Base de nascidos vivos do DataSUS

O DataSUS disponibiliza diversos arquivos de dados com relação a seus segurados, conforme a lei da transparência de informações públicas.

Essas informações podem ser obtidas pela internet,mas já deixamos o arquivo `SINASC_RO_2019.csv` já como vai ser encontrado no DataSUS. O dicionário de dados está no arquivo `estrutura_sinasc_para_CD.pdf` (o nome do arquivo tal qual no portal do DataSUS).

Nosso objetivo

Queremos deixar uma base organizada para podermos estudar a relação entre partos com risco para o bebê e algumas condições como tempo de parto, consultas de pré-natal etc.Con

Preparação da base

1 - Carregue a base 'SINASC_RO_2019.csv'. Conte o número de registros e o número de registros não duplicados da base. Dica: você aprendeu um método que remove duplicados, encadeie este método com um outro método que conta o número de linhas. **Há linhas duplicadas?**

In [7]:
```
import pandas as pd
import requests
import warnings
with warnings.catch_warnings():
    warnings.simplefilter(action='ignore', category=FutureWarning)

sinasc = pd.read_csv('SINASC_RO_2019.csv')
print(sinasc.shape)
sinasc.drop_duplicates().shape
```

Out[7]:
```
(27028, 69)
(27028, 69)
```

2 - Conte o número de valores *missing* por variável.

In [8]:
```
sinasc.isnull().sum()
```

Out[8]:
```
ORIGEM          0
CODESTAB      115
CODMUNNASC      0
LOCNASC         0
IDADEMAE        0
               ...
munResUf        0
munResLat       1
munResLon       1
munResAlt       1
munResArea      1
Length: 69, dtype: int64
```

3 - Crie uma seleção dessa base somente com as colunas que interessam. São elas:

```
['LOCNASC', 'IDADEMAE', 'ESTCIVMAE', 'ESCMAE', 'QTDFILVIVO',
    'GESTACAO', 'GRAVIDEZ', 'CONSULTAS', 'APGAR5']
```

Refaça a contagem de valores *missings*.

In [9]:
```
sinasc_reduzido = sinasc[['LOCNASC', 'IDADEMAE', 'ESTCIVMAE', 'ESCMAE', 'QTDFILVIVO'
 'GESTACAO', 'GRAVIDEZ', 'CONSULTAS', 'APGAR5']]
sinasc_reduzido.head()

sinasc_reduzido.isnull().sum()
```

Out[9]:
```
LOCNASC          0
IDADEMAE         0
ESTCIVMAE      317
ESCMAE         312
QTDFILVIVO    1573
GESTACAO      1232
GRAVIDEZ        79
CONSULTAS        0
APGAR5         103
dtype: int64
```

4 - Apgar é uma *nota* que o pediatra dá ao bebê quando nasce de acordo com algumas características associadas principalmente à respiração. Apgar 1 e Apgar 5 são as notas dadas em 1 e 5 minutos do nascimento. Apgar5 será a nossa variável de interesse principal. Então remova todos os registros com Apgar5 não preenchido. Para esta seleção, conte novamente o número de linhas e o número de *missings*.

In [10]:
```
sinasc_reduzido.dropna(subset = ['APGAR5'], inplace = True)
sinasc_reduzido.isnull().sum()
```

```
C:\Users\Artur\anaconda3\lib\site-packages\pandas\util\_decorators.py:311: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return func(*args, **kwargs)
```

Out[10]:
```
LOCNASC          0
IDADEMAE         0
ESTCIVMAE      315
ESCMAE         310
QTDFILVIVO    1566
GESTACAO      1216
GRAVIDEZ        76
CONSULTAS        0
APGAR5           0
dtype: int64
```

5 - Observe que as variáveis `['ESTCIVMAE', 'CONSULTAS']` possuem o código `9`, que significa *ignorado*. Vamos assumir que o não preenchido é o mesmo que o código `9`.

