rotina_1997

June 30, 2015

In [2]:
```
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 [3]:
```
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:
        od1997['coluna a receber o valor'] = od1997.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 [4]:
```
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 [5]:
```
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 [6]:
```
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:
        od1997['ID_DOM'] = od1997.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 [7]:
```
def gera_ID_FAM(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
        Uso:
        od1997['ID_FAM'] = od1997.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 [8]:
```
def gera_ID_PESS(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
        Uso:
        od1997['ID_PESS'] = od1997.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 [9]:
```
def gera_ID_VIAG(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
        Uso:
        od1997['ID_VIAG'] = od1997.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 [10]: #Reading csv file and store its contend in an intern dataframe
         od1997 = pd.read_csv('OD_1997.csv', sep=';', decimal=',')

In [11]: #Reading csv file and store its contend in an intern dataframe
         addcol_1997 = pd.read_csv('OD_1997_addcol.csv', sep=';', decimal=',')
```

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

<ipython-input-11-80b66fc8358a> in <module>()
      1 #Reading csv file and store its contend in an intern dataframe
----> 2 addcol_1997 = pd.read_csv('OD_1997_addcol.csv', sep=';', decimal=',')


/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'OD_1997_addcol.csv' does not exist
```

```
In [ ]: #Replacing a column from a dataframe to another
        od1997['CD_ENTRE'] = addcol_1997['RESUL_DOM']

In [12]: #Renaming the column UCOD to UCOD_DOM
         od1997.rename(columns={'UCOD':'UCOD_DOM'}, inplace=True)

In [13]: #Creating the column UCOD_ESC (it will go to the end of dataframe)
         od1997['UCOD_ESC']=None

In [14]: #Creating the column UCOD_TRAB1 (it will go to the end of dataframe)
         od1997['UCOD_TRAB1']=None

In [15]: #Creating the column UCOD_TRAB2 (it will go to the end of dataframe)
         od1997['UCOD_TRAB2']=None

In [16]: #Creating the column UCOD_ORIG (it will go to the end of dataframe)
         od1997['UCOD_ORIG']=None

In [17]: #Creating the column UCOD_DEST (it will go to the end of dataframe)
         od1997['UCOD_DEST']=None

In [18]: od1997 = od1997[:5000]

In [19]: #Reordering the columns, precisely, these that were just created (at the end of dataframe) nea
         od1997 = od1997[['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]:
```
#Storing the variables list in the "cols" variable
cols = od1997.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 [21]: if not impressao:
             #Describing data (whole dataframe)- count, mean, std, min and max
             display(od1997.describe())
```

	ANO	CD_ENTRE	DIA_SEM	UCOD_DOM	ZONA_DOM	SUBZONA_DOM	\
count	0	0	5000.000000	0	5000.000000	5000.000000	
mean	NaN	NaN	4.232600	NaN	8.027800	30.841200	
std	NaN	NaN	1.552346	NaN	4.047012	16.602684	
min	NaN	NaN	2.000000	NaN	1.000000	2.000000	
25%	NaN	NaN	3.000000	NaN	5.000000	16.000000	
50%	NaN	NaN	4.000000	NaN	9.000000	33.000000	
75%	NaN	NaN	6.000000	NaN	11.000000	44.000000	
max	NaN	NaN	6.000000	NaN	14.000000	55.000000	

	MUN_DOM	CO_DOM_X	CO_DOM_Y	ID_DOM	...	\
count	5000	0	0	5000.000000	...	
mean	36	NaN	NaN	832887.448600	...	
std	0	NaN	NaN	405509.413453	...	
min	36	NaN	NaN	121002.000000	...	
25%	36	NaN	NaN	514002.000000	...	
50%	36	NaN	NaN	911017.000000	...	
75%	36	NaN	NaN	1146031.000000	...	
max	36	NaN	NaN	1426037.000000	...	

	TIPO_VIAG	H_SAIDA	MIN_SAIDA	ANDA_ORIG	H_CHEG	\
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	
mean	1.815000	11.443800	15.748800	1.992800	11.682400	
std	1.068178	6.467921	17.814252	3.663473	6.555404	
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	5.000000	0.000000	12.000000	
75%	3.000000	17.000000	30.000000	2.000000	17.000000	
max	3.000000	23.000000	58.000000	40.000000	23.000000	

	MIN_CHEG	ANDA_DEST	DURACAO	TIPO_EST_AUTO	VALOR_EST_AUTO
count	5000.000000	5000.000000	5000.000000	0	0
mean	19.471800	2.011000	22.647000	NaN	NaN
std	17.841728	3.698705	22.384347	NaN	NaN
min	0.000000	0.000000	0.000000	NaN	NaN
25%	0.000000	0.000000	6.000000	NaN	NaN
50%	15.000000	0.000000	15.000000	NaN	NaN
75%	30.000000	2.000000	30.000000	NaN	NaN
max	59.000000	40.000000	180.000000	NaN	NaN

[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 1997" do arquivo UCOD-1997.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 [22]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_DOM" code
         od1997['UCOD_DOM'] = od1997.apply(lambda row: consulta_refext(row, 'UCOD-1997.csv', 'Zona 1997
```

```
In [23]: if not impressao:
             #Describing data ("UCOD_DOM" column) - count, mean, std, min and max
             display(od1997['UCOD_DOM'].describe())
```

```
count    5000.000000
mean        2.594000
std         1.952723
min         1.000000
25%         1.000000
50%         2.000000
75%         4.000000
max         7.000000
Name: UCOD_DOM, dtype: float64
```

```
In [24]: if not impressao:
             #Count for check "UCOD_DOM"
             display(od1997['UCOD_DOM'].value_counts())
```

```
2    1936
1    1747
7     566
5     385
4     366
dtype: int64
```

```
In [1]: #Verifying value interval for check - conditions: "UCOD_DOM < 1" and "UCOD_DOM > 67"
        verifica_RANGE(od1997, 'UCOD_DOM', 1, 67)
        #od1997_ex[(od1997['UCOD_DOM']<1) | (od1997['UCOD_DOM']>67)]


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

        NameError                                 Traceback (most recent call last)

        <ipython-input-1-6be5735e706e> in <module>()
          1 #Verifying value interval for check - conditions: "UCOD_DOM < 1" and "UCOD_DOM > 67"
        ----> 2 verifica_RANGE(od1997, 'UCOD_DOM', 1, 67)
          3 #od1997_ex[(od1997['UCOD_DOM']<1) | (od1997['UCOD_DOM']>67)]


        NameError: name 'verifica_RANGE' is not defined
```

