**CRIANDO, EXPORTANDO E IMPORTANDO DATAFRAMES**

- Criando uma Série simples

# Criando uma Série

s = pd.Series([3,2,1], index=['a','b','c'])

print(s,"\n")

- Criando um DataFrame

# Criando um DataFrame

''' Dicionário '''

data = {

    'País': ['Portugal', 'Peru', 'Chile'],

    'Capital': ['Lisboa', 'Lima', 'Santiago']

}

df = pd.DataFrame(data, columns=['País', 'Capital'])

print(df,"\n")

- Exportando nosso DataFrame

# Exportando nosso DataFrame

novo\_df.to\_csv('exemplo\_df.csv', sep=';', index=False)

- Importando nosso DataFrame

# Importando nosso Dataframe

dados = pd.read\_csv('exemplo\_df.csv',sep=';')

print(dados,"\n")

**OPERAÇÕES BÁSICAS**

- Incluindo uma nova linha

# Incluindo uma linha com dados no DataFrame

novo\_dado = ['Espanha', 'Madrid', 99]

dados.loc[len(dados)] = novo\_dado

print(dados,"\n")

- Incluindo uma nova coluna

# Incluindo uma coluna com dados no DataFrame

moeda\_local = 'Euro','Peso','Peso','Real','Euro'

dados.insert(loc=3, column="Moeda", value=moeda\_local)

print(dados,"\n")

- Incluindo uma linha

# Deletando linhas de um DataFrame

novo\_df = dados.drop(labels=1)

print(novo\_df,"\n")

- Incluindo uma coluna

# Deletando colunas do DataFrame

del novo\_df['População']

print(novo\_df,"\n")

- Renomeando o index

# Renomeando o index

novo\_df.index.names = ['Nº']

print(novo\_df,"\n")

- Renomeando o index com o método rename

dist\_freq\_qualitativas.rename(index= {0: 'Masculino', 1:'Feminino'}, inplace=True)

- Renomeando as colunas

# Renomeando as colunas

novo\_df.columns = ['Países', 'Capitais', 'População', 'Moeda']

print(novo\_df)

- Alterando o nome de uma coluna

# Alterando o nome da coluna

cursos = cursos.rename(columns = {'Nome do curso' : 'nome\_do\_curso'})

print(cursos.head())

- Incluindo uma coluna de id utilizando-a como o index

# Criando uma coluna para ser o index

# Usando a coluna id como index do DataFrame

cursos['id'] = cursos.index + 1

cursos = cursos.set\_index('id')

print(cursos.head())

- Trocando valores das linhas de um DataFrame

# Alterando para portugues a coluna dia da semana

print(gorjetas.dia\_da\_semana.unique())

dias = {              # Criando um dicionario para renomear os campos dia da semana

    'Sun' : 'Domingo',

    'Sat' : 'Sabado',

    'Thur' : 'Quinta',

    'Fri' : 'Sexta'

}

gorjetas.dia\_da\_semana = gorjetas.dia\_da\_semana.map(dias)

print('\n',gorjetas.head())

- Filtrando alguma coisa em alguma coluna

# Filtrando somente um campo em uma coluna

print(gorjetas[gorjetas.sobremesa=='Sim'])

- Conversão de object para datetime

chocolura['mes'] = pd.to\_datetime(chocolura['mes'])

print(chocolura.dtypes)

- Apresentando valores nulos

print("Quantidades de dados nulos:", chocolura.isna().sum().sum())

- Add os dias da semana com o método day\_name, quando temos uma coluna com data

# Utilizando o metodo day\_name para colocar o dia da semana

vendas\_por\_dia['dia\_da\_semana'] = vendas\_por\_dia['dia'].dt.day\_name()

print(vendas\_por\_dia.head(7))

**SELEÇÃO DE CONTEÚDO**

- Selecionando um valor no DataFrame informando a linha e coluna

# Seleciona um único valor por linha & coluna

d = dados.iloc[0][0]

print(d,'\n')

- Selecionando um valor no DataFrame informando a linha e coluna

# Seleciona um único valor por rótulo de linha & coluna

d = dados.iloc[0]['Pais']

print(d,'\n')