In [11]:
```
sinasc_reduzido.ESTCIVMAE.fillna(9, inplace = True)
sinasc_reduzido.ESCMAE.fillna(9, inplace = True)
sinasc_reduzido.isnull().sum()
```

```
C:\Users\Artur\anaconda3\lib\site-packages\pandas\core\generic.py:6392: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

Out[11]:
```
LOCNASC          0
IDADEMAE         0
ESTCIVMAE        0
ESCMAE           0
QTDFILVIVO    1566
GESTACAO      1216
GRAVIDEZ        76
CONSULTAS        0
APGAR5           0
dtype: int64
```

6 - Substitua os valores faltantes da quantitativa (`QTDFILVIVO`) por zero.

In [12]:
```
sinasc_reduzido.QTDFILVIVO.fillna(0, inplace = True)
sinasc_reduzido.isnull().sum()
```

Out[12]:
```
LOCNASC          0
IDADEMAE         0
ESTCIVMAE        0
ESCMAE           0
QTDFILVIVO       0
GESTACAO      1216
GRAVIDEZ        76
CONSULTAS        0
APGAR5           0
dtype: int64
```

7 - Das restantes, decida que valores te parece mais adequado (um 'não preenchido' ou um valor 'mais provável' como no item anterior) e preencha. Justifique. Lembre-se de que tratamento de dados é trabalho do cientista, e que estamos tomando decisões a todo o momento - não há necessariamente certo e errado aqui.

In [13]:
```
print(sinasc_reduzido.GESTACAO.value_counts(normalize = True)*100)
sinasc_reduzido.GESTACAO.fillna('nao preenchido', inplace = True)
sinasc_reduzido.isnull().sum()
```

```
37 a 41 semanas        87.607453
32 a 36 semanas         8.689564
42 semanas e mais       2.715003
28 a 31 semanas         0.630130
22 a 27 semanas         0.330624
Menos de 22 semanas     0.027228
Name: GESTACAO, dtype: float64
```

Out[13]:
```
LOCNASC        0
IDADEMAE       0
ESTCIVMAE      0
ESCMAE         0
QTDFILVIVO     0
GESTACAO       0
GRAVIDEZ      76
CONSULTAS      0
APGAR5         0
dtype: int64
```

Foi criado a opção 'não preenchido' para nao afetar a proporção das outras opções, o que poderia influenciar as análises que serão feitas.

In [14]:
```
print(sinasc_reduzido.GRAVIDEZ.value_counts(normalize = True)*100)
sinasc_reduzido.GRAVIDEZ.fillna('Única', inplace = True)
print(sinasc_reduzido.GRAVIDEZ.value_counts(normalize = True)*100)
sinasc_reduzido.isnull().sum()
```

```
Única              98.163805
Dupla               1.817572
Tríplice e mais     0.018623
Name: GRAVIDEZ, dtype: float64
Única              98.168988
Dupla               1.812442
Tríplice e mais     0.018570
Name: GRAVIDEZ, dtype: float64
```

Out[14]:
```
LOCNASC       0
IDADEMAE      0
ESTCIVMAE     0
ESCMAE        0
QTDFILVIVO    0
GESTACAO      0
GRAVIDEZ      0
CONSULTAS     0
APGAR5        0
dtype: int64
```

Para a variável Gravidez, foi decidido utilizar o valor 'Única' por ser a maior probabilidade de ocorrer sem prejudicar a proporção dos dados.

8 - O Apgar possui uma classificação indicando se o bebê passou por asfixia:

- Entre 8 e 10 está em uma faixa 'normal'.
- Entre 6 e 7, significa que o recém-nascido passou por 'asfixia leve'.
- Entre 4 e 5 significa 'asfixia moderada'.
- Entre 0 e 3 significa 'asfixia severa'.

Crie uma categorização dessa variável com essa codificação e calcule as frequências dessa categorização.

In [15]:
```
classifica_apgar = []
for i in sinasc_reduzido.APGAR5:
    if i >= 8:
        classifica_apgar.append('normal')
    elif (i >= 6) and (i < 8):
        classifica_apgar.append('asfixia leve')
    elif i >= 4 and i < 6:
        classifica_apgar.append('asfixia moderada')
    else:
        classifica_apgar.append('asfixia severa')

sinasc_reduzido['classificacao_a'] = classifica_apgar
sinasc_reduzido
```

```
C:\Users\Artur\AppData\Local\Temp/ipykernel_20664/382948417.py:12: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
```