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 [26]: #Assigning value '3' to all cels of the "ANO" column
         od1997["ANO"]=3
```

```
In [27]: if not impressao:
             #Describing data ("ANO" column) - count, mean, std, min and max
             display(od1997['ANO'].describe())
```

```
count    5000
mean        3
std         0
min         3
25%         3
50%         3
75%         3
max         3
Name: ANO, dtype: float64
```

0.4 Passo 3: "CD_ENTRE"

- Substituir todos valores **6** por **0**
- Substituir todos valores **7** por **1**

Valor	Descricao
0	Completa sem viagem
1	Completa com Viagem

Categorias:

```
In [28]: if not impressao:
             #Counting for check "CD_ENTRE"
             display(od1997['CD_ENTRE'].value_counts())
```

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

```
In [29]: #Replacing the values 6 for 0
         od1997.loc[od1997['CD_ENTRE']==6,'CD_ENTRE'] = 0
         #Replacing the values 7 for 1
         od1997.loc[od1997['CD_ENTRE']==7,'CD_ENTRE'] = 1
```

```
In [30]: if not impressao:
             #Counting "CD_ENTRE" in order to compare the values before and after the replacement
             display(od1997['CD_ENTRE'].value_counts())
```

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

```
In [31]: #Verifying if there was left some value other than 0 or 1
         verifica_DUMMY(od1997, 'CD_ENTRE')
```

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

ValueError                                Traceback (most recent call last)

<ipython-input-31-be8a481685b2> in <module>()
      1 #Verifying if there was left some value other than 0 or 1
----> 2 verifica_DUMMY(od1997, 'CD_ENTRE')


<ipython-input-4-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"

Checar se existe algum erro na coluna ####Categorias: Valor|Descrição —-|—- 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 [ ]: if not impressao:
            #Counting for check "DIA_SEM"
            display(od1997['DIA_SEM'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "DIA_SEM < 2" and "DIA_SEM > 6"
        verifica_RANGE(od1997, 'DIA_SEM', 2, 6)
        #od1997[(od1997['DIA_SEM']<2) | (od1997['DIA_SEM']>6)]['DIA_SEM'].value_counts()
```

0.6 Passo 5: "ZONA_DOM"

Checar se existe algum erro

Categorias:

1 a 389

[Teste: Checar se existe algum número < 1 ou > 389. Se encontrar, retornar erro indicando em qual linha.]

```
In [ ]: #Verifying value interval for check - conditions: "ZONA_DOM < 1" and "ZONA_DOM > 389"
        #od1997[(od1997['ZONA_DOM']<1) | (od1997['ZONA_DOM']>389)]
        verifica_RANGE(od1997, 'ZONA_DOM', 1, 389)
```

0.7 Passo 6: "SUBZONA_DOM"

FAZER!!!

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 [ ]: #Verifying value interval for check - conditions: "MUN_DOM < 1" and "MUN_DOM > 39"
        #od1997[(od1997['MUN_DOM']<1) | (od1997['MUN_DOM']>39)]
        verifica_RANGE(od1997, 'MUN_DOM', 1, 39)
```

0.9 Passo 8: "CO_DOM_X"

FAZER!!!

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"

FAZER!!!

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 [ ]: #Generating "ID_DOM" from the concatenation of "ANO", "ZONA_DOM" and "NO_DOM" variables
        od1997['ID_DOM'] = od1997.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 [ ]: #Verifying if there was left some value other than 0 or 1
        verifica_DUMMY(od1997, 'F_DOM')
```

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 [ ]: if not impressao:
            #Counting for check "TIPO_DOM"
            display(od1997['TIPO_DOM'].value_counts())
```

