Capítulo 9 - Solução

December 21, 2020

1 Atividade 3: Para casa

1.1 Exercício 1: extração de datasets

Extraia outros dois conjuntos de dados de sua preferência e faça uma análise sobre ele em markdown. Abordando as características, possíveis problemas de negócio.

Não precisa entrar em códigos.

Solução:

Os conjuntos de dados escolhidos foram:

1. `Electric_Production.csv`

Disponível em: > https://github.com/biangomes/projetos-e-estudos/blob/main/consumo-eletricidade/Electric_Production.csv

Descrição:

Analisando brevemente o conjunto de dados escolhido, é possível ver que existem duas colunas: `DATE` e `Value`. Trata-se de dados sobre consumo de eletricidade entre 01/01/1985 até 01/01/2018. Quando se fala de consumo de eletricidade, é interessante saber se há periodicidade no consumo de eletricidade. Como a distribuição temporal é feita mensalmente ao longo dos anos, pode ver se na época de verão o consumo é maior; nas férias escolares o consumo aumenta? As pessoas estão consumindo mais energia? Se sim, quanto a mais? Se não, quanto a menos estão consumindo?

2. `trem.csv`

Disponível em: > https://github.com/biangomes/projetos-e-estudos/blob/main/ferroviaria/trem.csv

Descrição:

Trata-se de um conjunto de dados de uma ferroviária composto por 3 colunas: `ID`, `Datetime` e `Count` em que:

- `ID`: número identificador;
- `Datetime`: data no formato de `dia-mês-ano horário`;
- `Count`: quantidade de passageiros no trem.

Para gerar insights às companhias, é necessário responder algumas perguntas como: nos horários de entrada e saída do trabalho/escola, a quantidade de passageiros é maior? quais são esses horários? a demanda por trens é maior em alguns meses? seria válido, em termos financeiros, aumentar a

quantidade de trens? seria válido, em termos financeiros, mudar os horários que estes trens passam? qual a diferença da quantidade de passageiros entre os meses de férias e de trabalho/escola?

1.2 Exercício 2: Identificação de problemas no arquivo precipitacao.xlsx

Aponte eventuais problemas no arquivo precipitacao.xlsx, com relação aos tipos de dados, NaN, etc.

Solução:

Para resolver este problema iremos utililizar o arquivo `precipitação.csv` (pode utilizar o arquivo no formato `.xlsx` também):

Para identificar os problemas, é necessário ver as características do conjunto de dados como o tipo dos objetos, presença de `NaN` e afins.

Considerando que o pacote do `pandas` e o dataset foram importados, podemos iniciar a verificação dos tipos dos objetos através do código

```
# importando o dataset
df = pd.read_excel ('precipitacao.xlsx')
# df = pd.read_csv('precipitacao.csv')

# verificando as informacoes sobre os objetos
df.info()
```

[11]:
```
# importando o pandas
import pandas as pd

# importando o dataset
df = pd.read_excel('precipitacao.xlsx')

# verificando as informacoes sobre o conjunto de dados
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87 entries, 0 to 86
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Ano           87 non-null     int64
 1   Precipitacao  87 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.5 KB
```

No conjunto de dados, existem 86 entradas de dados distribuídos em duas colunas: `Ano` e `Precipitacao`. Através da informação da coluna `Non-Null Count`, **não** existem valores nulos, o que quer dizer que todos os dados estão preenchidos. A coluna `Dtype` diz o tipo de dado que cada coluna recebe. A coluna `Precipitacao` recebe valores do tipo `float64`, o que já era esperado. Contudo, a coluna `Ano` deveria receber um dado do tipo `Datetime`. Além disso, como cada dado

corresponde unica e exclusivamente a um período (ano, neste caso) torna-se redundante ter uma coluna como index e essas outras duas.

Em suma, tem-se dois problemas (ou melhorias):

- conversão da coluna `Ano` para index;
- conversão da coluna `Ano` para o tipo `Datetime`.

Para resolver os problemas apontados, os códigos abaixo podem ser escritos:

```
# converte o tipo de objeto para datetime
df.Ano = pd.to_datetime(df.Ano, format='%Y')

# 'seta' a coluna Ano como index
df.index = df.Ano

# drop na coluna Ano para eliminar a redundancia
df.drop('Ano', inplace=True, axis=1)

# outra verificacao para ver se as modificacoes tiveram sucesso
df.info()
```

[13]:
```
# convertendo o tipo para datetime
df.Ano = pd.to_datetime(df.Ano, format='%Y')

# setando como index
df.index = df.Ano

# drop na coluna Ano
df.drop('Ano', inplace=True, axis=1)

# verificando o tipo dos objetos
df.info()
```

Problemas encontrados:

- Tipo inadequado para a coluna `Ano`: veja que ela está como `int64`. É de nosso interesse deixá-la como `Datetime`.

