media .monks

Processo seletivo – Tech (Estágio em Web Analytics)

Nome: Ryan Furtado de Almeida CPF: 493.122.468-79

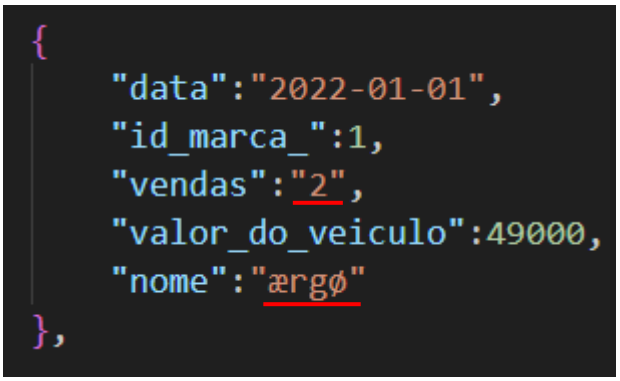
Explicação dos códigos

- Análise das issues:

Para esse desafio recebi dois arquivos *.json*, representando uma resposta do banco de dados, onde alguns valores estão corrompidos, sendo eles:

- Subtituição dos caracteres "a" por "æ" e "o" por "ø";

- Número de vendas representado como string.



```
{
    "data":"2022-01-01",
    "id_marca_":1,
    "vendas":"2",
    "valor_do_veiculo":49000,
    "nome":"ærgø"
},
```

- Primeira fase: Descorromper base de dados

Utilizei Javascript para fazer um simples script para ler a base, arrumar todas a issues e exportar para um novo arquivo Json.

```
function main(){
    let vendasJson = lerJson('Data/Broken/broken_database_1.json');
    let marcasJson = lerJson('Data/Broken/broken_database_2.json');

    vendasJson.forEach((venda, index) => {
        venda.id = index+1;
        venda.nome = substituirLetra(venda.nome,'æ', 'a');
        venda.nome = substituirLetra(venda.nome,'ø', 'o');

        venda.vendas = parseFloat(venda.vendas);
    });

    marcasJson.forEach(marcaObj => {
        marcaObj.marca = substituirLetra(marcaObj.marca,'æ', 'a');
        marcaObj.marca = substituirLetra(marcaObj.marca,'ø', 'o');
    });

    salvarJson('vendas_database', vendasJson);
    salvarJson('marcas_database', marcasJson);
}
```

Parte 1: Leitura dos json

Parte 2: Loop em todos valores da base de dados, e solução das issues.

Parte 3: Exportar os json

Dividi o código em pequenas funções para melhor reusabilidade e compreenção. Na proxima folha explico cada função individualmente.

- Parte 1:

```
let vendasJson = lerJson('Data/Broken/broken_database_1.json');
let marcasJson = lerJson('Data/Broken/broken_database_2.json');
```

Crio uma varíavel para cada base de dados, e chamo a função "lerJson()" passando o caminho até o arquivo, essa função está descrita abaixo;

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

function lerJson(filePath){
    try{
        let objectJson = fs.readFileSync(filePath, 'utf-8');
        objectJson = JSON.parse(objectJson);

        return objectJson;
    }
    catch(err){
        console.error(err);
    }

}
```

Antes da função, eu preciso do módulo fileSearch do node, poderia adicionar utilizando " import * as fs from 'node:fs' ", mas como eu iria usar o proprio node pra rodar o script, utilizei " require('fs') " mesmo.

Agora vamos para a função, como já dito antes, ela recebe o caminho até o arquivo, e com isso utilizo o método readFileSync() do fs e armazeno a resposta em uma varíavel.

Para essa resposta uso um método nativo do javascript para converter esse Json para um objeto, e por fim retorno o mesmo.

OBS: Nota-se que tudo que expliquei está envolvido por um "try, catch" que é uma notação para previnir alguns erros, por exemplo: Se o caminho do arquivo estiver errado, ele não vai encontrar, e ao inves de rodar normalmente, ele vai ir para o catch e assim mostrar no console o que houve de erro

- Parte 2:

Para ambos objetos, utilizo a estrutura de repetição " forEach() " para percorrer cada item da base de dados.

