Documentação - Códigos JavaScript e SQL

Andressa Colares

January 2024

1 Explicando cada função do código em JS

1.1 const fs = require('fs').promises

Lida com a leitura dos dois arquivos corrompidos da base de dados.

1.2 async function BrokenBase(caminhoArq1, caminhoArq2)

É a função princial do código, pois é ela que lida com a correção dos arquivos corrompidos.

1.3 function corrigindoDados(conteudo)

Essa função lida diretamente com os dados corrompidos e os concerta, como solicitado.

1.4 async function exportToJson(data, outputPath)

Exporta os dados corrigidos para um novo arquivo JSON.

1.5 if (index1.vendas !== undefined)

Essa função tem como objetivo converter a string "vendas" para uma váriavel do tipo number.

1.6 BrokenBase(caminhoArq1, caminhoArq2);

Chama a função principal para corrigir e exportar as bases de dados.

2 Tratamento de Erros em JS

2.1 Tratamento de Erros com try-catch

O uso de try-catch nas sessões de letitura e manipulação de dados é utilizado no código para encontrar possíveis erros de leitura que possam atrapalhar ou

interromper a execução da aplicação. O try-catch também é utilizado novamente na função "exportToJson", possuindo o mesmo objetivo descrito anteriormente.

2.2 Validação da conversão do objeto "vendas"

Antes da string "vendas" ser convertido para number, a função if também verifica se vendas está definida. Isso evita problemas se a propriedade não existir ou não for um número válido.

2.3 Corrigindo a database

A própria função "CorrigindoDados" possue o objetivo de concertar os dados corrompidos para o tratamento de análise que foi realizado posteriormente.

3 Explicando cada função do código SQL

3.1 CREATE TABLE

A função CREATE TABLE, na primeira linha de dados_corrigidos_1, possue o objetivo de criar a tabela e as colunas. Cada coluna será escrita com um tipo específico de variável. Nesse caso, os valores de cada coluna foram descritos da seguinte forma: data_TEXT,id_marca INTEGER, vendas INTEGER,valor_veiculo INTEGER,nome TEXT, marca_carros TEXT

3.2 INSERT INTO

A função INSERT INTO foi utilizada para preencher os valores na tabela. Exemplo: "INSERT INTO ('dados_corrigidos_1' data_TEXT,id_marca INTEGER, vendas INTEGER,valor_veiculo INTEGER,nome TEXT, marca_carros TEXT) VALUES ('2022-01-01', '1', '40', '29000', 'Mobi');"

3.3 UPDATE

A função UPDATE foi utilizada para agregar os valores da dados_corrigidos _2 ao dataframe criado com os valors de dados _corrigidos _1. A exemplo temos: "UPDATE dados_corrigidos _1 SET marca _carros = 'Renault' WHERE id_marca = 11"

4 Pontos Importantes

Para obter resultados mais satisfatórios em relação à este estudo de caso, foi optado por unificar os arquivos JSON dados_corrigidos _1 e dados_corrigidos _2 em uma única tabela SQL. Para isso, utilizei a função UPDATE para inserir a coluna "marca _carros " e após inserida neste dataframe, foi novamente utilizado a função UPDATE, desta vez para inserir cada marca pelo seu respectivo id.

Esse método foi utilizado pois a base de dados poderia ser utilizada de uma maneira mais direta e facilitando também a colheita de embasamentos para os dados apontados na análise de vendas.