1.3 Exercício 3: extração de informações estatísticas

Extraia informações estatísticas do conjunto de dados de ações. Faça para outros 5 ativos.

A sintaxe adotada pelo site do *Yahoo Finance* é: CODIGODAACAO.SA

[5]:
```
import pandas as pd
from pandas_datareader import data as web
```

[6]:
```
tickers = ['CMIG4.SA', 'OIBR4.SA', 'JHSF3.SA', 'WEGE3.SA', 'MGLU3.SA']
ativos = pd.DataFrame()
```

```
for t in tickers:
    ativos[t] = web.DataReader(t, data_source='yahoo', start='2020-07-01')['Adj␣
 ↪Close']
```

[7]:
```
ativos.rename(columns={'CMIG4.SA':'CEMIG', 'OIBR4.SA': 'OI', 'JHSF3.SA':␣
 ↪'JHSF', 'WEGE3.SA':'WEG', 'MGLU3.SA':'MAGAZINE LUIZA'}, inplace=True)
```

[8]:
```
for i in ativos:
    print('--\nATIVO: {}\n'.format(i))
    print(ativos[i].describe());
```

```
--
ATIVO: CEMIG

count    119.000000
mean      11.177562
std        0.939661
min       10.100000
25%       10.558518
50%       10.855820
75%       11.392100
max       14.010000
Name: CEMIG, dtype: float64
--
ATIVO: OI

count    119.000000
mean       2.428487
std        0.514262
min        1.450000
25%        2.260000
50%        2.410000
75%        2.750000
max        3.650000
Name: OI, dtype: float64
--
ATIVO: JHSF

count    119.000000
mean       7.912532
std        0.839854
min        6.340000
25%        7.312030
50%        7.700000
75%        8.350000
max       10.320243
Name: JHSF, dtype: float64
```

```
--
ATIVO: WEG

count    119.000000
mean      69.670034
std        8.481884
min       52.435562
25%       64.488003
50%       68.129219
75%       77.091316
max       85.706993
Name: WEG, dtype: float64
--
ATIVO: MAGAZINE LUIZA

count    119.000000
mean      22.678987
std        2.225097
min       17.364321
25%       20.925121
50%       22.535000
75%       24.540001
max       27.450001
Name: MAGAZINE LUIZA, dtype: float64
```

1.4 Exercício 4: Série temporal e `Datetime`

Utilizando o arquivo precipitacao.xlsx, deixe-a no formato de uma série temporal.

Lembre-se que uma série temporal contém as datas como index.

[9]:
```
# importando o pandas
import pandas as pd
```

[10]:
```
# colocando o arquivo em um dataframe
df = pd.read_excel('precipitacao.xlsx', engine='openpyxl')
```

[11]:
```
# visualizando os tipos dos objetos
df.dtypes
```

[11]:
```
Ano              int64
Precipitacao   float64
Unnamed: 2     float64
dtype: object
```

Como a coluna `Ano` está em int64 e queremos trabalhar com ela em `Datetime`, vamos convertê-la.

[12]:
```
df['Ano'] = pd.to_datetime(df['Ano'], format='%Y')
```

[13]:
```
df.dtypes
```

[13]:
```
Ano             datetime64[ns]
Precipitacao           float64
Unnamed: 2             float64
dtype: object
```

Passando a coluna `Ano` para index.

[14]:
```
df.index = df['Ano']
```

[15]:
```
df
```

[15]:

	Ano	Precipitacao	Unnamed: 2
Ano			
1900-01-01	1900-01-01	33.20	NaN
1901-01-01	1901-01-01	27.55	NaN
1902-01-01	1902-01-01	31.81	NaN
1903-01-01	1903-01-01	32.50	NaN
1904-01-01	1904-01-01	31.01	NaN
…	…	…	…
1982-01-01	1982-01-01	32.86	NaN
1983-01-01	1983-01-01	31.55	NaN
1984-01-01	1984-01-01	29.74	NaN
1985-01-01	1985-01-01	38.98	NaN
1986-01-01	1986-01-01	34.65	NaN

[87 rows x 3 columns]

O ‘Ano‘ já está como index e para não conter duplicidade removeremos a coluna.

[16]:
```
df.drop('Ano', inplace=True, axis=1)
```

[17]:
```
df
```

[17]:

	Precipitacao	Unnamed: 2
Ano		
1900-01-01	33.20	NaN
1901-01-01	27.55	NaN
1902-01-01	31.81	NaN
1903-01-01	32.50	NaN
1904-01-01	31.01	NaN
…	…	…
1982-01-01	32.86	NaN
1983-01-01	31.55	NaN
1984-01-01	29.74	NaN
1985-01-01	38.98	NaN
1986-01-01	34.65	NaN

```
[87 rows x 2 columns]
```

Agora é possível trabalhar com esse dataset como série temporal.