```
ser_guide/indexing.html#returning-a-view-versus-a-copy
  sinasc_reduzido['classificacao_a'] = classifica_apgar
```

Out[15]:

	LOCNASC	IDADEMAE	ESTCIVMAE	ESCMAE	QTDFILVIVO	GESTACAO	GRAVIDEZ	CONSULTA
0	1	19	5.0	8 a 11 anos	0.0	37 a 41 semanas	Única	
1	1	29	2.0	8 a 11 anos	1.0	37 a 41 semanas	Única	
2	1	37	9.0	8 a 11 anos	2.0	37 a 41 semanas	Única	
3	1	30	5.0	12 anos ou mais	0.0	37 a 41 semanas	Única	
4	1	30	2.0	8 a 11 anos	1.0	37 a 41 semanas	Única	
...	...	...	...	...	...	...	...	
27023	1	32	2.0	12 anos ou mais	1.0	32 a 36 semanas	Única	
27024	1	19	1.0	8 a 11 anos	0.0	37 a 41 semanas	Única	
27025	1	24	2.0	8 a 11 anos	0.0	37 a 41 semanas	Única	
27026	1	21	2.0	8 a 11 anos	1.0	32 a 36 semanas	Única	
27027	1	18	5.0	8 a 11 anos	1.0	37 a 41 semanas	Única	

26925 rows × 10 columns

9 - Renomeie as variáveis para que fiquem no *snake case*, ou seja, em letras minúsculas, com um *underscore* entre as palávras. Dica: repare que se você não quiser criar um *dataframe* novo, você vai precisar usar a opção `inplace = True` .

In [16]:

```
sinasc_reduzido.columns

sinasc_reduzido.rename(str.lower, axis='columns', inplace = True)
alterar = {'locnasc': 'loc_nasc', 'idademae' : 'idade_mae', 'estcivmae' : 'est_civ_m
         'qtdfilvivo': 'qtd_fil_vivo', 'gestacao' : 'gestacao',
         'gravidez' : 'gravidez', 'consultas' : 'consultas', 'apgar5' : 'apgar_5',
         'classificacao_a': 'classificacao_a'}
sinasc_reduzido.rename(columns = alterar, inplace = True)
sinasc_reduzido
```

```
C:\Users\Artur\anaconda3\lib\site-packages\pandas\core\frame.py:5039: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

Out[16]:

	loc_nasc	idade_mae	est_civ_mae	esc_mae	qtd_fil_vivo	gestacao	gravidez	consultas	apga
0	1	19	5.0	8 a 11	0.0	37 a 41	Única	4	1

	loc_nasc	idade_mae	est_civ_mae	esc_mae	qtd_fil_vivo	gestacao	gravidez	consultas	apga
				anos		semanas			
1	1	29	2.0	8 a 11 anos	1.0	37 a 41 semanas	Única	4	
2	1	37	9.0	8 a 11 anos	2.0	37 a 41 semanas	Única	4	1
3	1	30	5.0	12 anos ou mais	0.0	37 a 41 semanas	Única	3	1
4	1	30	2.0	8 a 11 anos	1.0	37 a 41 semanas	Única	4	1
...	...	...	...	...	...	...	...	...	
27023	1	32	2.0	12 anos ou mais	1.0	32 a 36 semanas	Única	4	
27024	1	19	1.0	8 a 11 anos	0.0	37 a 41 semanas	Única	4	
27025	1	24	2.0	8 a 11 anos	0.0	37 a 41 semanas	Única	4	1
27026	1	21	2.0	8 a 11 anos	1.0	32 a 36 semanas	Única	4	
27027	1	18	5.0	8 a 11 anos	1.0	37 a 41 semanas	Única	4	

