Documentação *Sprint* 2

NOTA DO AUTOR

Este trabalho está sendo conduzido como projeto integrados da FATEC são josé dos campos para a empresa SPC.

Os dados foram analisados no *jupyter notebook*. O arquivo tratado foi o "STG_MVT_CRD" e seus dados foram omitidos devido a proteção de sigilo. Só serão exibidas as informações; não os dados.

DESENVOLVIMENTO

O arquivo foi fornecido no excel com o formato ".xls". Ele foi convertido para a extensão ".csv" para análise no *jupyter notebook*. Os comandos utilizados foram a identificação de colunas com dados nulos, a conversão dos dados para o tipo correto, a contagem de dados nulos, a busca por dados iguais à 0 (zero) para a posterior substituição dos valores nulos, para a alteração de seu tipo de armazenamento; e a verificação de colunas com valores únicos, a fim de selecionar colunas como chave primária.

A biblioteca utilizada foi o pandas para a visualização dos dados e o método utilizado para sua importação foi o ".read_csv()" (Fig. 1). Pegou-se os dados gerais da tabela com um total de quinze linhas e doze colunas. Depreende-se, também, que as colunas de índice 1 à 4 contém valores nulos, analisando a contagem de valores não nulos[1]. Continuando a análise, é perceptível que a sequência de colunas de 1 à 3 tem as mesmas quantidade de dados nulos, o que é relevante dado que cada dado pertence a uma pessoa diferente que não se comunicam. Verifica-se, na coluna *Dtype* que os tipos de dados armazenados são: inteiro, ponto flutuante e string ou caracteres (Fig. 2).

[1] *Non-Null Count* na figura 2.

Figura 1: imagem das linhas 1 e 2. A linha 1 exibe a biblioteca pandas; a linha 2 mostra o código de importação do arquivo "STG_MVT_CRD" com sua extensão.

```
In [1]: import pandas as pd

In [2]: a = pd.read_csv('STG_MVT_CRD_csv.csv')
```

Figura 2: Informações da tabela. *RangeIndex* mostra quantas linhas tem; *Data columns* exibe o total de colunas; # exibe o incide; *Column* mostra o nome das colunas; *Non-Null Count* mostra quanto valores não nulos há e *Dtype* mostra o tipo de dado armazenado.

```
In [3]: a.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID_STG_MVT_CRD       15 non-null     int64
 1   VLR_SDO_UTZ_CRD_RTO  1 non-null      float64
 2   VLR_TOT_FAT          1 non-null      float64
 3   VLR_MIM_FAT          1 non-null      float64
 4   VLR_PCL_FAT          13 non-null     float64
 5   QTD_CLI_CAD_POS      15 non-null     int64
 6   QTD_MVT              15 non-null     int64
 7   DES_TIP_PSS          15 non-null     object
 8   ID_FNT_ITT           15 non-null     int64
 9   COD_MDL              15 non-null     object
 10  DAT_RSS_FNT_ITT      15 non-null     object
 11  DAT_INC_DBO          15 non-null     object
dtypes: float64(4), int64(4), object(4)
memory usage: 1.5+ KB
```

As colunas “DAT_RSS_FNT_ITT” e “DAT_INC_DBO” são do tipo data, apesar de ter sido consideradas *strings* ou caracteres. O método usado foi “.to_datetime()” - uma outra forma de fazer a mesma coisa é na hora de importar a tabela com o método “.read_csv()” adicionar como parâmetro “parse_dates=[ ]” e dentro dos colchetes escrever o nome das colunas - (Fig. 3).

Figura 3: método .to_datetime() para alterar tipos de dados para Data aplicado às colunas “DAT_RSS_FNT_ITT” e “DAT_INC_DBO”

```
In [4]: a['DAT_RSS_FNT_ITT'] = pd.to_datetime(a['DAT_RSS_FNT_ITT'])

In [5]: a['DAT_INC_DBO'] = pd.to_datetime(a['DAT_INC_DBO'])
```

Em seguida foi contado quanto valores nulos e não nulos havia nas colunas com pelo menos um valor nulo. Os métodos utilizados foram “.isnull()” e “.value_counts()”. Aquele retorna um valor booleano, *True* ou *False*, conforme seja nulo ou não, respectivamente; este conta a ocorrência de *Trues* e *Falses* na coluna (Fig. 4)[2].

[2] A figura só traz uma coluna com dados nulos, mas os códigos na íntegra estão na pasta “códigos” no github/equipespc.

Figura 4: código com os parâmetros ".isnull()" e ".value_counts()". ".isnull()" retorna um valor booleano comforme seja nulo ou não. ".value_counts()" conta a ocorrência de *Trues* e *Falses*.