```
vendasJson.forEach((venda, index) => {
    venda.id = index+1;
    venda.nome = substituirLetra(venda.nome,'æ', 'a');
    venda.nome = substituirLetra(venda.nome,'ø', 'o');

    venda.vendas = parseFloat(venda.vendas);
});

marcasJson.forEach(marcaObj => {
    marcaObj.marca = substituirLetra(marcaObj.marca,'æ', 'a');
    marcaObj.marca = substituirLetra(marcaObj.marca,'ø', 'o');
});
```

Primeiro para a base de dados que contem os registros de vendas, eu notei uma outra issue, que não havia sido citada, e me senti com a liberdade de aplicar essa melhoria. Não havia um identificador para os registros de venda, e é interessante ter para registro, criei uma propriedade "id" que recebe o valor do index + 1 (para evitar o id 0).

Agora para fazer as substituições, criei uma outra função "substituirLetra()" que recebe a string, qual letra quero trocar, e qual o novo caractere. Vamos ver como funciona abaixo:

```
function substituirLetra(string, oldChar, newChar){
    return string.replace(new RegExp(oldChar, 'g'), newChar);
}
```

É uma função simples que subtitui os valores passados, utilisei o método de string "replace()" ele só pede o antigo valor, e o novo, mas tem um problema, ele substituie a primeira interação só.

Para resolver isso, vc pode passar o valor antigo como uma expressão regular, com o parametro g no final, assim ele vai interar sob todas aparições do valor passado. Feito isso, só precisei retornar o novo valor.

Voltando para a main, para o problema de número estar como strint, só utilizei um "parseFloat()" que é uma função nativa para fazer essa conversão.

```
venda.vendas = parseFloat(venda.vendas);
```

Para a segunda base de dados, só utilizei a função "substituirLetra()", já explicada acima.

- Parte 3:

```
salvarJson('vendas_database', vendasJson);
salvarJson('marcas_database', marcasJson);
```

Para finalizar, só precisava salvar os objetos em formato json, então fiz outra função com esse objetivo, "salvarJson()", que recebe o nome do arquivo desejado, e qual objeto deseja salvar.

```
function salvarJson(fileName, json){

    try{
        json = JSON.stringify(json, null, 2);
        fs.writeFileSync(`Data/${fileName}.json`, json, 'utf-8');
    }
    catch(err){
        console.error(err);
    }
}
```

Assim, para realizar essa parte, utilizei a classe JSON nativa do javaScript, com o metodo "stringify()" que transforma o objeto em uma string JSON.

Agora utilizando novamento o fs, mas dessa vez pra escrever um arquivo, com o metodo "writeFileSync()".

E da mesma forma que para ler um outro arquivo, envolvi tudo em um "try, catch" para avisar se algum erro acontecer no processo.

- Segunda fase: Unir os dados em uma tabela única.

Foi utilizado SQL para manipular os dados e criar uma tabela com as informações necessarias.

Primeiramente, precisei criar a nova tabela, com as colunas que julguei uteis para o meu relátorio.

```
CREATE TABLE IF NOT EXISTS relatorio_vendas (
    id SERIAL PRIMARY KEY,
    data DATE,
    marca_id INT,
    marca_nome VARCHAR(255),
    quant_vendas INT,
    valor_venda INT,
    modelo_carro VARCHAR(255)
);
```