```
In [ ]: #Replacing the values 1 for 0
        od1997.loc[od1997['TIPO_DOM']==1,'TIPO_DOM'] = 0
        #Replacing the values 2 for 1
        od1997.loc[od1997['TIPO_DOM']==2,'TIPO_DOM'] = 1
        #Replacing the values 3 for 1
        od1997.loc[od1997['TIPO_DOM']==3,'TIPO_DOM'] = 1
```

```
In [ ]: if not impressao:
            #Counting "TIPO_DOM" in order to compare the values before and after the replacement
            display(od1997['TIPO_DOM'].value_counts())
```

```
In [ ]: #Verifying if there was left some value other than 0 or 1
        verifica_DUMMY(od1997, 'TIPO_DOM')
```

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 [ ]: #Generating "ID_FAM" from the concatenation of "ID_DOM" and "NO_FAM" variables
        od1997['ID_FAM'] = od1997.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 [ ]: #Verifying if there was left some value other than 0 or 1
        verifica_DUMMY(od1997, 'F_FAM')
```

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 [ ]: if not impressao:
            #Counting for check "COND_MORA"
            display(od1997['COND_MORA'].value_counts())
```

```
In [ ]: #Replacing the values 4 for 3
        od1997.loc[od1997['COND_MORA']==4,'COND_MORA'] = 3
        #Replacing the values 5 for 4
        od1997.loc[od1997['COND_MORA']==5,'COND_MORA'] = 4
```

```
In [ ]: if not impressao:
            #Counting "COND_MORA" in order to compare the values before and after the replacement
            display(od1997['COND_MORA'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "COND_MORA < 1" and "COND_MORA > 4"
        #od1997[(od1997['COND_MORA']<1) | (od1997['COND_MORA']>4)]
        verifica_RANGE(od1997, 'COND_MORA', 1, 4)
```

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"
Lembrando que quantidade de motos e de bicicletas não foram levantadas.

Substituir por 0??

0.21 Passo 16: "CD_RENFAM"

Substituir valores da coluna "CD_RENFAM"

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

Valor	Descrição
1	Renda Familiar Completa
2	Não Tem Renda
3	Renda Familiar Incompleta

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 [ ]: #Replacing a column from a dataframe to another
        od1997['CD_RENFAM'] = addcol_1997['RENDATRI']
```

```
In [ ]: if not impressao:
            #Counting for check "CD_RENFAM"
            display(od1997['CD_RENFAM'].value_counts())
```

```
In [ ]: #Replacing the values 2 for 0
        od1997.loc[od1997['CD_RENFAM']==2,'CD_RENFAM'] = 0
        #Replacing the values 3 for 2
        od1997.loc[od1997['CD_RENFAM']==3,'CD_RENFAM'] = 2
        #Replacing the values 4 for 2
        od1997.loc[od1997['CD_RENFAM']==4,'CD_RENFAM'] = 2
```

```
In [ ]: if not impressao:
            #Counting "CD_RENFAM" in order to compare the values before and after the replacement
            display(od1997['CD_RENFAM'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "CD_RENFAM < 0" and "CD_RENFAM > 2"
        #od1997[(od1997['CD_RENFAM']<0) | (od1997['CD_RENFAM']>2)]
        verifica_RANGE(od1997, 'CD_RENFAM', 0, 2)
```

0.22 "REN_FAM"

Fazer uma correção: Substituir os valores atuais (vindos da coluna 37 "RENDA_FA") pelos valores do arquivo OD_1997_addcol.csv (valores da coluna 39 "RENDATRI") >>> Preparar o csv das colunas a serem inseridas

```
In [ ]: #Replacing a column from a dataframe to another
        od1997['REN_FAM'] = addcol_1997['RENDATRI']
```

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 [ ]: #Generating "ID_PESS" from the concatenation of "ID_FAM" and "NO_PESS" variables
        od1997['ID_PESS'] = od1997.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 [ ]: #Verifying if there was left some value other than 0 or 1
        verifica_DUMMY(od1997, 'F_PESS')
```

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"

Não é preciso substituir valores da coluna "SIT_FAM"

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

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 [ ]: if not impressao:
            #Counting for check "SIT_FAM"
            display(od1997['SIT_FAM'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "SIT_FAM < 1" and "SIT_FAM > 6"
        #od1997[(od1997['SIT_FAM']<0) | (od1997['SIT_FAM']>6)]
        verifica_RANGE(od1997, 'SIT_FAM', 0, 6)
```

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 [ ]: if not impressao:
            #Counting for check "SEXO"
            display(od1997['SEXO'].value_counts())
```

```
In [ ]: #Replacing the values 2 for 0
        od1997.loc[od1997['SEXO']==2,'SEXO'] = 0
```

```
In [ ]: if not impressao:
            #Counting "SEXO" in order to compare the values before and after the replacement
            display(od1997['SEXO'].value_counts())
```