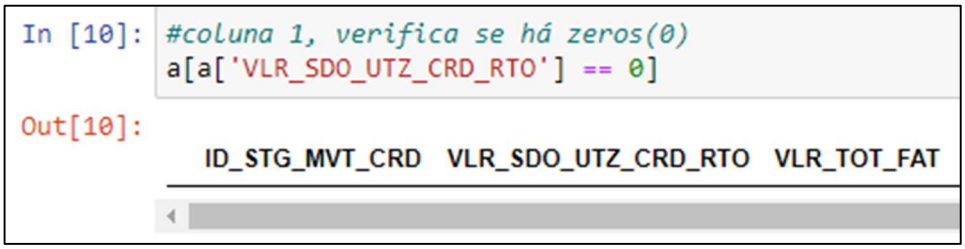
```
In [6]: #coluna 1, verifica quantos valores nulos há
        b = a['VLR_SDO_UTZ_CRD_RTO'].isnull().value_counts()
        b

Out[6]: True     14
        False     1
        Name: VLR_SDO_UTZ_CRD_RTO, dtype: int64
```

Dados nulos são dados indisponíveis ou não atribuídos, o que o torna diferente de zero, que é considerado número inteiro. Para a aplicação do código de valores nulos, que substitui valores nulos por zeros, procurou-se pôr zeros para que não houvesse modificações significativas nas colunas. O intuito de modificar os valores nulos para zero foi alterar o tipo de armazenamento de ponto flutuante para inteiro. O ponto flutuante ocupa um maior espaço na memória (4 bytes) em relação ao número inteiro (2 bytes). O código da figura 5 faz uma busca e exibição de todas as linhas cuja coluna explicitada é igual a zero. Como só houve o retorno dos nomes das colunas sem voltar nenhuma linha de dados, conclui-se que zeros são inexistentes naquela determinada coluna; caso contrário os valores nulos não seriam alterados. Deu-se prosseguimento a aplicação do código nulo.

Para melhor esclarecimento do código, faz-se necessário explicar sua composição. A porção "a['VLR_SDO_UTZ_CRD_RTO']"[3] resgata todos os valores da coluna "VLR_SDO_UTZ_CRD_RTO'" e verifica se é igual (símbolo: ==) à 0. Ele substituirá os valores por *Trues* e *Falses*. Todo o código "a['VLR_SDO_UTZ_CRD_RTO'] == 0" é inserido nos colchetes de a[ ] [4] a fim de retornar as linhas com os dados da tabela que satisfazem a condição. Conforme supracitado, não houve dados zeros (Fig. 5).

[3] código esboçado: a['VLR_SDO_UTZ_CRD_RTO'] == 0
[4] a[a['VLR_SDO_UTZ_CRD_RTO''] == 0]

Figura 5: Código de procura de valores zeros. “a[‘VLR_SDO_UTZ_CRD_RTO’] == 0” retorna valores booleanos; “a[a[‘VLR_SDO_UTZ_CRD_RTO’]]” resgata as linhas que satisfazem a condição.



Avaliou-se que os dados nulos das colunas poderiam ser substituídos por zeros sem alteração significativa nos dados. Após a aplicação do código de valores nulos criado, o arquivo foi chamado e armazenado na variável “a1” e em seguida verificou-se que os dados nulos foram de fato substituídos por zeros e o tipo de armazenamento mudou de ponto flutuante para números inteiros (Fig. 6).

Em seguida, foi buscado todas as colunas com valores únicos como candidatas a identificadores exclusivos. Os resultados das figuras 7 e 8 mostram que as colunas 0, 5, 6, e 9 contém valores únicos.

As colunas 1, 2, 3 e 4, que tem valores nulos, apresentaram dois valores únicos para as colunas 1 à 3; e a coluna 4, com exceção dos valores nulos, que são 2, tem todos os valores únicos (Fig. 9).

Figura 6: Arquivo com os valores nulos substituídos por zero e a visão geral do conteúdo. Como pode ser observado, não existem mais valores nulos nas colunas de um à três, corroborando que zero não é nulo.

```
In [14]: a1 = pd.read_csv('a1_STG_MVT_CRD_csv.csv')

In [15]: a1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID_STG_MVT_CRD       15 non-null     int64
 1   VLR_SDO_UTZ_CRD_RTO  15 non-null     int64
 2   VLR_TOT_FAT          15 non-null     int64
 3   VLR_MIM_FAT          15 non-null     int64
 4   VLR_PCL_FAT          15 non-null     int64
 5   QTD_CLI_CAD_POS      15 non-null     int64
 6   QTD_MVT              15 non-null     int64
 7   DES_TIP_PSS          15 non-null     object
 8   ID_FNT_ITT           15 non-null     int64
 9   COD_MDL              15 non-null     object
 10  DAT_RSS_FNT_ITT      15 non-null     object
 11  DAT_INC_DBO          15 non-null     object
dtypes: int64(8), object(4)
memory usage: 1.5+ KB
```

Figuras 7 e 8: As colunas 0, 5, 6 e 9 contém valores únicos.

```
In [20]: #coluna 9
         a1['COD_MDL'].is_unique

Out[20]: True

In [21]: #coluna 0
         a1['ID_STG_MVT_CRD'].is_unique

Out[21]: True
```

```
In [26]: #coluna 5
         a1['QTD_CLI_CAD_POS'].is_unique

Out[26]: True

In [27]: #coluna 6
         a1['QTD_MVT'].is_unique

Out[27]: True
```

Figura 9: Exibe as colunas com valores nulos e a quantidade de exemplares em cada uma. Destaca-se a coluna quatro que tem todos os valores únicos com exceção dos nulos, que são dois.

```
In [34]: #coluna 1
         a1['VLR_SDO_UTZ_CRD_RTO'].value_counts().values

Out[34]: array([14,  1], dtype=int64)

In [35]: #coluna 2
         a1['VLR_TOT_FAT'].value_counts().values

Out[35]: array([14,  1], dtype=int64)

In [36]: #coluna 3
         a1['VLR_MIM_FAT'].value_counts().values

Out[36]: array([14,  1], dtype=int64)

In [37]: #coluna 4, tirando as valores nulos, todos são exclusivos.
         a1['VLR_PCL_FAT'].value_counts().values

Out[37]: array([2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```