od2007['MOTIVO_ORIG'].value_counts())
```

```
8     1998
4      726
3      714
0      705
2      344
10     151
7      112
5      101
1       74
6       71
9        4
dtype: int64
```

```
In [148]: #Replacing the values 10 for 9
          od2007.loc[od2007['MOTIVO_ORIG']==10,'MOTIVO_ORIG'] = 9
```

```
In [149]: if not impressao:
              #Counting "MOTIVO_ORIG" in order to compare the values before and after the replacement
              display(od2007['MOTIVO_ORIG'].value_counts())
```

```
8    1998
4     726
3     714
0     705
2     344
9     155
7     112
5     101
1      74
6      71
dtype: int64
```

```
In [150]: #Verifying value interval for check - conditions: "MOTIVO_ORIG < 1" and "MOTIVO_ORIG > 9"
          #od2007[(od2007['MOTIVO_ORIG']<1) | (od2007['MOTIVO_ORIG']>9)]
          verifica_RANGE(od2007, 'MOTIVO_ORIG', 1, 9)
```

```
0    705
dtype: int64
```

0.71 Passo 60: "MOTIVO_DEST"

Substituir valores da coluna "MOTIVO_DEST"

- Substituir todos valores **10** por **9**

Valor	Descrição
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Educação
5	Compras
6	Saúde
7	Lazer
8	Residência
9	Procurar Emprego
10	Assuntos Pessoais

Categorias anteriores

Valor	Descrição
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 < 1 ou > 9. Se encontrar, retornar erro indicando em qual linha.]

```
In [151]: if not impressao:
              #Counting for check "MOTIVO_DEST"
              display(od2007['MOTIVO_DEST'].value_counts())
```

```
8     1959
4      731
3      727
0      705
2      355
10     147
7      127
5      102
1       72
6       71
9        4
dtype: int64
```

```
In [152]: #Replacing the values 10 for 9
          od2007.loc[od2007['MOTIVO_DEST']==10,'MOTIVO_DEST'] = 9
```

```
In [153]: if not impressao:
              #Counting "MOTIVO_DEST in order to compare the values before and after the replacement
              display(od2007['MOTIVO_DEST'].value_counts())
```

```
8    1959
4     731
3     727
0     705
2     355
9     151
7     127
5     102
1      72
6      71
dtype: int64
```

```
In [154]: #Verifying value interval for check - conditions: "MOTIVO_DEST < 1" and "MOTIVO_DEST > 9"
          #od2007[(od2007['MOTIVO_DEST']<1) | (od2007['MOTIVO_DEST']>9)]
          verifica_RANGE(od2007, 'MOTIVO_DEST', 1, 9)
```

```
0    705
dtype: int64
```

0.72 Passo 61: "MODO1"

Substituir valores da coluna "MODO1"

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

- Substituir todos valores **10** por **6**
- Substituir todos valores **11** por **6**
- Substituir todos valores **12** por **7**
- Substituir todos valores **13** por **8**
- Substituir todos valores **14** por **9**
- Substituir todos valores **15** por **10**
- Substituir todos valores **16** por **11**
- Substituir todos valores **17** por **12**

Valor	Descrição
1	Ônibus Município S.Paulo
2	Ônibus Outros Municípios
3	Ônibus Metropolitano
4	Ônibus Fretado
5	Escolar
6	Dirigindo Automóvel
7	Passageiro de Automóvel
8	Táxi
9	Microônibus/Van Município de S.Paulo
10	Microônibus/Van Outros Municípios
11	Microônibus/Van Metropolitano
12	Metrô
13	Trem
14	Moto
15	Bicicleta
16	A Pé
17	Outros

Categorias anteriores

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

Valor	Descrição
9	Moto
10	Bicicleta
11	A Pé
12	Outros

Categorias novas [Teste: Checar se existe algum número < 1 ou > 12. Se encontrar, retornar erro indicando em qual linha.]