```
In [ ]: #Verifying if there was left some value other than 0 or 1
        verifica_DUMMY(od1997, 'SEXO')
```

0.29 Passo 21: "ESTUDA"

Substituir valores da coluna "ESTUDA"

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

Categorias anteriores Valor|Descrição —-|—- 1|Não 2|Creche/Pré-Escola 3|1º Grau/2º Grau/3º Graus 4|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 [ ]: if not impressao:
            #Counting for check "ESTUDA"
            display(od1997['ESTUDA'].value_counts())
```

```
In [ ]: #Replacing the values 1 for 0
        od1997.loc[od1997['ESTUDA']==1,'ESTUDA'] = 0
        #Replacing the values 2 for 1
        od1997.loc[od1997['ESTUDA']==2,'ESTUDA'] = 1
        #Replacing the values 3 for 1
        od1997.loc[od1997['ESTUDA']==3,'ESTUDA'] = 1
        #Replacing the values 4 for 1
        od1997.loc[od1997['ESTUDA']==4,'ESTUDA'] = 1
```

```
In [ ]: if not impressao:
            #Counting "ESTUDA" in order to compare the values before and after the replacement
            display(od1997['ESTUDA'].value_counts())
```

```
In [ ]: #Verifying if there was left some value other than 0 or 1
        verifica_DUMMY(od1997, 'ESTUDA')
```

0.30 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 **2**
- Substituir todos valores **5** por **2**
- Substituir todos valores **6** por **3**
- Substituir todos valores **7** por **3**
- Substituir todos valores **8** por **4**

Valor	Descrição
1	Não-alfabetizado
2	Pré-escola
3	1º Grau incompleto
4	1º Grau completo
5	2º Grau incompleto
6	2º Grau completo
7	Superior Incompleto
8	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 [ ]: if not impressao:
            #Counting for check "GRAU_INSTR"
            display(od1997['GRAU_INSTR'].value_counts())
```

```
In [ ]: #Replacing the values 2 for 1
        od1997.loc[od1997['GRAU_INSTR']==2,'GRAU_INSTR'] = 1
        #Replacing the values 3 for 1
        od1997.loc[od1997['GRAU_INSTR']==3,'GRAU_INSTR'] = 1
        #Replacing the values 4 for 2
        od1997.loc[od1997['GRAU_INSTR']==4,'GRAU_INSTR'] = 2
        #Replacing the values 5 for 2
        od1997.loc[od1997['GRAU_INSTR']==5,'GRAU_INSTR'] = 2
        #Replacing the values 6 for 3
        od1997.loc[od1997['GRAU_INSTR']==6,'GRAU_INSTR'] = 3
        #Replacing the values 7 for 3
        od1997.loc[od1997['GRAU_INSTR']==7,'GRAU_INSTR'] = 3
        #Replacing the values 8 for 4
        od1997.loc[od1997['GRAU_INSTR']==8,'GRAU_INSTR'] = 4
```

```
In [ ]: if not impressao:
            #Counting "GRAU_INSTR" in order to compare the values before and after the replacement
            display(od1997['GRAU_INSTR'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "GRAU_INSTR < 1" and "GRAU_INSTR > 4"
        #od1997[(od1997['GRAU_INSTR']<1) | (od1997['GRAU_INSTR']>4)]
        verifica_RANGE(od1997, 'GRAU_INSTR', 1, 4)
```

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 **5** por **3**
- Substituir todos valores **6** por **5**
- Substituir todos valores **7** por **6**
- Substituir todos valores **8** por **7**

Valor	Descrição
1	Ocupado
2	Ocupado eventualmente
3	Em licença
4	Não ocupado
5	Aposentado / Pensionista
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 [ ]: #Replacing a column from a dataframe to another
        od1997['OCUP'] = addcol_1997['CD_ATIVI']
```

```
In [ ]: if not impressao:
            #Counting for check "OCUP"
            display(od1997['OCUP'].value_counts())
```

```
In [ ]: #Replacing the values 2 for 1
        od1997.loc[od1997['OCUP']==2,'OCUP'] = 1
        #Replacing the values 3 for 2
        od1997.loc[od1997['OCUP']==3,'OCUP'] = 2
        #Replacing the values 5 for 3
        od1997.loc[od1997['OCUP']==5,'OCUP'] = 3
        #Replacing the values 6 for 5
        od1997.loc[od1997['OCUP']==6,'OCUP'] = 5
        #Replacing the values 7 for 6
        od1997.loc[od1997['OCUP']==7,'OCUP'] = 6
        #Replacing the values 8 for 7
        od1997.loc[od1997['OCUP']==8,'OCUP'] = 7
```

```
In [ ]: if not impressao:
            #Counting "OCUP" in order to compare the values before and after the replacement
            display(od1997['OCUP'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "OCUP < 1" and "OCUP > 7"
        #od1997[(od1997['OCUP']<1) | (od1997['OCUP']>7)]
        verifica_RANGE(od1997, 'OCUP', 1, 7)
```

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-1997.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 [ ]: if not impressao:
            #Counting for check "SETOR_ATIV"
            display(od1997['SETOR_ATIV'].value_counts())
```

```
In [ ]: #Getting from the csv file the "CD_UNIF" (unified code for activity sector) correspondent to th
        od1997['SETOR_ATIV'] = od1997.apply(lambda row: consulta_refext(row, 'setor_ativ-1997.csv', 'CO
```