**RECUPERANDO INFORMAÇÕES**

INFORMAÇÕES BÁSICAS

- Retorna a quantidade de linhas x colunas

# (linhas, colunas)

d = dados.shape

print(d,"\n")

- Retorna uma explicação do index

# Descreve o indice

d = dados.index

print(d,"\n")

- Retorna os nomes das colunas

# Descreve as colunas

d = dados.columns

print(d,"\n")

- Retorna informações sobre o DataFrame

# Info sobre o dataframe

d = dados.info()

print(d,"\n")

- Retorna os números válidos

# Numeros de valores não-NA

d = dados.count()

print(d,"\n")

RESUMOS

- Retorna a soma de valores da coluna fornecida

# Soma de valores

d = dados.População.sum()

print(d,"\n")

- Retorna a soma acumulada da coluna fornecida

# Soma acumulada

d = dados.População.cumsum()

print(d,"\n")

- Retorna os valores de max e min da coluna fornecida

# Valores max e min

d = dados.População.min

d2 = dados.População.max

print(f"O valor minimo é: {d}")

print(f"O valor máximo é: {d2}")

- Retorna um balanço de dados estatísticos do DataFrame

# Sumário estatistico

d = dados.describe()

print(d,"\n")

- Retorna a média dos valores

# Média dos valores

d = dados.mean()

print(d,"\n")

- Retorna a mediana dos valores

# Mediana dos valores

d = dados.median()

print(d,"\n")

**APRESENTAÇÃO**

- Apresentando os valores únicos de uma coluna

print(notas['nota'].unique())

- Apresentando quantas vezes os valores únicos da coluna apareceu

print(notas['nota'].value\_counts())

- Verificando se a base tem valores null

d = notas.isnull()

- Apresentando o histograma de uma coluna do DataFrame

notas.nota.plot(kind='hist')

plt.title("Histograma das notas dos filmes")

plt.show()

- Apresentando o boxplot de uma coluna do DataFrame

sns.boxplot(notas.nota)

plt.title("Boxplot das notas dos filmes")

plt.show()

- Gerando um gráfico scatterplot simples

# Gerando um gráfico scatterplot, comparativo entre o eixo x e y

porcentagem\_conta = sns.scatterplot(x='valor\_da\_conta', y='porcentagem', data=gorjetas)

plt.title("Visualmente o valor da conta não é proporcional ao valor da gorjeta")

- Gerando um gráfico de linha com o medoto relplot

porcentagem\_conta\_linha = sns.relplot(x='valor\_da\_conta', y='porcentagem', kind='line',data=gorjetas)

plt.show()

- Gerando um gráfico de pontos que mostra a tendência com o lmplot

# Apresentando o grafico anterior demonstrando a queda

porcentagem\_conta\_linha = sns.lmplot(x='valor\_da\_conta', y='porcentagem',data=gorjetas)

plt.show()

- Gerando um gráfico swarm tipo arvore de natal

# Espalha alguns pontos proximos de forma distribuida para uma melhor visualização

sns.catplot(x='hora\_do\_dia',y='valor\_da\_conta', kind='swarm',data=gorjetas)

plt.show()

- Gerando um gráfico violino

# Demonstra a visualização em formato de violino, onde a maior concentração será apresentada na parte mais gorda

sns.violinplot(x='hora\_do\_dia',y='valor\_da\_conta',data=gorjetas)

plt.show()

**EXCLUSÃO DE VALORES**

Removendo duplicadas de um DF

tipo\_de\_imovel.drop\_duplicates(inplace=True)

Removendo NaN de uma coluna

dados.dropna(subset = ['Valor'], inplace = True)

Alterando os valores de NaN para 0

dataset.fillna(0, inplace = True)

Excluindo colunas do DF com del

del dados\_aux['Valor Bruto']

Excluindo colunas do DF com pop

dados\_aux.pop('Valor Bruto m2')

Excluindo colunas do DF com drop

dados\_aux.drop(['Valor m2'], axis = 1, inplace=True)

**BUSCA**

Realizando uma busca query de nota em uma coluna

id1 = notas.query("filmeId==1").nota.mean().round(3)

Agrupando todos os filmesId e apresentando a media das notas

medias\_por\_filme = notas.groupby("filmeId").mean()["nota"].round(3)