```
In [155]: if not impressao:
              #Counting for check "MODO1"
              display(od2007['MODO1'].value_counts())
```

```
16    2299
0      705
1      693
12     479
6      402
7      197
15      61
5       50
14      31
13      28
8       24
4       13
9       12
2        4
3        1
10       1
dtype: int64
```

```
In [156]: #Replacing the values 2 for 1
          od2007.loc[od2007['MODO1']==2,'MODO1'] = 1
          #Replacing the values 3 for 1
          od2007.loc[od2007['MODO1']==3,'MODO1'] = 1
          #Replacing the values 4 for 2
          od2007.loc[od2007['MODO1']==4,'MODO1'] = 2
          #Replacing the values 5 for 2
          od2007.loc[od2007['MODO1']==5,'MODO1'] = 2
          #Replacing the values 6 for 3
          od2007.loc[od2007['MODO1']==6,'MODO1'] = 3
          #Replacing the values 7 for 4
          od2007.loc[od2007['MODO1']==7,'MODO1'] = 4
          #Replacing the values 8 for 5
          od2007.loc[od2007['MODO1']==8,'MODO1'] = 5
          #Replacing the values 9 for 6
          od2007.loc[od2007['MODO1']==9,'MODO1'] = 6
          #Replacing the values 10 for 6
          od2007.loc[od2007['MODO1']==10,'MODO1'] = 6
          #Replacing the values 11 for 6
```

```
od2007.loc[od2007['MODO1']==11,'MODO1'] = 6
#Replacing the values 12 for 7
od2007.loc[od2007['MODO1']==12,'MODO1'] = 7
#Replacing the values 13 for 8
od2007.loc[od2007['MODO1']==13,'MODO1'] = 8
#Replacing the values 14 for 9
od2007.loc[od2007['MODO1']==14,'MODO1'] = 9
#Replacing the values 15 for 10
od2007.loc[od2007['MODO1']==15,'MODO1'] = 10
#Replacing the values 16 for 11
od2007.loc[od2007['MODO1']==16,'MODO1'] = 11
#Replacing the values 17 for 11
od2007.loc[od2007['MODO1']==17,'MODO1'] = 12
```

```
In [157]: if not impressao:
              #Counting "MODO1 in order to compare the values before and after the replacement
              display(od2007['MODO1'].value_counts())
```

```
11    2299
0      705
1      698
7      479
3      402
4      197
2       63
10      61
9       31
8       28
5       24
6       13
dtype: int64
```

```
In [158]: #Verifying value interval for check - conditions: "MODO1 < 1" and "MODO1 > 12"
          #od2007[(od2007['MODO1']<1) | (od2007['MODO1']>12)]
          verifica_RANGE(od2007, 'MODO1', 1, 12)
```

```
0    705
dtype: int64
```

0.73 Passo 62: "MODO2"

Substituir valores da coluna "MODO2"
˜ mesmas categorias utilizadas no MODO1 ˜

```
In [159]: if not impressao:
              #Counting for check "MODO2"
              display(od2007['MODO2'].value_counts())
```

```
0     4732
1      145
12      87
13      19
9        8
4        3
6        2
```

```
2        2
11       1
7        1
dtype: int64
```

```
In [160]: #Replacing the values 2 for 1
          od2007.loc[od2007['MODO2']==2,'MODO2'] = 1
          #Replacing the values 3 for 1
          od2007.loc[od2007['MODO2']==3,'MODO2'] = 1
          #Replacing the values 4 for 2
          od2007.loc[od2007['MODO2']==4,'MODO2'] = 2
          #Replacing the values 5 for 2
          od2007.loc[od2007['MODO2']==5,'MODO2'] = 2
          #Replacing the values 6 for 3
          od2007.loc[od2007['MODO2']==6,'MODO2'] = 3
          #Replacing the values 7 for 4
          od2007.loc[od2007['MODO2']==7,'MODO2'] = 4
          #Replacing the values 8 for 5
          od2007.loc[od2007['MODO2']==8,'MODO2'] = 5
          #Replacing the values 9 for 6
          od2007.loc[od2007['MODO2']==9,'MODO2'] = 6
          #Replacing the values 10 for 6
          od2007.loc[od2007['MODO2']==10,'MODO2'] = 6
          #Replacing the values 11 for 6
          od2007.loc[od2007['MODO2']==11,'MODO2'] = 6
          #Replacing the values 12 for 7
          od2007.loc[od2007['MODO2']==12,'MODO2'] = 7
          #Replacing the values 13 for 8
          od2007.loc[od2007['MODO2']==13,'MODO2'] = 8
          #Replacing the values 14 for 9
          od2007.loc[od2007['MODO2']==14,'MODO2'] = 9
          #Replacing the values 15 for 10
          od2007.loc[od2007['MODO2']==15,'MODO2'] = 10
          #Replacing the values 16 for 11
          od2007.loc[od2007['MODO2']==16,'MODO2'] = 11
          #Replacing the values 17 for 11
          od2007.loc[od2007['MODO2']==17,'MODO2'] = 12
```