```
In [ ]: if not impressao:
            #Counting "SETOR_ATIV" in order to compare the values before and after the replacement
            display(od1997['SETOR_ATIV'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "SETOR_ATIV < 1" and "SETOR_ATIV > 10"
        #od1997[(od1997['SETOR_ATIV']<1) | (od1997['SETOR_ATIV']>10)]
        verifica_RANGE(od1997, 'SETOR_ATIV', 1, 10)
```

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 respondeu

Categorias anteriores

Valor	Descrição
1	Tem renda

Valor	Descrição
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 [ ]: #Verifying value interval for check - conditions: "CD_RENIND < 1" and "CD_RENIND > 3"
        #od1997[(od1997['CD_RENIND']<1) | (od1997['CD_RENIND']>3)]
        verifica_RANGE(od1997, 'CD_RENIND', 1, 3)
```

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 1997" do arquivo UCOD-1997.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 [ ]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ESC" code
        od1997['UCOD_ESC'] = od1997.apply(lambda row: consulta_refext(row, 'UCOD-1997.csv', 'Zona 1997'
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_ESC" column) - count, mean, std, min and max
            display(od1997['UCOD_ESC'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_ESC"
            display(od1997['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
        #od1997[(od1997['UCOD_ESC']<1) | (od1997['UCOD_ESC']>67)]
        verifica_RANGE(od1997, 'UCOD_ESC', 1, 67)
```

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 [ ]: #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 purp
        #od1997[(od1997['ZONA_ESC']<1) | (od1997['ZONA_ESC']>460)]
        verifica_RANGE(od1997, 'ZONA_ESC', 1, 460)
```

0.37 Passo 28: "SUBZONA_ESC"

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

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 [ ]: #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 purp
        #od1997[(od1997['MUN_ESC']<1) | (od1997['MUN_ESC']>39)]
        verifica_RANGE(od1997, 'MUN_ESC', 1, 39)
```

0.39 Passo 30: "CO_ESC_X"

FAZER!!!

[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"

FAZER!!!

[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 1997" do arquivo UCOD-1997.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 [ ]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB1" code
        od1997['UCOD_TRAB1'] = od1997.apply(lambda row: consulta_refext(row, 'UCOD-1997.csv', 'Zona 199
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_TRAB1" column) - count, mean, std, min and max
            display(od1997['UCOD_TRAB1'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_TRAB1"
            display(od1997['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 school p
        #od1997[(od1997['UCOD_TRAB1']<1) | (od1997['UCOD_TRAB1']>67)]
        verifica_RANGE(od1997, 'UCOD_TRAB1', 1, 67)
```

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 [ ]: #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 pur
        #od1997[(od1997['ZONA_TRAB1']<1) | (od1997['ZONA_TRAB1']>460)]
        verifica_RANGE(od1997, 'ZONA_TRAB1', 1, 460)
```

0.43 Passo 34: "SUBZONA_TRAB1"

FAZER!!!

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 [ ]: #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 pu
        #od1997[(od1997['MUN_TRAB1']<1) | (od1997['MUN_TRAB1']>39)]
        verifica_RANGE(od1997, 'MUN_TRAB1', 1, 39)
```

0.45 Passo 36: "CO_TRAB1_X"

FAZER!!!

[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"

FAZER!!!

[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 1997" do arquivo UCOD-1997.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 [ ]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_TRAB2" code
        od1997['UCOD_TRAB2'] = od1997.apply(lambda row: consulta_refext(row, 'UCOD-1997.csv', 'Zona 199
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_TRAB2" column) - count, mean, std, min and max
            display(od1997['UCOD_TRAB2'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_TRAB2"
            display(od1997['UCOD_TRAB2'].value_counts())
```

```
In [ ]: #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 p
        #od1997[(od1997['UCOD_TRAB2']<1) | (od1997['UCOD_TRAB2']>67)]
        verifica_RANGE(od1997, 'UCOD_TRAB2', 1, 67)
```

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 [ ]: #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 pur
        #od1997[(od1997['ZONA_TRAB2']<1) | (od1997['ZONA_TRAB2']>460)]
        verifica_RANGE(od1997, 'ZONA_TRAB2', 1, 460)
```

0.49 Passo 40: "SUBZONA_TRAB2"

FAZER!!!

