Desafios de Programação

São propostos a seguir três desafios de programação. As respostas devem ser implementadas em Python 3, e serão executadas em um ambiente contendo apenas as bibliotecas padrão da linguagem, jupyter, sqlite3, numpy, pandas, scipy, statsmodels, matplotlib e saborn.

Os enunciados abaixo são versões simplificadas de problemas reais que ja enfrentamos. Todas as respostas serão avaliadas tanto do ponto de vista funcional, com testes automáticos, quanto sob o aspecto da qualidade do código fonte produzido (clareza, eficiência, complexidade, etc.).

É importante destacar que o resultado desse desafio não é o código em si, e sim sua capacidade de explicar e justificar e estruturação do código construido. Portanto o uso de google, stackoverflow é liberado para esse teste.

A entrega do desafio deve ser feita por meio do github em um repositório publico cujo link deve ser enviado por email para nós antes do prazo final da availiação.

reconcile_accounts

Escreva uma função que faça a conciliação de dois grupos de transações financeiras. Sua função, *reconcile_accounts*, deve receber duas listas de listas (representando as linhas dos dados financeiros) e deve devolver cópias dessas duas listas de listas com uma nova coluna acrescentada à direita das demais, que designará se a transação pôde ser encontrada (*FOUND*) na outra lista ou não (*MISSING*).

As listas de listas representarão os dados em quatro colunas tipo string:

- Data (em formato yyyy-mm-dd)
- Departamento
- Valor
- Beneficiário

Dados o arquivo transactions1.csv:

```
2020-12-04,Tecnologia,16.00,Bitbucket
2020-12-04,Jurídico,60.00,LinkSquares
2020-12-05,Tecnologia,50.00,AWS
```

E o arquivo transactions2.csv :

```
2020-12-04,Tecnologia,16.00,Bitbucket
2020-12-05,Tecnologia,49.99,AWS
2020-12-04,Jurídico,60.00,LinkSquares
```

Sua função deve funcionar do seguinte modo:

```
>>> import csv
>>> from pathlib import Path
>>> from pprint import pprint
>>> transactions1 = list(csv.reader(Path('transactions1.csv').open()))
>>> transactions2 = list(csv.reader(Path('transactions2.csv').open()))
>>> out1, out2 = reconcile_accounts(transactions1, transactions2)
>>> pprint(out1)
[['2020-12-04', 'Tecnologia', '16.00', 'Bitbucket', 'FOUND'],
 ['2020-12-04', 'Jurídico', '60.00', 'LinkSquares', 'FOUND'],
 ['2020-12-05', 'Tecnologia', '50.00', 'AWS', 'MISSING']]
>>> pprint(out2)
[['2020-12-04', 'Tecnologia', '16.00', 'Bitbucket', 'FOUND'],
 ['2020-12-05', 'Tecnologia', '49.99', 'AWS', 'MISSING'],
 ['2020-12-04', 'Jurídico', '60.00', 'LinkSquares', 'FOUND']]
```

Sua função deve levar em conta que em cada arquivo pode haver transações duplicadas. Nesse caso, cada transação de um arquivo deve corresponder uma única outra transaçãodo outro.

Cada transação pode corresponder a outra cuja data seja do dia anterior ou posterior, desde que as demais colunas contenham os mesmos valores. Quando houver mais de uma possibilidade de correspondência para uma dada transação, ela deve ser feita com a transação que ocorrer mais cedo. Por exemplo, uma transação na primeira lista com data 2020-12-25 deve corresponder a uma da segunda lista, ainda sem correspondência, de data 2020-12-24 antes de corresponder a outras equivalentes (a menos da data) com datas 2020-12-25 ou 2020-12-26.

data_modeling

Em um script ou jupyter notebook explore o dataset fornecido no arquivo breast_cancer.csv de forma a responder as completar o checklist a seguir.

O conjunto de dados "Breast Cancer Wisconsin (Diagnostic)" contém características extraídas dos exames de diagnóstico de câncer de mama, obtidas a partir de imagens digitalizadas das células, e a resposta, se o tumor é maligno ou benigno.

Do total de variáveis, foi feita uma seleção inicial, que levou às colunas

- Pontos côncavos médios: mean concave points
- Perímetro médio: mean perimeter
- Dimensão fractal média: mean fractal dimension
- Pior perímetro: worst perimeter
- Pior textura: worst texture
- Pior área: worst area
- **Tipo: target (1: benigno, 0: maligno).**

1 – Investigue a associação de cada uma dessas variáveis com o tipo de tumor via análise exploratória de dados. Comente os resultados obtidos. (sugestão: boxplots, coeficientes de correlação, pairplot...)

2 – Construa uma regressão linear para prever o tipo do tumor, indique as variaveis preditoras e as com significancia marginal neste modelo. Dica, utilize o modulo “OLS” do “statsmodels”. Comente os resultados obtidos.

```
>>> import statsmodels as sm
>>> res = sm.OLS(y, x).fit()
>>> print(res.summary())
```

3 – Construa uma regressão logistica para prever o tipo do tumor, indique as variaveis preditoras e as com significancia marginal neste modelo. Dica utilize o modulo “GLM” do “statsmodels” com reposta binomial. Comente os resultados obtidos.

```
>>> import statsmodels.api as sm
>>> res = sm.GLM(y, x, family=sm.families.Binomial()).fit()
>>> print(res.summary())
```

SQL

Considere as tabelas abaixo, que fazem parte do banco de dados de uma loja virtual:

Tabela `Clientes`:

ClienteID	Nome	Email
1	Ana Silva	ana@email.com
2	João Pereira	joao@email.com
3	Maria Souza	maria@email.com

Tabela `Pedidos`:

PedidoID	ClienteID	DataPedido	ValorTotal
101	1	2025-01-10	250.00
102	2	2025-01-12	100.00
103	2	2025-01-15	150.00
104	4	2025-01-17	200.00

Tabela `Pagamentos`:

PagamentoID	PedidoID	DataPagamento	ValorPago
1001	101	2025-01-11	250.00
1002	103	2025-01-16	150.00

1 – Escreva uma consulta SQL que mostre a lista de todos os clientes e seus respectivos pedidos, mesmo que o cliente não tenha feito nenhum pedido. A tabela resultante deve conter as colunas: Nome, PedidoID, DataPedido e ValorTotal.

2 – Escreva uma consulta SQL que exiba o total de pedidos realizados e o valor total de pedidos por cliente, apenas para os clientes que possuem pedidos registrados. A tabela resultante deve conter as colunas: Nome, QuantidadePedidos e ValorTotalPedidos.

3 – Escreva uma consulta SQL que exiba os pedidos que não possuem pagamentos registrados. A tabela resultante deve conter as colunas: PedidoID, DataPedido e ValorTotal.

O Arquivo do banco de dados em SQLLITE pode ser encontrado na pasta do projeto. Para abri-lo basta:

```
>>> import sqlite3
>>> conn = sqlite3.connect('caminho_do_database.sqlite')
>>> cursor = conn.cursor()
>>> resposta = cursor.execute(“SELECT * FROM…”).fetchall()
```