26925 rows × 10 columns

Projeto

Marcação de bom e mau

O objetivo da modelagem é classificar o risco de inadimplência, ou como se diz no meio, o risco de *default*. Podemos fazer longas discussões sobre o conceito de *default* com base em estudos e exigências regulatórias, para efeitos deste estudo, um cliente em *default* é aquele que está em 60 dias de atraso ou mais. Então classificaremos os clientes como 'bons' e 'maus' assim:

- **Maus** pagadores: são aqueles que entraram em 'default' (atraso 60 dias ou mais) nos 24 meses seguintes à aquisição do cartão de crédito.
- **Bons** pagadores: são considerados todos os demais.
- **Excluídos**: Clientes que não adquiriram um cartão de crédito (seja por recusa, seja por desistência) não possuem informações de pagamento, portanto não se pode identificar se são bons ou maus. Há uma longa discussão e literatura sobre *inferência de rejeitados* que está fora do escopo deste exercício.

Bases disponíveis

Temos duas bases importantes aqui: uma de propostas, com diversas informações dos vários solicitantes de cartão de crédito, e uma base de pagamentos. A base de pagamentos será utilizada para identificar a ocorrência de *default*. A base de propostas tem diversas informações coletadas no momento da solicitação do crédito (isto é importante: qualquer informação

posterior a essa data é impossível de ser coletada na aplicação do modelo e não pode ser utilizada).

As variáveis delas são:

Base de propostas - application_records.csv

Nome da Variável	Description	Tipo
ID	identificador do cliente (chave)	inteiro
CODE_GENDER	M = 'Masculino'; F = 'Feminino'	M/F
FLAG_OWN_CAR	Y = 'possui'; N = 'não possui'	Y/N
FLAG_OWN_REALTY	Y = 'possui'; N = 'não possui'	Y/N
CNT_CHILDREN	Quantidade de filhos	inteiro
AMT_INCOME_TOTAL	Annual income	inteiro
NAME_INCOME_TYPE	Tipo de renda (ex: assaliariado, autônomo etc)	texto
NAME_EDUCATION_TYPE	Nível de educação (ex: secundário, superior etc)	texto
NAME_FAMILY_STATUS	Estado civil (ex: solteiro, casado etc)	texto
NAME_HOUSING_TYPE	tipo de residência (ex: casa/apartamento, com os pais etc)	texto
DAYS_BIRTH	Count backwards from current day (0), -1 means yesterday	inteiro
DAYS_EMPLOYED	Count backwards from current day (0), -1 means yesterday	inteiro
FLAG_MOBIL	Indica se possui celular (1 = sim, 0 = não)	binária
FLAG_WORK_PHONE	Indica se possui telefone comercial (1 = sim, 0 = não)	binária
FLAG_PHONE	Indica se possui telefone (1 = sim, 0 = não)	binária
FLAG_EMAIL	Indica se possui e-mail (1 = sim, 0 = não)	binária
OCCUPATION_TYPE	Occupation	Qualitativa
CNT_FAM_MEMBERS	quantidade de pessoas na residência	inteiro

Base de pagamentos - pagamentos_largo.csv

Nome da Variável	Description	Tipo
ID	identificador do cliente (chave)	inteiro
mes_00 a mes_24	faixa de atraso mês a mês do cliente 0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: more than 150 days C: paid off that month X: No loan for the month	Qualitativa

Construindo a variável resposta

A base de pagamentos está em um formato de 'base larga'. Essa base possui informações de pagamentos do cliente mês a mês a partir do mês de aquisição do crédito (mês 0) até o vigésimo quarto mês após a aquisição do crédito (mês 24). Utilizaremos essa base para determinar se um proponente é considerado 'bom pagador' ou caso apresente atraso representativo, será considerado 'mau pagador'.

Base larga vs base longa

A base ser larga significa que há uma linha para cada cliente, e que as informações estarão nas colunas, em contraste com a 'base longa', em que haveria uma linha para cada combinação cliente/mês, uma coluna indicando o cliente, outra indicando o mês, e apenas uma coluna com a informação do atraso.