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 [ ]: #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 pu
        #od1997[(od1997['MUN_TRAB2']<1) | (od1997['MUN_TRAB2']>39)]
        verifica_RANGE(od1997, 'MUN_TRAB2', 1, 39)
```

0.51 Passo 42: "CO_TRAB2_X"

FAZER!!!

[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"

FAZER!!!

[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 [ ]: #Generating "ID_VIAG" from the concatenation of "ID_PESS" and "NO_VIAG" variables
        od1997['ID_VIAG'] = od1997.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 [ ]: od1997 = od1997.drop('F_VIAG', 1)
```

```
In [ ]: #Storing the variables list in the "cols" variable
        cols = od1997.columns.tolist()
        if not impressao:
            #printing "cols" variable to check if the reorder operation was effective
            display(cols)
```

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 1997" do arquivo UCOD-1997.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 [ ]: #Getting from the csv file the "UCOD" code correspondent to the "ZONA_ORIG" code
        od1997['UCOD_ORIG'] = od1997.apply(lambda row: consulta_refext(row, 'UCOD-1997.csv', 'Zona 1997
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_ORIG" column) - count, mean, std, min and max
            display(od1997['UCOD_ORIG'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_ORIG"
            display(od1997['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 are not school pu
        #od1997[(od1997['UCOD_ORIG']<1) | (od1997['UCOD_ORIG']>67)]
        verifica_RANGE(od1997, 'UCOD_ORIG', 1, 67)
```

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 [ ]: #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 purp
        #od1997[(od1997['ZONA_ORIG']<1) | (od1997['ZONA_ORIG']>460)]
        verifica_RANGE(od1997, 'ZONA_ORIG', 1, 460)
```

0.59 Passo 48: "SUBZONA_ORIG"

FAZER!!!

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 [ ]: #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 pur
        #od1997[(od1997['MUN_ORIG']<1) | (od1997['MUN_ORIG']>39)]
        verifica_RANGE(od1997, 'MUN_ORIG', 1, 39)
```

0.61 Passo 50: "CO_ORIG_X"

FAZER!!!

[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"

FAZER!!!

[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 1997" do arquivo UCOD-1997.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
        od1997['UCOD_DEST'] = od1997.apply(lambda row: consulta_refext(row, 'UCOD-1997.csv', 'Zona 1997
```

```
In [ ]: if not impressao:
            #Describing data ("UCOD_DEST" column) - count, mean, std, min and max
            display(od1997['UCOD_DEST'].describe())
```

```
In [ ]: if not impressao:
            #Count for check "UCOD_DEST"
            display(od1997['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 are not school pu
        #od1997[(od1997['UCOD_DEST']<1) | (od1997['UCOD_DEST']>67)]
        verifica_RANGE(od1997, 'UCOD_DEST', 1, 67)
```

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 [ ]: #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 purp
        #od1997[(od1997['ZONA_DEST']<1) | (od1997['ZONA_DEST']>460)]
        verifica_RANGE(od1997, 'ZONA_DEST', 1, 460)
```

0.65 Passo 54: "SUBZONA_DEST"

FAZER!!!

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 [ ]: #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 pur
        #od1997[(od1997['MUN_DEST']<1) | (od1997['MUN_DEST']>39)]
        verifica_RANGE(od1997, 'MUN_DEST', 1, 39)
```

0.67 Passo 56: "CO_DEST_X"

FAZER!!!

[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"

FAZER!!!

[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"

FAZER!!!

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"

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

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

Categorias anteriores

Valor	Descrição
1	Trabalho/Indústria
2	Trabalho/Comércio
3	Trabalho/Serviços
4	Educação
5	Compras
6	Saúde

Valor	Descrição
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 [ ]: if not impressao:
            #Counting "MOTIVO_ORIG" in order to compare the values before and after the replacement
            display(od1997['MOTIVO_ORIG'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MOTIVO_ORIG < 1" and "MOTIVO_ORIG > 9"
        #od1997[(od1997['MOTIVO_ORIG']<1) | (od1997['MOTIVO_ORIG']>9)]
        verifica_RANGE(od1997, 'MOTIVO_ORIG', 1, 9)
```

0.71 Passo 60: "MOTIVO_DEST"

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

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

Categorias anteriores

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

Valor	Descrição
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 [ ]: if not impressao:
            #Counting "MOTIVO_DEST in order to compare the values before and after the replacement
            display(od1997['MOTIVO_DEST'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MOTIVO_DEST < 1" and "MOTIVO_DEST > 9"
        #od1997[(od1997['MOTIVO_DEST']<1) | (od1997['MOTIVO_DEST']>9)]
        verifica_RANGE(od1997, 'MOTIVO_DEST', 1, 9)
```

0.72 Passo 61: "MODO1"

Substituir valores da coluna "MODO1"

- 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**
- Substituir todos valores **9** por **8**
- Substituir todos valores **10** por **9**
- Substituir todos valores **11** por **10**
- Substituir todos valores **12** por **11**
- Substituir todos valores **13** por **12**

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

Valor	Descrição
9	Trem
10	Moto
11	Bicicleta
12	A Pé
13	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
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 [ ]: if not impressao:
            #Counting for check "MODO1"
            display(od1997['MODO1'].value_counts())
```

```
In [ ]: #Replacing the values 2 for 1
        od1997.loc[od1997['MODO1']==2,'MODO1'] = 1
        #Replacing the values 3 for 2
        od1997.loc[od1997['MODO1']==3,'MODO1'] = 2
        #Replacing the values 4 for 3
        od1997.loc[od1997['MODO1']==4,'MODO1'] = 3
        #Replacing the values 5 for 4
        od1997.loc[od1997['MODO1']==5,'MODO1'] = 4
        #Replacing the values 6 for 5
        od1997.loc[od1997['MODO1']==6,'MODO1'] = 5
        #Replacing the values 7 for 6
        od1997.loc[od1997['MODO1']==7,'MODO1'] = 6
        #Replacing the values 8 for 7
```