```
In [161]: if not impressao:
              #Counting "MODO2 in order to compare the values before and after the replacement
              display(od2007['MODO2'].value_counts())
```

```
0    4732
1     147
7      87
8      19
6       9
2       3
3       2
4       1
dtype: int64
```

```
In [162]: #Verifying value interval for check - conditions: "MODO2 < 1" and "MODO2 > 12"
```

```
            #od2007[(od2007['MODO2']<1) | (od2007['MODO2']>12)]
            verifica_RANGE(od2007, 'MODO2', 1, 12)
```

```
0    4732
dtype: int64
```

0.74 Passo 63: "MODO3"

Substituir valores da coluna "MODO3"
˜ mesmas categorias utilizadas no MODO1 ˜

```
In [163]: if not impressao:
                #Counting for check "MODO3"
                display(od2007['MODO3'].value_counts())
```

```
0     4976
1       16
12       3
4        2
7        1
2        1
9        1
dtype: int64
```

```
In [164]: #Replacing the values 2 for 1
          od2007.loc[od2007['MODO3']==2,'MODO3'] = 1
          #Replacing the values 3 for 1
          od2007.loc[od2007['MODO3']==3,'MODO3'] = 1
          #Replacing the values 4 for 2
          od2007.loc[od2007['MODO3']==4,'MODO3'] = 2
          #Replacing the values 5 for 2
          od2007.loc[od2007['MODO3']==5,'MODO3'] = 2
          #Replacing the values 6 for 3
          od2007.loc[od2007['MODO3']==6,'MODO3'] = 3
          #Replacing the values 7 for 4
          od2007.loc[od2007['MODO3']==7,'MODO3'] = 4
          #Replacing the values 8 for 5
          od2007.loc[od2007['MODO3']==8,'MODO3'] = 5
          #Replacing the values 9 for 6
          od2007.loc[od2007['MODO3']==9,'MODO3'] = 6
          #Replacing the values 10 for 6
          od2007.loc[od2007['MODO3']==10,'MODO3'] = 6
          #Replacing the values 11 for 6
          od2007.loc[od2007['MODO3']==11,'MODO3'] = 6
          #Replacing the values 12 for 7
          od2007.loc[od2007['MODO3']==12,'MODO3'] = 7
          #Replacing the values 13 for 8
          od2007.loc[od2007['MODO3']==13,'MODO3'] = 8
          #Replacing the values 14 for 9
          od2007.loc[od2007['MODO3']==14,'MODO3'] = 9
          #Replacing the values 15 for 10
          od2007.loc[od2007['MODO3']==15,'MODO3'] = 10
          #Replacing the values 16 for 11
          od2007.loc[od2007['MODO3']==16,'MODO3'] = 11
```

```
#Replacing the values 17 for 11
od2007.loc[od2007['MODO3']==17,'MODO3'] = 12
```

```
In [165]: if not impressao:
              #Counting "MODO3 in order to compare the values before and after the replacement
              display(od2007['MODO3'].value_counts())
```

```
0    4976
1      17
7       3
2       2
6       1
4       1
dtype: int64
```

```
In [166]: #Verifying value interval for check - conditions: "MODO3 < 1" and "MODO3 > 12"
          #od2007[(od2007['MODO3']<1) | (od2007['MODO3']>12)]
          verifica_RANGE(od2007, 'MODO3', 1, 12)
```

```
0    4976
dtype: int64
```

0.75 Passo 64: "MODO4"

Substituir valores da coluna "MODO4"
˜ mesmas categorias utilizadas no MODO1 ˜

```
In [167]: if not impressao:
              #Counting for check "MODO4"
              display(od2007['MODO4'].value_counts())
```