In [17]:
```
import pandas as pd
import numpy as np
```

In [18]:
```
propostas = pd.read_csv('application_record.csv')
pg = pd.read_csv('pagamentos_largo.csv')
```

In [19]:
```
pg.head()
```

Out[19]:

	ID	mes_0	mes_1	mes_10	mes_11	mes_12	mes_13	mes_14	mes_15	mes_16	...	mes_22
0	5001718	0	0	0	0	0	0	0	0	NaN	...	NaN
1	5001719	0	0	C	C	C	C	C	C	C	...	C
2	5001720	0	0	0	0	0	0	0	0	0	...	1
3	5001723	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
4	5001726	0	0	C	C	C	C	C	C	C	...	C

5 rows × 26 columns

Tarefa 1) Marcar *default* no mês

Faça uma indicadora de se o cliente está em *default* em cada uma das marcações (mes_00 a mes_24). Dica: você pode utilizar o método `.isin()` do Pandas. Consulte a documentação caso necessário.

In [20]:
```
pg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20937 entries, 0 to 20936
Data columns (total 26 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   ID      20937 non-null  int64 
 1   mes_0   20937 non-null  object
 2   mes_1   19216 non-null  object
 3   mes_10  17455 non-null  object
 4   mes_11  16972 non-null  object
 5   mes_12  16943 non-null  object
 6   mes_13  16540 non-null  object
 7   mes_14  16525 non-null  object
 8   mes_15  16155 non-null  object
 9   mes_16  16093 non-null  object
 10  mes_17  15911 non-null  object
 11  mes_18  15630 non-null  object
 12  mes_19  15336 non-null  object
 13  mes_2   19348 non-null  object
```

```
 14  mes_20  15184 non-null  object
 15  mes_21  15076 non-null  object
 16  mes_22  14797 non-null  object
 17  mes_23  14598 non-null  object
 18  mes_24  14433 non-null  object
 19  mes_3   18925 non-null  object
 20  mes_4   18802 non-null  object
 21  mes_5   18512 non-null  object
 22  mes_6   18214 non-null  object
 23  mes_7   17762 non-null  object
 24  mes_8   17741 non-null  object
 25  mes_9   17510 non-null  object
dtypes: int64(1), object(25)
memory usage: 4.2+ MB
```

In [21]:
```
pg.shape
```

Out[21]:
```
(20937, 26)
```

In [22]:
```
default = pg.isin(['2','3', '4', '5'])
default
```

Out[22]:

	ID	mes_0	mes_1	mes_10	mes_11	mes_12	mes_13	mes_14	mes_15	mes_16	...	mes_2
0	False	False	False	False	False	False	False	False	False	False	...	Fal
1	False	False	False	False	False	False	False	False	False	False	...	Fal
2	False	False	False	False	False	False	False	False	False	False	...	Fal
3	False	False	False	False	False	False	False	False	False	False	...	Fal
4	False	False	False	False	False	False	False	False	False	False	...	Fal
...	...	...	...	...	...	...	...	...	...	...	...	
20932	False	False	False	False	False	False	False	False	False	False	...	Fal
20933	False	False	False	False	False	False	False	False	False	False	...	Fal
20934	False	False	False	False	False	False	False	False	False	False	...	Fal
20935	False	False	False	False	False	False	False	False	False	False	...	Fal
20936	False	False	False	False	False	False	False	False	False	False	...	Fal

20937 rows × 26 columns

Tarefa 2) 'bons' e 'maus' ao longo de todos os 24 meses de desempenho

Marque para cada cliente se ele teve pelo menos um episódio de *default* entre o mês 0 e o mês 24. Dica: o método `sum()` pode ajudar. Caso precise, consulte a documentação e procure pelo argumento `axis`, você viu outros métodos que possuem esse argumento também. Tendo o número de meses em default de cada cliente, basta marcar `True` para todos aqueles que possuem pelo menos 1 mês em *default* e `False` para os demais.

