

Notebook Jupyter 1b_trataABIMOTO13

Importação e tratamento Tabela ABIMOTO v13

A tabela ABIMOTO v13 é fornecida em PDF, que foi manualmente transformado em uma planilha do Excel no formato XLSX.

Importação das Bibliotecas

```
In [1]: import pandas as pd
import numpy as np
import os, time
```

```
In [2]: # Data e hora da execução do script
initot=time.time()
print(f'Código executado em {time.strftime("%d/%m/%Y às %H:%M", time.localtime(time.time()))}')
Código executado em 25/01/2022 às 12:13
```

Definindo dados gerais

Cria variáveis com nomes das colunas e seus tipos

```
In [3]: tipos = { 'ITEM' : int,
                 'PARTES E PEÇAS' : str,
                 'MOTO PARTS' : str,
                 'NCM' : str,
                 'UNI' : str,
                 'VMLE' : float }
colunas = list(tipos.keys())
```

Inicializa a variável que conterá o tamanho total do dataset original

```
In [4]: tamanhoDataset=0
```

Inicializa um dataframe vazio que conterá os dados finais

```
In [5]: df = pd.DataFrame(columns = colunas)
df.head()
```

```
Out[5]:
```

ITEM	PARTES E PEÇAS	MOTO PARTS	NCM	UNI	VMLE
------	----------------	------------	-----	-----	------

Importa o arquivo Excel XLSX, trata e concatena no DataFrame final

```
In [6]: # Executa a importação e tratamento do arquivo
arq = './bases/ABIMOTO13.xlsx'
print(f'Iniciando {arq}.')
df = pd.read_excel(arq, dtype=tipos, names=colunas)
print('DataFrame carregado...\nAplicando filtros...')
# Elimina os registros sem valores ou nulos
df = df.dropna()
# Incrementa o tamanho do Dataset
tamanhoDataset += df[df.columns[0]].count()
print('Quantidade de registros válidos: ' + str(tamanhoDataset))
# Filtra o DataFrame somente com os registros de interesse
# Filtro 1: NCM de interesse: 87141000
indiceNCM = df['NCM'] == '87141000'
df = df[indiceNCM]
# Filtro 2: Descrição contendo 'kit' e 'transm'
df = df[df['PARTES E PEÇAS'].str.contains("kit", case=False)]
df = df[df['PARTES E PEÇAS'].str.contains("transm", case=False)]
print(arq + ' finalizado.\n')
```

```
Iniciando ./bases/ABIMOTO13.xlsx.
DataFrame carregado...
Aplicando filtros...
Quantidade de registros válidos: 129
./bases/ABIMOTO13.xlsx finalizado.
```

Ajustando a descrição e indicação de presença de corrente com retentor no kit

```
In [7]: df.head()
```

Out[7]:

	ITEM	PARTES E PEÇAS	MOTO PARTS	NCM	UNI	VMLE
77	78	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	TRANS.SET 1045 C100 BIZ 15D+428HX108+35D W/PR.	87141000	RETENTOR KIT COM	6.47
78	79	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...	TRANS.SET 1045 C100 BIZ 15D+428HX108+35D W/PR.	87141000	RETENTOR KIT SEM	3.90
79	80	KIT TRANSMISSÃO 1045 (CORRENTE.COROA. PINHAO) ...	TRANS.SET 1045 (CHAIN + SPROCKET) CBX 250 TWISTER	87141000	KIT COM RETENTOR	9.30
80	81	KIT TRANSMISSÃO 1045 (CORRENTE.COROA. PINHAO) ...	TRANS.SET 1045 (CHAIN + SPROCKET) CBX 250 TWISTER	87141000	KIT SEM RETENTOR	6.25
81	82	KIT TRANSMISSÃO 1045 FAZER 250	TRANSMISSION SET + H CHAIN FAZER 251	87141000	RETENTOR KIT COM	8.20

Em virtude de haver observação se há ou não corrente com retentor na coluna UNI faremos a adequação da descrição pra conter tal informação.

```
In [8]: obs = ["COM RETENTOR" if "COM" in x else "SEM RETENTOR" for x in df.UNI.str.split()]
df["PARTES E PEÇAS"] = df["PARTES E PEÇAS"] + " " + obs
```

Em seguida define-se a coluna UNI como 'KIT'

```
In [9]: df['UNI'] = 'KIT'
```

E por fim cria-se a coluna RETENTOR do tipo *boolean* para indicar ou não a presença do retentor no kit.

```
In [10]: from funcoesTCC import retentorAux # função importada do módulo de funções criarMod
elo
df['RETENTOR'] = df['PARTES E PEÇAS'].apply(retentorAux)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\47929790304\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

Resetando o índice do DataFrame importado

```
In [11]: df.reset_index(inplace=True, drop=True)
```

Excluindo colunas desnecessárias

```
In [12]: excluir=['ITEM',
                 'MOTO PARTS',
                 'NCM',
                 'UNI']
```

```
In [13]: df=df.drop(excluir, axis=1)
```

Verificando o DataFrame final

```
In [14]: df.head(3)
```

```
Out[14]:
```

		PARTES E PEÇAS	VMLE	RETENTOR
0	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...		6.47	True
1	KIT TRANSMISSÃO 1045 C100 BIZ 15D+428HX108+35D...		3.90	False
2	KIT TRANSMISSÃO 1045 (CORRENTE.COROA. PINHAO) ...		9.30	True

```
In [15]: df.tail(3)
```

```
Out[15]:
```

		PARTES E PEÇAS	VMLE	RETENTOR
13	KIT TRANSMISSÃO XRE - 300 SEM RETENTOR		6.0	False
14	KIT TRANSMISSÃO XTZ - 250 LANDER COM RETENTOR		8.2	True
15	KIT TRANSMISSÃO XTZ - 250 LANDER SEM RETENTOR		4.4	False

```
In [16]: df.shape
```

```
Out[16]: (16, 3)
```

Exportando o DataFrame

Exportando para um arquivo CSV

```
In [17]: df.to_csv(r'./bases/dfABIMOTOv13.csv', index = False, header = True)
```

Exportando para um arquivo de planilha do Excel

```
In [18]: df.to_excel(r'./bases/dfABIMOT0v13.xlsx', index = False, header = True)
```

Compara o tamanho total do Dataset inicial e final

```
In [19]: print('Qtd de registros Dataset original: ' + str(tamanhoDataset))
print('Qtd de registros Dataset final:      ' + str(df[df.columns[0]].count()))
```

```
Qtd de registros Dataset original: 129
Qtd de registros Dataset final:    16
```

```
In [20]: tempotot=time.time()-initot
if tempotot>60:
    print(f'Tempo total de execução: {tempotot/60:.2f} minutos.')
else:
    print(f'Tempo total de execução: {tempotot:.2f} segundos.')
```

```
Tempo total de execução: 1.20 segundos.
```