DOCUMENTAÇÃO
TRABALHO REALIZADO

HYDEOSHY KALED RAMOS DE OLIVEIRA

SOROCABA – 2023

INTRODUÇÃO

Este documento tem a finalidade de apresentar o raciocínio e a explicação que obtivemos acerca do desenvolvimento do projeto apresentado no processo seletivo da Rocky.Monks.

Será apresentado a explicação do código Sql que tratamos para adquirir o nosso relatório e também o código JS utilizado para arrumarmos o broken_database.JSON 1 e 2.

Código JavaScript

Primeiro começamos adicionando a linha de código abaixo que é uma forma de importarmos o módulo ‘fs’ que é uma biblioteca nativa do Node.js.

```
const fs = require('fs'); // Biblioteca node
```

Função responsável pela leitura do arquivo JSON

```
function readFile(filename) {

  let arquiJson = fs.readFileSync(filename);

  return JSON.parse(arquiJson);
```

Essa função é responsável por ler o arquivo JSON, em resumo ela faz a leitura do conteúdo especificado no parâmetro da função(‘filename’), analisa o conteúdo do arquivo JSON e retorna um objeto javascript.

Função responsável por fazer a troca dos caracteres errados nos arquivos JASON

```
function fixedCharacters(database, key) {

Object.keys(database).forEach(element=>{database[eleme
nt][key]=database[element][key].replace(/ø/g,'o').repl
ace(/æ/g, 'a');

  })

  return database;
```

Utilizamos a linha de código acima para podermos fazer a mudança de caracteres que permaneciam com letras trocadas dentro das chaves dos arquivos JSON para conseguirmos concluir este feito definimos uma função

com parâmetros ‘database’ e ‘key’ onde um deles será o banco de dados enquanto o outro é a chave (key) que identifica qual o campo que será utilizado.

Seguindo neste bloco de códigos foi utilizado o método ‘forEach’ que será responsável por permitir executar um ação para cada elemento do código, e realizando assim com o método ‘replace’ as seguintes alterações de caracteres no código: (**ø** por **'o'** e **æ** por **'a')**.

E por fim retorna o banco de dados com as alterações realizadas.

Função responsável pela conversão de valor String para number

```
function convertSells(dataBase, key) {

Object.keys(dataBase).forEach(element=>{dataBase[eleme
nt][key] = parseInt(dataBase[element][key])

  }) return dataBase;

}
```

Nesta função foi definidos os mesmos padrões da função anterior pois ainda estamos trabalhando dentro dos elementos e seus respectivos campos deste modo foi utilizado novamente o método forEach cujo sua funcionalidade já está descrita acima o que mudou nesta função e que ela terá como funcionalidade e objetivo converter o valor da (key) em number que no caso foi selecionado para fazer as alterações das quantidades de vendas que são números porém estavam em formatos de string.

E por fim, assim como a outra retorna o valor do banco de dados devidamente corrigido com os valores em seu formato real.

Função responsável por exportar os novos arquivos JSON já corrigidos

```
function writeFiles(filename, data) {

fs.writeFileSync(filename, JSON.stringify(data));
```

Esta função é responsável por escrever dados em formato JSON em um arquivo com o nome especificado pelo parâmetro ‘filename’, que para realizar este feito está sendo utilizado método ‘fs.writeFileSync’, onde é o responsável por gravar os dados em um arquivo.

Já no segundo parâmetro é o conteúdo e a extensão que é obtida pela conversão do objeto (data) para um string JASON realizado pelo método JSON.stringify.

Variáveis complementares

```
let dataBase = readFile('broken_database_1.json');

dataBase = fixedCharacters(dataBase, 'nome');

dataBase = convertSells(dataBase, 'vendas');

 writeFiles('fixxed_database_1.json', dataBase);


 let dataBase2 = readFile('broken_database_2.json');

 dataBase2 = fixedCharacters(dataBase2, 'marca');

 writeFiles('fixxed_database_2.json', dataBase2);
```

Para finalizarmos o nosso código e conseguirmos gerar os dois arquivos broken_database devidamente corrigidos, foi preciso fazer o callback com variáveis que fizeram os chamados das funções concluídas acima.

Iniciamos as chamadas das funções para alterações no broken_database1, começamos pela função **readFile** onde foi definido o arquivo que seria lido e feito as primeiras alterações, onde ficou salvo na variável database.

Após fizemos a chamada da função **fixedCharacters** onde foi definido o database que armazena o arquivo JSON e a seguir foi colocado a campo(**key)** onde será realizado a alteração determinada na função.

Depois foi necessário chamarmos a função **convertSells** que é a responsável por alterar os valores da **key/campo** selecionado que neste caso foi **'vendas'** onde os elementos não estavam com os seus respectivos valores…

E por fim porém não menos importante a função **writeFiles** responsável por criar um novo arquivo por eles está abaixo das demais funções ele já vem com todo retificado de acordo com as alterações determinadas pelas funções e para finalizar converte o arquivo para string JSON, recebendo o nome de '**fixxed_database_1.json**' nome que foi incluído dentro do parâmetro **filename**. Após isso finalizamos as alterações no broken_database_1 e seguimos para o 2.

Para deixarmos o arquivo broken_database_2 corrigido foi necessário criar uma nova variável (dataBase2) responsável por armazenar o arquivo lido pela função **readFile,** após isto chamamos a função **fixedCharacters** para

fazer as alterações das caracteres que estão descritas na função e documentário e para este feito necessário definir o campo de alteração que no **broken_database2** foi **‘marca’**. Para finalizarmos, fizemos o chamado da função **writeFiles** que foi a encarregada de gerar o arquivo retificado e transformado em string JSON pelo método **stringfy.JSON**.