```
od1997.loc[od1997['MODO1']==8,'MODO1'] = 7
#Replacing the values 9 for 8
od1997.loc[od1997['MODO1']==9,'MODO1'] = 8
#Replacing the values 10 for 9
od1997.loc[od1997['MODO1']==10,'MODO1'] = 9
#Replacing the values 11 for 10
od1997.loc[od1997['MODO1']==11,'MODO1'] = 10
#Replacing the values 12 for 11
od1997.loc[od1997['MODO1']==12,'MODO1'] = 11
#Replacing the values 13 for 12
od1997.loc[od1997['MODO1']==13,'MODO1'] = 12
```

In [ ]:
```
if not impressao:
    #Counting "MODO1 in order to compare the values before and after the replacement
    display(od1997['MODO1'].value_counts())
```

In [ ]:
```
#Verifying value interval for check - conditions: "MODO1 < 1" and "MODO1 > 12"
#od1997[(od1997['MODO1']<1) | (od21997['MODO1']>12)]
verifica_RANGE(od1997, 'MODO1', 1, 12)
```

0.73 Passo 62: "MODO2"

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

In [ ]:
```
if not impressao:
    #Counting for check "MODO2"
    display(od1997['MODO2'].value_counts())
```

In [ ]:
```
#Replacing the values 2 for 1
od1997.loc[od1997['MODO2']==2,'MODO2'] = 1
#Replacing the values 3 for 2
od1997.loc[od1997['MODO2']==3,'MODO2'] = 2
#Replacing the values 4 for 3
od1997.loc[od1997['MODO2']==4,'MODO2'] = 3
#Replacing the values 5 for 4
od1997.loc[od1997['MODO2']==5,'MODO2'] = 4
#Replacing the values 6 for 5
od1997.loc[od1997['MODO2']==6,'MODO2'] = 5
#Replacing the values 7 for 6
od1997.loc[od1997['MODO2']==7,'MODO2'] = 6
#Replacing the values 8 for 7
od1997.loc[od1997['MODO2']==8,'MODO2'] = 7
#Replacing the values 9 for 8
od1997.loc[od1997['MODO2']==9,'MODO2'] = 8
#Replacing the values 10 for 9
od1997.loc[od1997['MODO2']==10,'MODO2'] = 9
#Replacing the values 11 for 10
od1997.loc[od1997['MODO2']==11,'MODO2'] = 10
#Replacing the values 12 for 11
od1997.loc[od1997['MODO2']==12,'MODO2'] = 11
#Replacing the values 13 for 12
od1997.loc[od1997['MODO2']==13,'MODO2'] = 12
```

In [ ]:
```
if not impressao:
    #Counting "MODO2 in order to compare the values before and after the replacement
    display(od1997['MODO2'].value_counts())
```

```
In [ ]: #Verifying value interval for check - conditions: "MODO2 < 1" and "MODO2 > 12"
        #od1997[(od1997['MODO2']<1) | (od1997['MODO2']>12)]
        verifica_RANGE(od1997, 'MODO2', 1, 12)
```

0.74 Passo 63: "MODO3"

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

```
In [32]: if not impressao:
             #Counting for check "MODO3"
             display(od1997['MODO3'].value_counts())
```

```
0    4984
1      13
8       3
dtype: int64
```

```
In [33]: #Replacing the values 2 for 1
         od1997.loc[od1997['MODO3']==2,'MODO3'] = 1
         #Replacing the values 3 for 2
         od1997.loc[od1997['MODO3']==3,'MODO3'] = 2
         #Replacing the values 4 for 3
         od1997.loc[od1997['MODO3']==4,'MODO3'] = 3
         #Replacing the values 5 for 4
         od1997.loc[od1997['MODO3']==5,'MODO3'] = 4
         #Replacing the values 6 for 5
         od1997.loc[od1997['MODO3']==6,'MODO3'] = 5
         #Replacing the values 7 for 6
         od1997.loc[od1997['MODO3']==7,'MODO3'] = 6
         #Replacing the values 8 for 7
         od1997.loc[od1997['MODO3']==8,'MODO3'] = 7
         #Replacing the values 9 for 8
         od1997.loc[od1997['MODO3']==9,'MODO3'] = 8
         #Replacing the values 10 for 9
         od1997.loc[od1997['MODO3']==10,'MODO3'] = 9
         #Replacing the values 11 for 10
         od1997.loc[od1997['MODO3']==11,'MODO3'] = 10
         #Replacing the values 12 for 11
         od1997.loc[od1997['MODO3']==12,'MODO3'] = 11
         #Replacing the values 13 for 12
         od1997.loc[od1997['MODO3']==13,'MODO3'] = 12
```

```
In [34]: if not impressao:
             #Counting "MODO3 in order to compare the values before and after the replacement
             display(od1997['MODO3'].value_counts())
```

