Teste de conhecimentos para vaga de estágio em TI Rocky

Descrição

O contexto do teste é uma atualização de sistema que "quebrou" o banco de dados, alterando algumas letras por caracteres, alterando o formato dos preços e removendo valores de quantidade de estoque. No teste deve-se tratar estes erros, pode-se utilizar liguagem python ou javascript. A minha escolha foi o Python pela familiaridade que tenho com a linguagem e pela facilidade que a liguagem traz para a manipulação de dados.

Utilizei para este script:

- Python 3.8
- Pandas
- NumPy
- VS Code
- Jupyter Notebook

Explicação do script - Tratamento

Importação das bibliotecas pandas e numpy:

```
import pandas as pd
import numpy as np
```

Função de leitura de dados

Definição da função de leitura do arquivo json, o pandas tem um metodo de leitura para cada tipo de arquivo, tranformando o resuldado em um DataFrame, que é uma estrutura propria do pandas, esta função retorna este DataFrame que será utilizado nas outras funções. Esta função tem um tratamento de erros, se por algum motivo o script não encotrar o arquivo ou não conseguir abri-lo, o script retorna-ra uma mensagem avisando.

```
def read_data():
    return pd.read_json("broken-database.txt")
```

Função de correção dos nomes

Para a correção de nomes foi utilizada a função replace da estrutura DataFrame, para esta função funcionar podemos definir um dicionario, onde a chave é o que será removido, e o valor o novo paramentro que ficará, esta fução troca os valores existentes nos dados pelos parametros passados com base neste dicionário, para troca apenas dos caracteres incorretos podemos passar o parametro "regex=True" para a função, que alterara apenas os caracteres. Passamos ainda o parametro "inplace=True" para manter as alterações feitas.

```
def correct_names(data):
    replace_dict = {"æ":"a", "¢":"c","ø":"o","ß":"b"}
```

```
    data["name"].replace(to_replace=replace_dict, regex=True,
inplace=True)
```

Função de correção de preços

Para esta função, utilizamos uma função de que converte o tipo de um dado especifico pelo passado como parametro da função, neste caso tinhamos valores salvos como string e o convertemos para tipo float por se tratar de preços.

```
def correct_price(data):
    data["price"].astype(np.float64)
```

Função de correção da quantidade

Aqui utilizamos uma função dos pandas que preenche valos não numericos NaN, Null pelo valor passado como parametro, neste caso queriamos preencher com zeros

```
def correct_quantity(data):
    data["quantity"] = data["quantity"].fillna(0)
```

Função de exportação de dados

Assim como para a leitura, o pandas possui um infinidade de extesões para as quais podemos exportar nossos dados após a finalização do tratamento, neste caso queremos exportar para json mesmo.

```
def export_json(data):
    data.to_json("saida.json", orient="records")
```

Explicação do script - Validação

Função de leitura do banco de dados correto

Para a validação é preciso fazer uma nova leitura dos dados, mas agora com o arquivo com os dados corrigidos. O arquivo saída.json, da mesma forma como na primeira função de leitura, esta função retorna dos dados em um formato de DataFrame que será utilizado nas funções seguintes.

```
def read_correct_data():
    return pd.read_json("saida.json")
```

Função para imprimir os produtos

Para uma primeira validação é necessário imprimir a lista de produtos ordenados primeiramente por categoria e logo após por id, aqui utilizei a função sort_values do pandas, é só passar uma lista com os nomes das colunas pelas quais quero ordenar os dados que esta função retorna o dataframe como queremos, para manter estas alterações passamos como parametro também "inplace=True", depois só precisamos printar estes dados na tela.

```
def print_list_of_products(data):
    data.sort_values(by=["category", "id"], inplace=True)
    print("\n\n", data, "\n\n")
```

Função para soma de estoque agrupado por categoria

Esta função pede para que somemos os valores dos produtos em estoque para cada categoria, para isso é necessários sabermos os valores totais de cada produto, multiplicando suas quantidade no estoque pelo preço unitário, criei uma nova coluna para armazenar estes valores. Com os valores totais de cada produto já contabilizado, utilizei a função groupby, que primeiramente agrupa os dados pelo parametro que passei, no caso a coluna "category", logo após pego os valores totais de cada produto e aplico a função sum(), que vai somar os totais de cada categoria. Com os dados prontos só preciso imprimi-los na tela.

```
def print_total_of_categories(data):
    data["total_supply"] = data["quantity"] * data["price"]
    print(data.groupby("category")["total_supply"].sum(), "\n\n")
```

A função main

A função main é a que roda quando o script é inicializado, esta função é responsável por chamar cada função criada anteriormente, para não haver erros cada função de leitura de arquivo tem um tratamento de erros para caso não consiga encontrar o arquivo na pasta especificada, avisando o usuário sobre o erro.

```
def main():
    data = read_data()
    correct_names(data)
    correct_price(data)
    correct_quantity(data)
    export_json(data)

    correct_data = read_correct_data()
    print_list_of_products(correct_data)
    print_total_of_categories(correct_data)
```