Explicação dos Códigos

Carlos Eduardo Alves Ferreira

Questão 01:

Objetivo: Criar uma função para ler o arquivo broken_database_1.json e broken_database_2.json, e com isso criar funções para percorrer o banco de dados corrompido e corrigir erros descritos, além de uma função para exportar os arquivos.json como saída.

a) Ler os arquivos JSON

O código utiliza o módulo fs (File System) para ler o conteúdo de dois arquivos JSON (broken_database_1.json e broken_database_2.json). A função lerArquivo recebe o nome do arquivo como parâmetro, lê o conteúdo do arquivo e converte esse conteúdo para um objeto JavaScript usando JSON.parse.

```
const fs = require('fs');

function lerArquivo(nomeArquivo) {
  try {
    const conteudo = fs.readFileSync('Database/'+ nomeArquivo, 'utf-8');
    return JSON.parse(conteudo);
  } catch (erro) {
    console.error(`Erro ao ler o arquivo ${nomeArquivo}: ${erro.message}`);
    return null;
  }
}

const database1 = lerArquivo('broken_database_1.json');

const database2 = lerArquivo('broken_database_2.json');
```

b) Corrigir nomes de marca e veículo

Primeiro iremos definir duas expressões regulares (regexA e regexO) para substituir caracteres específicos nas strings.

A função corrigirNomes recebe a base de dados como parâmetro e percorre suas entradas. Para cada entrada, se existir propriedade marca ou nome, ela substitui caracteres específicos nessas strings, corrigindo assim os erros que eram encontrados nas marcas e nome dos veiculos.

```
function corrigirNomes(database) {
  const regexA = /æ/g;
  const regexO = /ø/g;

  for (const entrada of database) {
    if (entrada.marca) {
      entrada.marca = entrada.marca.replace(regexA, 'a').replace(regexO, 'o');
    }
    if (entrada.nome) {
      entrada.nome = entrada.nome.replace(regexA, 'a').replace(regexO, 'o');
    }
  }
}
corrigirNomes(database1);
corrigirNomes(database2);
```

c) Corrigir vendas

A função corrigirVendas recebe a base de dados como parâmetro e percorre suas entradas. Para cada entrada, converte o valor da propriedade vendas para um número usando Number().

```
function corrigirVendas(database) {
  for (const entrada of database) {
    entrada.vendas = Number(entrada.vendas);
  }
}
corrigirVendas(database1);
corrigirVendas(database2);
```

d) Exportar um arquivo JSON com o banco corrigido

A função exportarArquivo é responsável por exportar um banco de dados corrigido para um novo arquivo JSON. Ela recebe o nome do arquivo e o banco de dados como parâmetros. O banco de dados é convertido para uma string JSON formatada e é escrita no arquivo especificado usando fs.writeFileSync. Além disso, as mensagens de sucesso ou erro são exibidas no console.

```
function exportarArquivo(nomeArquivo, database) {
  try {
    const conteudo = JSON.stringify(database, null, 2);
    fs.writeFileSync(nomeArquivo, conteudo, 'utf-8');
    console.log(`Arquivo ${nomeArquivo} exportado com sucesso.`);
  } catch (erro) {
    console.error(`Erro ao exportar o arquivo ${nomeArquivo}: ${erro.message}`);
  }
}
exportarArquivo('corrigido_database_1.json', database1);
exportarArquivo('corrigido_database_2.json', database2);
```

Questão 02:

Objetivo: Utilize a linguagem SQL para criar uma tabela única que contenha todos os dados necessários para o seu relatório, você deverá importar seus 2 arquivos JSON corrigidos para a plataforma SQL Online, no final você deverá exportar sua tabela resultante como um arquivo .CSV para utilizar no seu relatório.

a) Criar a tabela unificada

Este comando SQL cria uma nova tabela chamada 'tabela_unificada', cuja estrutura e dados são definidos pela consulta SELECT seguinte. A consulta SELECT seleciona diversas colunas das tabelas 'corrigido_database_1' e 'corrigido_database_2', incluindo dados como data, ID de marca, vendas, valor do veículo, nome da marca e vendas por marca. A tabela principal para a consulta é 'corrigido_database_1', e ela é unida à tabela 'corrigido_database_2' usando a condição de junção ON, que especifica que a junção deve ocorrer onde os valores da coluna 'id_marca_' em 'corrigido_database_1' são iguais aos valores da coluna 'id_marca' em 'corrigido_database_2'.

```
CREATE TABLE tabela_unificada AS

SELECT corrigido_database_1.data, corrigido_database_1.id_marca_,
corrigido_database_1.vendas, corrigido_database_1.valor_do_veiculo,
corrigido_database_1.nome, corrigido_database_2.marca, corrigido_database_
2.vendas_por_marca

FROM corrigido_database_1

JOIN corrigido_database_2

ON corrigido_database_1.id_marca_ = corrigido_database_2.id_marca;
```

b)Qual marca teve o maior volume de vendas?

```
SELECT marca, SUM(vendas) AS total_vendas

FROM tabela_unificada

GROUP BY marca
```

```
ORDER BY total_vendas DESC
LIMIT 1;
```

c) Qual veículo gerou a maior e menor receita?

```
SELECT nome, MAX(vendas * valor_do_veiculo) AS receita_maxima
FROM tabela_unificada;
SELECT nome, MIN(vendas * valor_do_veiculo) AS receita_minima
FROM tabela_unificada;
```

d) Qual a média de vendas do ano por marca?

```
SELECT marca, AVG(vendas) AS media_vendas
FROM tabela_unificada
GROUP BY marca;
```

e)Quais marcas geraram uma receita maior com número menor de vendas?

```
SELECT marca, SUM(valor_do_veiculo) / SUM(vendas) AS receita_por_venda
FROM tabela_unificada
GROUP BY marca
ORDER BY receita_por_venda DESC;
```

Adicionalmente, a linguagem de programação Python foi utilizada para a criação dos gráficos e para a análise de alguns dados.