In [23]:
```
default['soma_default'] = default.sum(axis = 1)
default.loc[default['soma_default'] == 0, 'default'] = 'bom'
default.loc[default['soma_default'] > 0, 'default'] = 'mau'
```

default

Out[23]:

	ID	mes_0	mes_1	mes_10	mes_11	mes_12	mes_13	mes_14	mes_15	mes_16	...	mes_2
0	False	False	False	False	False	False	False	False	False	False	...	Fal
1	False	False	False	False	False	False	False	False	False	False	...	Fal
2	False	False	False	False	False	False	False	False	False	False	...	Fal
3	False	False	False	False	False	False	False	False	False	False	...	Fal
4	False	False	False	False	False	False	False	False	False	False	...	Fal
...	...	...	...	...	...	...	...	...	...	...	...	
20932	False	False	False	False	False	False	False	False	False	False	...	Fal
20933	False	False	False	False	False	False	False	False	False	False	...	Fal
20934	False	False	False	False	False	False	False	False	False	False	...	Fal
20935	False	False	False	False	False	False	False	False	False	False	...	Fal
20936	False	False	False	False	False	False	False	False	False	False	...	Fal

20937 rows × 28 columns

Tarefa 3) Marcando proponentes expostos ao risco de crédito

Marcando proponentes que se tornaram tomadores: lembre-se de que clientes que não adquiriram o cartão devem ser desconsiderados. A base de pagamentos possui apenas clientes que adquiriram cartão de crédito, então você pode selecionar somente os clientes da base de propostas que se encontram na base de pagamentos.

In [24]:
```
propostas.head()
```

Out[24]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TO
0	5008804	M	Y	Y	0	4275(
1	5008805	M	Y	Y	0	4275(
2	5008806	M	Y	Y	0	1125(
3	5008808	F	N	Y	0	2700(
4	5008809	F	N	Y	0	2700(

In [25]:
```
df = pg.merge(propostas, on = 'ID', how = 'left')
df
```

Out[25]:

	ID	mes_0	mes_1	mes_10	mes_11	mes_12	mes_13	mes_14	mes_15	mes_16	...	N
0	5001718	0	0	0	0	0	0	0	0	NaN	...	
1	5001719	0	0	C	C	C	C	C	C	C	...	

	ID	mes_0	mes_1	mes_10	mes_11	mes_12	mes_13	mes_14	mes_15	mes_16	...	N
2	5001720	0	0	0	0	0	0	0	0	0	...	
3	5001723	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
4	5001726	0	0	C	C	C	C	C	C	C	...	
...	...	...	...	...	...	...	...	...	...	...	...	
20932	5150475	C	C	C	C	C	C	C	C	C	...	
20933	5150476	0	0	0	0	0	0	0	0	0	...	
20934	5150480	0	0	C	C	C	C	C	C	C	...	
20935	5150482	0	0	0	0	C	C	C	C	C	...	
20936	5150487	C	C	C	C	C	C	C	C	C	...	

20937 rows × 43 columns

Tarefa 4) Consolidando as informações

Faça uma junção das informações da base de propostas com a variável de *default* que você acabou de construir. Talvez você consiga realizar a tarefa 3 e tarefa 4 em uma única linha de código ;)

In [26]:
```
df['default'] = default['default']
df
```

Out[26]:

	ID	mes_0	mes_1	mes_10	mes_11	mes_12	mes_13	mes_14	mes_15	mes_16	...	N
0	5001718	0	0	0	0	0	0	0	0	NaN	...	
1	5001719	0	0	C	C	C	C	C	C	C	...	
2	5001720	0	0	0	0	0	0	0	0	0	...	
3	5001723	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	
4	5001726	0	0	C	C	C	C	C	C	C	...	
...	...	...	...	...	...	...	...	...	...	...	...	
20932	5150475	C	C	C	C	C	C	C	C	C	...	
20933	5150476	0	0	0	0	0	0	0	0	0	...	
20934	5150480	0	0	C	C	C	C	C	C	C	...	
20935	5150482	0	0	0	0	C	C	C	C	C	...	
20936	5150487	C	C	C	C	C	C	C	C	C	...	

20937 rows × 44 columns

Tarefa 5) Verificando

Faça uma contagem dos valores do *default* que você construiu.

In [27]:
```
df['default'].value_counts()
```

Out[27]:

```
bom    20506
mau      431
Name: default, dtype: int64
```