```
0    5000
dtype: int64
```

```
In [168]: #Replacing the values 2 for 1
          od2007.loc[od2007['MODO4']==2,'MODO4'] = 1
          #Replacing the values 3 for 1
          od2007.loc[od2007['MODO4']==3,'MODO4'] = 1
          #Replacing the values 4 for 2
          od2007.loc[od2007['MODO4']==4,'MODO4'] = 2
          #Replacing the values 5 for 2
          od2007.loc[od2007['MODO4']==5,'MODO4'] = 2
          #Replacing the values 6 for 3
          od2007.loc[od2007['MODO4']==6,'MODO4'] = 3
          #Replacing the values 7 for 4
          od2007.loc[od2007['MODO4']==7,'MODO4'] = 4
          #Replacing the values 8 for 5
          od2007.loc[od2007['MODO4']==8,'MODO4'] = 5
          #Replacing the values 9 for 6
          od2007.loc[od2007['MODO4']==9,'MODO4'] = 6
          #Replacing the values 10 for 6
          od2007.loc[od2007['MODO4']==10,'MODO4'] = 6
          #Replacing the values 11 for 6
          od2007.loc[od2007['MODO4']==11,'MODO4'] = 6
```

```
            #Replacing the values 12 for 7
            od2007.loc[od2007['MODO4']==12,'MODO4'] = 7
            #Replacing the values 13 for 8
            od2007.loc[od2007['MODO4']==13,'MODO4'] = 8
            #Replacing the values 14 for 9
            od2007.loc[od2007['MODO4']==14,'MODO4'] = 9
            #Replacing the values 15 for 10
            od2007.loc[od2007['MODO4']==15,'MODO4'] = 10
            #Replacing the values 16 for 11
            od2007.loc[od2007['MODO4']==16,'MODO4'] = 11
            #Replacing the values 17 for 11
            od2007.loc[od2007['MODO4']==17,'MODO4'] = 12
```

```
In [169]: if not impressao:
              #Counting "MODO4 in order to compare the values before and after the replacement
              display(od2007['MODO4'].value_counts())
```

```
0    5000
dtype: int64
```

```
In [170]: #Verifying value interval for check - conditions: "MODO4 < 1" and "MODO4 > 12"
          #od2007[(od2007['MODO4']<1) | (od2007['MODO4']>12)]
          verifica_RANGE(od2007, 'MODO4', 1, 12)
```

```
0    5000
dtype: int64
```

0.76 Passo 65: "MODO_PRIN"

Substituir valores da coluna "MODO_PRIN"
˜ mesmas categorias utilizadas no MODO_1 ˜

```
In [171]: if not impressao:
              #Counting for check "MODO_PRIN"
              display(od2007['MODO_PRIN'].value_counts())
```

```
16    2299
0      705
1      636
12     569
6      400
7      192
15      61
5       50
14      31
8       22
13      17
4       10
9        5
2        2
10       1
dtype: int64
```

In [172]:
```
#Replacing the values 2 for 1
od2007.loc[od2007['MODO_PRIN']==2,'MODO_PRIN'] = 1
#Replacing the values 3 for 1
od2007.loc[od2007['MODO_PRIN']==3,'MODO_PRIN'] = 1
#Replacing the values 4 for 2
od2007.loc[od2007['MODO_PRIN']==4,'MODO_PRIN'] = 2
#Replacing the values 5 for 2
od2007.loc[od2007['MODO_PRIN']==5,'MODO_PRIN'] = 2
#Replacing the values 6 for 3
od2007.loc[od2007['MODO_PRIN']==6,'MODO_PRIN'] = 3
#Replacing the values 7 for 4
od2007.loc[od2007['MODO_PRIN']==7,'MODO_PRIN'] = 4
#Replacing the values 8 for 5
od2007.loc[od2007['MODO_PRIN']==8,'MODO_PRIN'] = 5
#Replacing the values 9 for 6
od2007.loc[od2007['MODO_PRIN']==9,'MODO_PRIN'] = 6
#Replacing the values 10 for 6
od2007.loc[od2007['MODO_PRIN']==10,'MODO_PRIN'] = 6
#Replacing the values 11 for 6
od2007.loc[od2007['MODO_PRIN']==11,'MODO_PRIN'] = 6
#Replacing the values 12 for 7
od2007.loc[od2007['MODO_PRIN']==12,'MODO_PRIN'] = 7
#Replacing the values 13 for 8
od2007.loc[od2007['MODO_PRIN']==13,'MODO_PRIN'] = 8
#Replacing the values 14 for 9
od2007.loc[od2007['MODO_PRIN']==14,'MODO_PRIN'] = 9
#Replacing the values 15 for 10
od2007.loc[od2007['MODO_PRIN']==15,'MODO_PRIN'] = 10
#Replacing the values 16 for 11
od2007.loc[od2007['MODO_PRIN']==16,'MODO_PRIN'] = 11
#Replacing the values 17 for 11
od2007.loc[od2007['MODO_PRIN']==17,'MODO_PRIN'] = 12
```

In [173]:
```
if not impressao:
    #Counting "MODO_PRIN in order to compare the values before and after the replacement
    display(od2007['MODO_PRIN'].value_counts())
```

