Javascript

Foram criadas duas constantes contendo o nome dos arquivos corrompidos.

A função de ler os arquivos recebe o nome do arquivo com parâmetro para fazer a busca. Como neste caso, os arquivos estavam na mesma pasta, então foi preciso colocar apenas um ./ antes do nome para a função require conseguir encontrar o .json corrompido. Caso seja passado um nome errado, esse erro é tratado e é avisado no terminal o erro. Caso não tenha erro, a função retorna um array em que cada posição é um objeto do arquivo json.

```
const b1 = 'broken_database_1.json'
const b2 = 'broken_database_2.json'

function lerArquivo(nome){
  try{
    var dado = require('./' + nome)
    return dado
  }catch(e){
    console.log('Arquivo não encontrado. Erro: ' + e.message)
    throw e
  }
}
```

A função de corrigir o nome recebe um array e um tipo como parâmetros (no caso do problema só tinha veículo e marca). O parâmetro tipo é apenas para que seja feito o tratamento no atributo correto dos arquivos.

\xF8 e \xF6 devem ser trocados por ø e æ, respectivamente. para isso é necessário um loop para percorrer as strings do objeto no array, e é necessário um outro laço de repetição dentro desse loop para percorrer os caracteres da string e fazer a troca.

No tipo veículo, já há a chamada para a função que corrige o número de vendas. Não há a chamada no tipo marca para que não seja criado um atributo vazio em cada objeto do array.

```
function CorrigirNome(tipo, data){
  if(tipo == 'veiculo'){
    data = CorrigirVendas(data)
    for(var i = 0; i < data.length; i++){
      for(var j = 0; j < data[i].nome.length; j++){
        data[i].nome = data[i].nome.replace('\xF8', 'o')
        data[i].nome = data[i].nome.replace('\xE6', 'a')
      }
    }
  }else if(tipo == 'marca'){
    for(var i in data){
      for(var j = 0; j < data.length; j++){
        data[i].marca = data[i].marca.replace('\xF8', 'o')
        data[i].marca = data[i].marca.replace('\xE6', 'a')
      }
    }
  }
  return data
}
```

A função de corrigir as vendas apenas troca o atributo de vendas para número. Caso haja um erro,é impresso no terminal em qual objeto está o erro.

```
function CorrigirVendas(data){
  for(var i in data){
    try{
      data[i].vendas = Number(data[i].vendas)
    }catch(e){
      console.log('A string está no formato errado. Objeto: ' + data[i])
      throw e
    }
  }
  return data
}
```

Por fim é exportado o arquivo com o array corrigido, como é o mesmo nome do .json corrompido, então os dados são sobrescritos nele.

A função de exportar é chamada e o retorno das outras funções são utilizados como parâmetro da função de exportação.

```
function exportaArquivo(nome, data){

  var fs = require('fs');

  const jsonData = JSON.stringify(data)

  fs.writeFileSync(nome, jsonData, 'utf-8', err => {
    if(err) throw err
  })
}


exportaArquivo(b2, CorrigirNome('marca', lerArquivo(b2)))
exportaArquivo(b1, CorrigirNome('veiculo', lerArquivo(b1)))
```

SQL

Os nomes das colunas foram renomeados na busca para facilitar a leitura dos dados. As tabelas foram unidas a partir da correspondência do atributo id_marca_ da tabela broken_database_1.json com o atributo id_marca de broken_database_2.json. Os dados tabela resultado dessa união são organizados pelo id_marca_ e nome, nesta ordem de prioridade.

SELECT broken_database_1.c1 AS data_venda,
broken_database_1.c2 AS id_marca,
broken_database_1.c3 AS quantidade,
broken_database_1.c4 AS valor,
broken_database_1.c5 AS nome_modelo,
broken_database_2.c2 AS nome_marca
FROM broken_database_1 INNER JOIN broken_database_2 ON broken_database_1.c2 = broken_database_2.c1
ORDER BY broken_database_2.C1, broken_database_1.c5