escola britânica de artes criativas & tecnologia

Analista de Dados

Data Wrangling I

Módulo | Análise de Dados: Data Wrangling I

Caderno de Aula

Professor André Perez

Tópicos

1. DataFrame Pandas;
2. Selecão e Filtros;
3. Inserção, Deleção e Atualização.

Aulas

0. Estruturas de dados

- **Não estruturado**: texto, imagem, aúdio, etc.
- **Semi estruturado**: html, json, etc.
- **Estruturado**: tabelas, planilhas, etc.

1. DataFrame Pandas

1.1. Pacote Pandas

Pacote Python para manipulação de dados (talvez o mais utilizado). A documentação pode ser encontrada neste link. A abstração base do pandas é o DataFrame, uma estrutura de dados Python de duas dimensões utilizado para representar tabelas.

Exemplo: Criar um DataFrame Pandas a partir de um dicionário Python.

In [ ]:

```
transacoes = dict(
    id=[571, 572, 573],
    data=['19-01-2021', '19-01-2021', '23-01-2021'],
    valor=[371.30, 57.19, 101.21],
    categoria=['supermercado', 'farmacia', 'outros']
)
```

In [ ]:
```
import pandas as pd

transacoes_df = pd.DataFrame(transacoes)
```

In [ ]:
```
transacoes_df
```

In [ ]:
```
type(transacoes_df)
```

- Atributos

In [ ]:
```
transacoes_df.columns
```

In [ ]:
```
transacoes_df.dtypes
```

In [ ]:
```
transacoes_df.index
```

In [ ]:
```
transacoes_df.shape
```

- Métodos

In [ ]:
```
transacoes_df.head(n=1)
```

In [ ]:
```
transacoes_df.tail()
```

In [ ]:
```
transacoes_df.info()
```

In [ ]:
```
transacoes_df[['id', 'valor']].describe().T # colunas numéricas
```

In [ ]:
```
transacoes_df[['data', 'categoria']].describe().T # colunas categóricas
```

Exemplo: Criar um DataFrame Pandas a partir de um arquivo csv.

In [ ]:
```
%%writefile github.csv
ranking;project;language;stars;stars_today;forks
1;plow;go;1304;574;38
2;n8n;typescript;15668;280;1370
3;slides;go;3218;265;80
4;defi-developer-road-map;;636;247;49
5;pytorch-image-models;python;11065;101;1646
6;javascript-algorithms;javascript;110768;248;18331
7;paddleclas;python;1429;283;323
8;reddit_sentiment_trader;python;369;71;60
9;augly;python;2849;393;99
10;self-taught-guide-to-cloud-computing;;863;179;84
```

In [ ]:
```
import pandas as pd

github_df = pd.read_csv('github.csv', sep=';')
```

In [ ]:
```
github_df
```

In [ ]:
```
type(github_df)
```

- Atributos

In [ ]:
```
github_df.columns
```

In [ ]:
```
github_df.dtypes
```

In [ ]:
```
github_df.index
```

In [ ]:
```
github_df.shape
```

- Métodos

In [ ]:
```
github_df.head()
```

In [ ]:
```
github_df.tail()
```

In [ ]:
```
github_df.info()
```

In [ ]:
```
github_df[['project', 'language']].describe().T # colunas categóricas
```

In [ ]:
```
github_df[['ranking', 'stars', 'stars_today', 'forks']].describe().T
# colunas numéricas
```

1.2. Anatomia de um DataFrame

- **Série**: `Coluna` de um DataFrame;
- **Índice**: Indentificador de uma `linha` de um DataFrame.

In [ ]:
```
github_df
```

2. Seleção e Filtros

2.1. Série

Uma **série** é uma **coluna** de um **dataframe**. Para selecionar uma coluna utilizamos a seguinte

notação (similiar a indexação de listas Python):

```
serie = dataframe['<nome-da-coluna>']
```

Nota: Repare no uso das chaves simples `[ ]`.

- **Exemplo**: Coluna linguagem de programação do dataframe `github_df`:

In [ ]:
```
linguagem_serie = github_df['language']
```

In [ ]:
```
linguagem_serie
```

In [ ]:
```
type(linguagem_serie)
```

2.1.1 Seleção

- **Exemplo**: Indexação simples com método `loc` (similar a lista Python):

In [ ]:
```
top_1_linguagem = linguagem_serie.loc[0]
```

In [ ]:
```
top_1_linguagem
```

In [ ]:
```
type(top_1_linguagem)
```

- **Exemplo**: Fatiamento ou *slicing* com método `loc` (similar a lista Python):

In [ ]:
```
top_5_linguagem = linguagem_serie.loc[0:5]
```

In [ ]:
```
top_5_linguagem
```

In [ ]:
```
type(top_5_linguagem)
```

2.1.2 Filtros

- **Exemplo**: Filtro funcional:

In [ ]:
```
linguagem_serie[lambda linguagem: linguagem == 'python']
```

- **Exemplo**: Filtro funcional com novos índices:

In [ ]:
```
linguagem_serie[
    lambda linguagem: linguagem == 'python'
].reset_index(drop=True)
```

2.2. DataFrame

Um **conjunto** de **colunas** ou **séries** é um novo **dataframe**. Para selecionar um conjunto de colunas utilizamos a seguinte notação:

```
novo_dataframe = dataframe[['<nome-da-coluna-a>', '<nome-da-coluna-b>', ...]]
```

Nota: Repare no uso das chaves duplas `[[ ]]`.