```
11    2299
0      705
1      638
7      569
3      400
4      192
10      61
2       60
9       31
5       22
8       17
6        6
dtype: int64
```

In [174]:
```
#Verifying value interval for check - conditions: "MODO_PRIN < 1" and "MODO_PRIN > 12"
#od2007[(od2007['MODO_PRIN']<1) | (od2007['MODO_PRIN']>12)]
verifica_RANGE(od2007, 'MODO_PRIN', 1, 12)
```

```
0    705
dtype: int64
```

0.77 "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"

0.78 Passo 66: "TIPO_EST_AUTO"

Substituir valores da coluna "TIPO_EST_AUTO"

- Substituir todos valores **2** por **5**
- Substituir todos valores **3** por **4**
- Substituir todos valores **4** por **3**
- Substituir todos valores **6** por **2**
- Substituir todos valores **7** por **2**
- Substituir todos valores **8** por **0**

Valor	Descrição
1	Não Estacionou
2	Zona Azul / Zona Marrom
3	Estacionamento Patrocinado
4	Estacionamento Próprio
5	Meio-Fio
6	Avulso
7	Mensal
8	Não Respondeu

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 [175]: if not impressao:
              #Counting for check "TIPO_EST_AUTO"
              display(od2007['TIPO_EST_AUTO'].value_counts())
```

```
0    4596
4     142
5      82
3      72
1      48
7      39
8      12
6       5
2       4
dtype: int64
```

```
In [176]: #Replacing the values 2 for 5
          od2007.loc[od2007['TIPO_EST_AUTO']==2,'TIPO_EST_AUTO'] = 5
          #Replacing the values 3 for 4
          od2007.loc[od2007['TIPO_EST_AUTO']==3,'TIPO_EST_AUTO'] = 4
          #Replacing the values 4 for 3
          od2007.loc[od2007['TIPO_EST_AUTO']==4,'TIPO_EST_AUTO'] = 3
          #Replacing the values 6 for 2
          od2007.loc[od2007['TIPO_EST_AUTO']==6,'TIPO_EST_AUTO'] = 2
          #Replacing the values 7 for 2
          od2007.loc[od2007['TIPO_EST_AUTO']==7,'TIPO_EST_AUTO'] = 2
          #Replacing the values 8 for 0
          od2007.loc[od2007['TIPO_EST_AUTO']==8,'TIPO_EST_AUTO'] = 0
```

```
In [177]: if not impressao:
              #Counting "TIPO_EST_AUTO in order to compare the values before and after the replacement
              display(od2007['TIPO_EST_AUTO'].value_counts())
```

```
0    4608
3     214
5      86
1      48
2      44
dtype: int64
```

```
In [178]: #Verifying value interval for check - conditions: "TIPO_EST_AUTO < 0" and "TIPO_EST_AUTO > 5"
          #od2007[(od2007['TIPO_EST_AUTO']<0) | (od2007['TIPO_EST_AUTO']>5)]
          verifica_RANGE(od2007, 'TIPO_EST_AUTO', 0, 5)
```

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

0.79 "VALOR_EST_AUTO"

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

```
In [ ]:
```