```
0    4984
1      13
7       3
dtype: int64
```

```
In [35]: #Verifying value interval for check - conditions: "MODO3 < 1" and "MODO3 > 12"
         #od1997[(od1997['MODO3']<1) | (od1997['MODO3']>12)]
         verifica_RANGE(od1997, 'MODO3', 1, 12)
```

```
0    4984
dtype: int64
```

0.75 Passo 64: "MODO4"

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

```
In [36]: if not impressao:
             #Counting for check "MODO4"
             display(od1997['MODO4'].value_counts())
```

```
0    5000
dtype: int64
```

```
In [37]: #Replacing the values 2 for 1
         od1997.loc[od1997['MODO4']==2,'MODO4'] = 1
         #Replacing the values 3 for 2
         od1997.loc[od1997['MODO4']==3,'MODO4'] = 2
         #Replacing the values 4 for 3
         od1997.loc[od1997['MODO4']==4,'MODO4'] = 3
         #Replacing the values 5 for 4
         od1997.loc[od1997['MODO4']==5,'MODO4'] = 4
         #Replacing the values 6 for 5
         od1997.loc[od1997['MODO4']==6,'MODO4'] = 5
         #Replacing the values 7 for 6
         od1997.loc[od1997['MODO4']==7,'MODO4'] = 6
         #Replacing the values 8 for 7
         od1997.loc[od1997['MODO4']==8,'MODO4'] = 7
         #Replacing the values 9 for 8
         od1997.loc[od1997['MODO4']==9,'MODO4'] = 8
         #Replacing the values 10 for 9
         od1997.loc[od1997['MODO4']==10,'MODO4'] = 9
         #Replacing the values 11 for 10
         od1997.loc[od1997['MODO4']==11,'MODO4'] = 10
         #Replacing the values 12 for 11
         od1997.loc[od1997['MODO4']==12,'MODO4'] = 11
         #Replacing the values 13 for 12
         od1997.loc[od1997['MODO4']==13,'MODO4'] = 12
```

```
In [38]: if not impressao:
             #Counting "MODO4 in order to compare the values before and after the replacement
             display(od1997['MODO4'].value_counts())
```

```
0    5000
dtype: int64
```

```
In [39]: #Verifying value interval for check - conditions: "MODO4 < 1" and "MODO4 > 12"
         #od1997[(od1997['MODO4']<1) | (od1997['MODO4']>12)]
         verifica_RANGE(od1997, '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 [40]: if not impressao:
             #Counting for check "MODO_PRIN"
             display(od1997['MODO_PRIN'].value_counts())
```

```
12    1800
1      754
4      701
0      675
8      563
5      370
6       42
3       35
10      26
2       15
11       6
7        5
13       5
9        3
dtype: int64
```

```
In [41]: #Replacing the values 2 for 1
         od1997.loc[od1997['MODO_PRIN']==2,'MODO_PRIN'] = 1
         #Replacing the values 3 for 2
         od1997.loc[od1997['MODO_PRIN']==3,'MODO_PRIN'] = 2
         #Replacing the values 4 for 3
         od1997.loc[od1997['MODO_PRIN']==4,'MODO_PRIN'] = 3
         #Replacing the values 5 for 4
         od1997.loc[od1997['MODO_PRIN']==5,'MODO_PRIN'] = 4
         #Replacing the values 6 for 5
         od1997.loc[od1997['MODO_PRIN']==6,'MODO_PRIN'] = 5
         #Replacing the values 7 for 6
         od1997.loc[od1997['MODO_PRIN']==7,'MODO_PRIN'] = 6
         #Replacing the values 8 for 7
         od1997.loc[od1997['MODO_PRIN']==8,'MODO_PRIN'] = 7
         #Replacing the values 9 for 8
         od1997.loc[od1997['MODO_PRIN']==9,'MODO_PRIN'] = 8
         #Replacing the values 10 for 9
         od1997.loc[od1997['MODO_PRIN']==10,'MODO_PRIN'] = 9
         #Replacing the values 11 for 10
         od1997.loc[od1997['MODO_PRIN']==11,'MODO_PRIN'] = 10
         #Replacing the values 12 for 11
         od1997.loc[od1997['MODO_PRIN']==12,'MODO_PRIN'] = 11
         #Replacing the values 13 for 12
         od1997.loc[od1997['MODO_PRIN']==13,'MODO_PRIN'] = 12
```

```
In [42]: if not impressao:
             #Counting "MODO_PRIN in order to compare the values before and after the replacement
             display(od1997['MODO_PRIN'].value_counts())
```

```
11    1800
```

```
1      769
3      701
0      675
7      563
4      370
5       42
2       35
9       26
10       6
6        5
12       5
8        3
dtype: int64
```

```
In [43]: #Verifying value interval for check - conditions: "MODO_PRIN < 1" and "MODO_PRIN > 12"
         #od1997[(od1997['MODO_PRIN']<1) | (od1997['MODO_PRIN']>12)]
         verifica_RANGE(od1997, 'MODO_PRIN', 1, 12)
```

```
0    675
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 "TIPO_EST_AUTO"

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

---

0.79 "VALOR_EST_AUTO"

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

```
In [ ]:
```