Resumo

Em resumo as nossas linhas de códigos realiza a leitura do arquivo JSON retornado em objeto JS, dando a possibilidade de acessar e alterar o arquivo em seguida fazer a troca de caracteres dos campos alterados e converter os valores de string para numbers presentes no arquivo e por fim retornar um arquivo totalmente retificado e com as alterações requisitadas nas funções.

Tratamentos feitos no código para evitar bugs

Neste código foi desenvolvido um pensamento na hora da criação das funções que foi não determinar os nomes dos arquivos, campos, elementos e objetos que foram acessados dentro das função para que não haja nenhum conflito nos dados, por haver dois arquivos JSON e como estamos usando a mesma função para fazer alterações em dois arquivos deixamos para determinar os campos e arquivos lidos no momento que fossemos chamar as funções após finalizadas. Fazendo foi determinado um código com menor complexidade visual, ao decorrer do desenvolvimento do mesmo foi feita a formatação do texto para que fosse possível identificar erros de digitação e pontuação dentro do código.

Em prol deste ato não foi necessário o desenvolvimento de outras funções para pudéssemos fazer a alteração em um segundo arquivo, o que precisamos fazer foi apenas chamá-las dentro de cada arquivo e chave diferentes.

Código SQL

Qual marca teve o maior número de vendas?

Agora falaremos um pouco sobre o código Sql utilizado para realizar as atividades solicitadas. Para atendermos a solicitação de qual marca teve o maior o volume de vendas utilizamos o código abaixo:

```
SELECT marca, COUNT (marca) AS qtd_vendido
FROM fixxed_database_1 JOIN fixxed_database_2 ON
fixxed_database_1.id_marca_ = fixxed_database_2.id_marca GROUP BY
fixxed_database_2.marca ORDER BY qtd_vendido DESC
```

Neste código utilizamos o **SELECT** para indicar qual coluna estará retornando depois utilizamos o método **COUNT** para contar quantas vezes apareceu a marca de cada veículos junto com este usamos o **AS** que irá criar uma nova coluna neste arquivo que está sendo gerado para que na hora da finalização do código ele nos retorna os números de vendas por marca em seguida utilizamos o método **FROM** para especificar a tabela que será utilizado neste projeto e em seguida o método **JOIN** que nos permite juntar 2 ou mais tabelas em um linha de códigos para fazer a junção dos mesmos, depois usamos o **ON** para indicar quais colunas das tabelas será feita a junção que neste caso foi os **id** de cada tabela, como ambos possuem as mesma numeração sendo que um contém o id de veículos e o outro de suas respectivas marcas fazendo a junção conseguimos chegar o resultado desejado, e para finalizar agrupamos a coluna **'marca'** da tabela database_2 com o método **GROUP BY** e **ORDER BY** para ordenarmos as qtd_vendido por ordem decrescente ou seja do maior para o menor.

Qual veículo gerou a maior e a menor receita?

```
SELECT nome, SUM (valor_do_veiculo*vendas) AS valor_gerado
FROM fixxed_database_1 GROUP BY nome ORDER BY
valor_gerado DESC
```

O código acima foi feito deste jeito para obter a soma das vendas pelo valor do veículo utilizando o método **SUM** o **AS** para criar a nova tabela que será gerada os valores da somas o **FROM** para juntar a tabela necessária para esta tarefa e para finalizar agrupamos e ordenamos com **GROUP BY** e **ORDER BY** e colocamos tudo em ordem decrescente.

Qual a média de vendas do ano por marca?

```
SELECT marca,AVG(fixxed_database_1.vendas) AS
    media_vendas FROM fixxed_database_1 JOIN
fixxed_database_2 ON fixxed_database_1.id_marca_=
fixxed_database_2.id_marca GROUP BY marca;
```

Neste código foi necessário utilizar o **AVG** que será o responsável por fazer todo o cálculo da média de vendas do ano e seguida foi necessário fazer a junção as das duas tabelas em referência para que pudéssemos fazer o agrupamento da coluna marca que está presente em tabelas de diferente que a coluna de vendas.

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

```
SELECT marca, COUNT (marca) AS qtd_vendido,SUM
(valor_do_veiculo*vendas) AS  valor_gerado FROM

fixxed_database_1 JOIN fixxed_database_2 ON
```

fixxed_database_1.id_marca_ = fixxed_database_2.id_marca **GROUP BY** fixxed_database_2.marca **ORDER BY** valor_gerado **DESC**

Para esta atividade em questão foi necessário realizar a contagem de quantas vendas cada marca obteve para isso utilizamos **COUNT** que ficou encarregado desta função a seguir foi preciso fazer a multiplicação do valor do veículo com o número de vendas dos veículos para descobrimos a maior receita gerada e também a menor gerada das vendas, dando seguimento fizemos a junção das duas tabelas em referência e o agrupamento pela '**marca'** e sua ordenação pelo **'valor_gerado'** que foi a tabela criada para representar os valores da receita de vendas dos veículos, os quais foram ordenados em ordem decrescente.

Existe alguma relação entre os veículos mais vendidos?

SELECT data, nome, marca, id_marca, id_marca _,valor_do_veiculo, vendas **FROM** fixxed_database_1 **JOIN**fixxed_database_2 **ON** fixxed_database_1.id_marca_ = fixxed_database_2.id_marca;

Para que fosse possível fazer a localização das relações dos veículos mais vendidos fizemos a junção de todas as tabelas das colunas para que tivéssemos uma melhor visualização dos dados para fazer isso apenas usamos o **SELECT** para selecionar as tabelas que nós vamos usar e fizemos a junção delas com o **FROM** e **JOIN** e depois usamos o **ON** para finalizar a junção deles pelo id.