Explicação dos códigos

Resumo

Este documento destina-se a função de orientar e relatar o funcionamento do processo de recuperação do de banco de dados corrompido no exercício proposto pela empresa Rocky.monks para a vaga de estágio em Web Analytics.

OBS: Para fins de fácil entendimento do código, optei por utilizar uma linguagem universal nos nomes de minhas funções e variaveis, o inglês.

Recuperando o arquivo json

Na proposta, é relatado que os arquivos json foram corrompidos e alguns caracteres foram substituidos por caracteres especias, além de alguns números de venda terem sido transformados em string.

Função de leitura dos arquivos Json

Para poder tratar os arquivos da forma correta, utilizei dentro de minha função readJson o recurso “require()” onde pude guardar os conteúdos dentro de duas variáveis e as retorno em formato de vetor para a Main() em uma constante chamada allData, após isso cada item deste vetor é armazenado em uma variavel que serão posteriormente enviadas para a função de ajuste de marca e nome do veiculo.

```
function readJson(dataBrandsCorrompido, dataSalesCorrompido){
    let dataBrands = require(dataBrandsCorrompido);
    let dataSales = require(dataSalesCorrompido);

    return [dataBrands, dataSales];
}
```

```
function main(){
    const allData = readJson('./broken_database1.json', './broken_database2.json');
    let dataSales = allData[0];
    let dataBrands = allData[1];
```

OBS: Essa função necessita do caminho dos arquivos corrompidos nos seus parametros

Função FixBrandAndVehicle

É aqui onde ocorre a primeira varredura, ela procura erros nos nomes das marcas e dos veículos através de um forEach. Nessa função utilizo o recurso “Replace()” somado ao parametro “g” que indica que esse caractere deve ser alterado globalmente na string.

```
function fixBrandAndVehicle(dataSales, dataBrands){
    dataBrands.forEach(element => {
        element.marca = element.marca.replace(/æ/g, 'a');
        element.marca = element.marca.replace(/ø/g, 'o');

    });

    dataSales.forEach(element => {
        element.nome = element.nome.replace(/æ/g, 'a');
        element.nome = element.nome.replace(/ø/g, 'o');

    });
}
```

Função fixSales

É onde ocorre a segunda varredura utilizando o forEach, destina-se a forçar a mudança dos dados armazenados em vendas para números inteiros, utilizo como forma de apoio o recurso “Number.ParseInt()” que realiza essa tarefa de transforma os dados em números inteiros.

```
function fixSales(dataSales){
    dataSales.forEach(element => {
        element.vendas = Number.parseInt(element.vendas);
    });
}
```

Função exportJson

É aqui onde se é feito a efetiva criação dos arquivos corrigidos, como parametros, essa função recebe as duas estruturas json ainda não transformadas em string, por isso utilizo o recurso “JSON.stringfy()” para que o arquivo possa ser criado mediante o modulo “fs” que significa FileSystem, esse modulo auxilia o código a concretamente realizar a criação dos arquivos usando o recurso “fs.writeFile()”,

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

```
function exportJson(dataSales, dataBrands){
    let dataSalesJsonString = JSON.stringify(dataSales);
    let dataBrandsJsonString = JSON.stringify(dataBrands);

    fs.writeFile('fixed_DataBase1.json' ,dataSalesJsonString, (err) =>{
        if(err) throw err;
        console.log('JSON Saved');
    });
    fs.writeFile('fixed_DataBase2.json' ,dataBrandsJsonString, (err) =>{
        if(err) throw err;
        console.log('JSON saved');
    });
};
```

OBS: Na imagem 1 cria-se a conexão com o modulo fs, fazendo com que, posteriormente, possa ser utilizado o recurso “fs,writeFile” que cria o arquivo corrigido na raiz do código fonte.

A função do “(err)” é retornar o tipo de erro, caso ocorra “If”, para o log da aplicação, isso ajuda em manutenções ou otimizações que possam ocorrer no código.

Partindo para o banco de dados

Após possuir os arquivos corrigidos, no SQL Online realizei o upload e a junção das tabelas a partir da referencia “id_marca”, para isso utilizei o recurso JOIN do SQL

```
CREATE TABLE tabelaCompleta AS
SELECT  b.Data, b.id_marca, a.nome_marca, b.Nome_Veiculo, b.Vendas, b.Valor_Veiculo
FROM fixed_DataBase1 b
JOIN fixed_DataBase2 a ON a.id_marca = b.id_marca;
```

Na conclusão deste código pude visualizar as duas tabelas devidamente mescladas e então realizar a exportação do arquivo .csv posteriormente utilizado na analise dos dados.