```sql
INSERT INTO relatorio_vendas (id, data,
    marca_id, marca_nome, quant_vendas,
    valor_venda,modelo_carro)
SELECT
    v.id,
    v.data,
    v.marca_id,
    m.Nome AS marca_nome,
    v.quant_vendas,
    v.valor AS valor_venda,
    v.modelo AS modelo_carro
FROM Vendas v
JOIN Marcas m ON v.marca_id = m.id;
```

Com a tabela criada, só preciso puxar os valores das tabelas existentes, começando com um insert into, e colocando os valores que desejo adicionar, na ordem certa.

Assim com o select passo os valores das outras tabelas, atenção, quando o nome for for diferente, é necessario utilizar "AS" para marcar qual o nome na nova tabela.

Percebe-se que usei um valor simplificado, para os nomes, pois ao dizer de onde os dados vem "FROM" passei um "*apelido*" para o SQL.

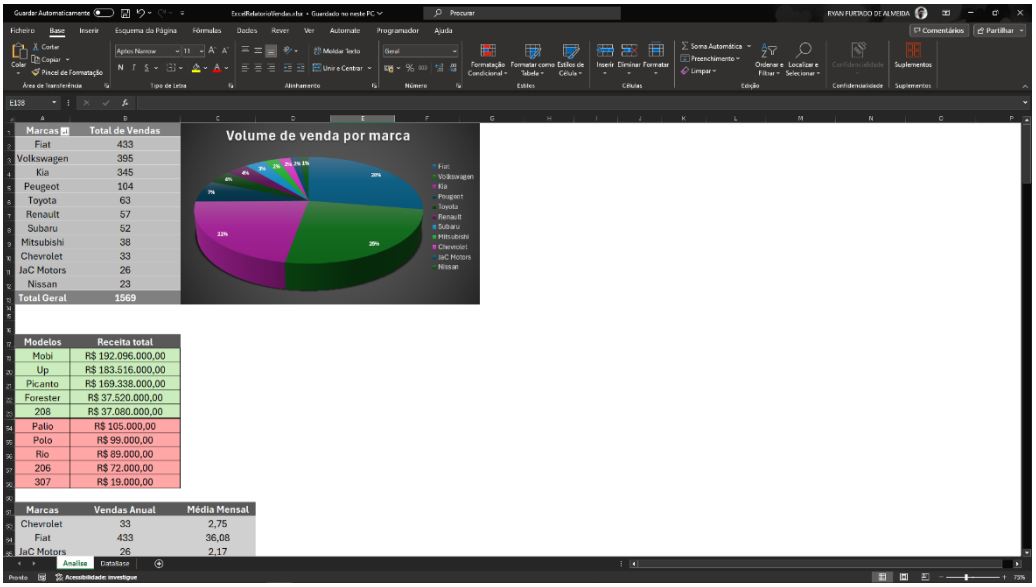
Mas ainda estaria faltando o nome da marca, então eu inclui "JOIN" a tabela Marcas nessa nova tabela, e para saber qual valor vai ser inserido, verifiquei quando "ON" o "id" coincide com o "marca_id" da tabela de vendas.

Assim já está feito tenho uma nova tabela com todos os valores necessarios para realizar os relatorios.

OBS: Talvez não precisasse mais do Id da marca, tanto que não utilizei depois, mas escolhi manter na tabela, por que poderia ser uma maneira mais facíl de manipular dependendo do que queria analisar.

- Terceira fase: Análise final dos dados.

Para finalizar como ficou em aberto como fazer a analise, eu decidi utilizar o Excel com tabelas dinamicas, pois acho uma forma rápida e bem visual para colocar no relatorio depois.



Marcas	Total de Vendas
Fiat	433
Volkswagen	395
Kia	345
Peugeot	104
Toyota	63
Renault	57
Subaru	52
Mitsubishi	38
Chevrolet	33
JaC Motors	26
Nissan	23
Total Geral	1569

Modelos	Receita total
Mobi	R$ 192.096.000,00
Up	R$ 183.516.000,00
Picanto	R$ 169.338.000,00
Forester	R$ 37.520.000,00
208	R$ 37.080.000,00
Palio	R$ 105.000,00
Polo	R$ 99.000,00
Rio	R$ 89.000,00
206	R$ 72.000,00
307	R$ 19.000,00

Marcas	Vendas Anual	Média Mensal
Chevrolet	33	2,75
Fiat	433	36,08
JaC Motors	26	2,17