- **Exemplo**: Colunas ranking e linguagem de programação do dataframe `github_df`:

In [ ]:
```
ranking_linguagem_df = github_df[['ranking', 'language']]
```

In [ ]:
```
ranking_linguagem_df
```

In [ ]:
```
type(ranking_linguagem_df)
```

2.2.1 Seleção

- **Exemplo**: Indexação simples (linha) com método `loc` (similar a lista Python):

In [ ]:
```
top_1_linguagem = ranking_linguagem_df.loc[0]
```

In [ ]:
```
top_1_linguagem
```

In [ ]:
```
type(top_1_linguagem)
```

- **Exemplo**: Indexação simples (linha e coluna) com método `loc` (similar a lista Python):

In [ ]:
```
top_1_linguagem = github_df.loc[0, ['ranking', 'language']]
```

In [ ]:
```
top_1_linguagem
```

In [ ]:
```
type(top_1_linguagem)
```

- **Exemplo**: Fatiamento ou *slicing* (linhas) com método `loc` (similar a lista Python):

In [ ]:
```
top_5_ranking_linguagem = ranking_linguagem_df.loc[0:5]
```

In [ ]:
```
top_5_ranking_linguagem
```

In [ ]:
```
type(top_5_ranking_linguagem)
```

- **Exemplo**: Fatiamento ou *slicing* (linhas e colunas) com método `loc` (similar a lista Python):

In [ ]:
```
top_5_ranking_linguagem = github_df.loc[0:5, ['ranking', 'language']]
```

In [ ]:
```
top_5_ranking_linguagem
```

In [ ]:
```
type(top_5_ranking_linguagem)
```

2.2.2 Filtros

- **Exemplo**: Filtro com o método `query`:

In [ ]:
```
ranking_linguagem_df.query('language == "python"')
```

In [ ]:
```
ranking_linguagem_df.query('language == "python" & ranking > 5')
```

In [ ]:
```
ranking_linguagem_df.query('language == "python" | language == "go"')
```

3. Inserção, Deleção e Atualização

3.1. Série

3.1.1 Inserção

- **Exemplo**: Adição de elementos com o método `append`:

In [ ]:
```
linguagem_serie.append(pd.Series(['java', 'python']), ignore_index=True)
```

In [ ]:
```
linguagem_serie
```

In [ ]:
```
linguagem_serie = linguagem_serie.append(
    pd.Series('java'),
    ignore_index=True
)
```

In [ ]:
```
linguagem_serie
```

3.1.2 Deleção

- **Exemplo**: Remoção de elementos com filtro funcional:

In [ ]:
```
linguagem_serie = linguagem_serie[lambda linguagem: linguagem != 'python']
```

3.1.3 Atualização

- **Exemplo**: Atualização de elementos com indexação simples:

In [ ]:
```
linguagem_serie.loc[0] = 'Go'
```

In [ ]:
```
linguagem_serie
```

- **Exemplo**: Atualização de elementos com fatiamento ou *slicing*:

In [ ]:
```
linguagem_serie.loc[0:2] = pd.Series(['Go', 'Typescript', 'Go'])
```

In [ ]:
```
linguagem_serie
```

- **Exemplo**: Atualização de elementos com filtro funcional:

In [ ]:
```
linguagem_serie[lambda linguagem: linguagem == 'python'] = 'Python'
```

In [ ]:
```
linguagem_serie
```

3.2. DataFrame

3.1.1 Inserção

- **Exemplo**: Adição de linhas com o método `append`:

In [ ]:
```
projeto = dict(
    ranking=[11],
    project=['signoz'],
    language=['typescript'],
    stars=[2651],
    stars_today=[491],
    forks=[115]
)
```

In [ ]:
```
github_df.append(pd.DataFrame(projeto), ignore_index=True)
```

In [ ]:
```
github_df
```

In [ ]:
```
github_df = github_df.append(pd.DataFrame(projeto), ignore_index=True)
```

In [ ]:
```
github_df
```

3.1.2 Deleção

- **Exemplo**: Remoção de linhas com o método `query`:

In [ ]:
```
github_df.query('language != "Python"')
```

3.1.3 Atualização

- **Exemplo**: Atualização de **um** elemento com o método `loc`:

In [ ]:
```
github_df.loc[0, 'language'] = 'go'
```

In [ ]:
```
github_df
```

- **Exemplo**: Atualização de **diversos** elementos com o método `apply`:

In [ ]:
```
github_df['language'] = github_df['language'].apply(
    lambda linguagem: 'python' if linguagem == 'Python' else linguagem
)
```

In [ ]:
```
github_df
```