

Processo seletivo tech

Explicação dos códigos referente as funções e o SQL

Sorocaba

2024

Sumário

Function.js	4
Function LerDatabase	4
Function CorrigirValor	5
Function ConverterParaInt	6
Function ExportarArquivoJson	6
Código SQL	7
Consultas (Selects) para o relatório das vendas	8
2. Qual veículo gerou a maior e menor receita?	8
3. Qual a média de vendas do ano por marca?	9
4. Quais marcas geraram uma receita maior com número menor de vendas? ..	10
5. Existe alguma relação entre os veículos mais vendidos?	10

Function.js

Chama a função LerDatabase para ler os arquivos .json do **broken_database_1** e **broken_database_2**.

```
const fs = require('fs');
const { database1, database2 } = LerDatabase('broken_database_1.json',
'broken_database_2.json');
```

Chama a função para corrigir os nomes de marca e veículpp do arquivo 1 e 2

```
const dadosCorrigidos1 = CorrigirValor(database1); // Chama a função para
corrigir os nomes de marca e veículo do arquivo 1
const dadosCorrigidos2 = CorrigirValor(database2); // Chama a função para
corrigir os nomes de marca e veículo do arquivo 2

const dadoCorretoVenda1 = ConverterParaInt(dadosCorrigidos1);
const dadoCorretoVenda2 = ConverterParaInt(dadosCorrigidos2);

ExportarArquivoJson(dadoCorretoVenda1, 'database_certa_1.json');
ExportarArquivoJson(dadoCorretoVenda2, 'database_certa_2.json');
```

Function LerDatabase

A função tem como objetivo ler os arquivos do **broken_database_1** e **broken_database_2**, utilizando o “fs.readFileSync”. Após a leitura dos arquivos o conteúdo dos arquivos são convertidos em objetos.

```
function LerDatabase(broken_database_1, broken_database_2) {
  try {
    const database1 = JSON.parse(fs.readFileSync(broken_database_1,
'utf8'));
    const database2 = JSON.parse(fs.readFileSync(broken_database_2,
'utf8'));
    return { database1, database2 };
  } catch (erro) {
    console.error('Erro ao ler os arquivos:', erro);
  }
}
```

```
    return { database1: [], database2: [] };  
  }  
}
```

Depois da leitura, a função retornará dois objetos (database 1 e database 2) contendo os dados dos arquivos **broken_database_1** e **broken_database_2**. Porém, em casos de erro, ele retornará valores nulos e irá imprimir a mensagem de erro.

Function CorrigirValor

A função corrige os nomes da marca e veículo que foram corrompidos. Ela recebe um array de objetos dos dados corrompidos na variável **data** e depois retorna um array com os dados corrigidos. Em caso de erro, a função retornará um array vazio e imprimirá a mensagem de erro.

```
function CorrigirValor(data) {  
  try {  
    const valor_corrigido = data.map(entry => ({  
      ...entry,  
      nome: entry.nome ? entry.nome  
        .replace(/æ/g, 'a')  
        .replace(/ø/g, 'o') : entry.nome,  
      marca: entry.marca ? entry.marca  
        .replace(/æ/g, 'a')  
        .replace(/ø/g, 'o') : entry.marca  
    }));  
    return valor_corrigido;  
  } catch (erro) {  
    console.error('Erro ao corrigir os nomes dos campos:', erro);  
    return [];  
  }  
}
```

Function ConverterParaInt

A função recebe o array de objetos do **data**, depois é feita a verificação se a variável é uma string ou não, se caso for uma string, ela converte para inteiro usando o **ParseInt** e depois retorna o **array**. Porém caso de erro, a função retornará um array vazio e imprimirá a mensagem de erro.

```
function ConverterParaInt(data) {
  try {
    const alterar_tipo = data.map(entry => ({
      ...entry,
      vendas: typeof entry.vendas === 'string' ? parseInt(entry.vendas)
: entry.vendas
    }));
    return alterar_tipo;
  } catch (erro) {
    console.error('Erro ao alterar formato de vendas para inteiro', erro);
    return [];
  }
}
```

Function ExportarArquivoJson

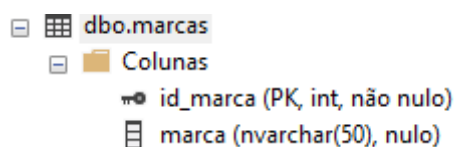
Por fim, a função converte os dados para json por meio do **JSON.stringify**, depois utilizando o **fs.writeFileSync** para escrever o dados do json especificado por “arquivo”. Caso a exportação funcionar, irá imprimir uma mensagem dizendo que os dados foram exportados, caso ao contrário ele imprimirá uma mensagem de erro.

```
function ExportarArquivoJson(data, arquivo) {
  try {
    const json = JSON.stringify(data, null, 2);
    fs.writeFileSync(arquivo, json);
    console.log(`Dados corrigidos exportados para ${arquivo}`);
  } catch (erro) {
    console.error('Erro ao exportar', erro);
  }
}
```

Código SQL

Para exportar os arquivos .json para o banco de dados, foram criadas 2 tabelas dentro do banco de dados. A tabela Marcas que corresponde ao arquivo **database_certa_2.json** e a tabela de Vendas_veículos que contém os dados do arquivo **database_certa_1.json**.

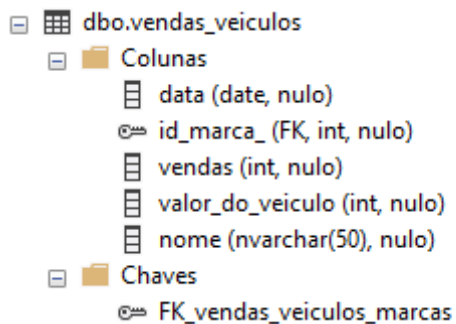
Tabela dbo.marcas



dbo.marcas
Colunas
id_marca (PK, int, não nulo)
marca (nvarchar(50), nulo)

Tabela dbo.vendas_veículos

Nessa tabela é contido a chave estrangeira da tabela marcas.



dbo.vendas_veiculos
Colunas
data (date, nulo)
id_marca_ (FK, int, nulo)
vendas (int, nulo)
valor_do_veiculo (int, nulo)
nome (nvarchar(50), nulo)
Chaves
FK_vendas_veiculos_marcas

Consultas (Selects) para o relatório das vendas

1. Qual marca teve o maior volume de vendas?

Retorna a lista dos marcas que mais venderam na concessionária.

```
select vv.id_marca_, m.marca, sum(vv.vendas) AS total_de_vendas
from vendas_veiculos vv
inner join marcas m on vv.id_marca_ = m.id_marca
group by vv.id_marca_, m.marca
order by total_de_vendas desc;
```

Resultado da consulta:

	id_marca_	marca	total_de_vendas
1	1	Fiat	433
2	2	Volkswagen	395
3	3	Kia	345
4	4	Peugeot	104
5	5	Toyota	63
6	11	Renault	57
7	8	Subaru	52
8	7	Mitsubishi	38
9	9	Chevrolet	33
10	10	JaC Motors	26
11	6	Nissan	23

2. Qual veículo gerou a maior e menor receita?

Retornará o veículo que gerou a maior receita:

```
select top 1 vv.nome, m.marca,
sum(vv.vendas * vv.valor_do_veiculo) as receita_total
from vendas_veiculos vv
INNER JOIN marcas m on vv.id_marca_ = m.id_marca
group by vv.nome, m.marca
order by receita_total desc;
```

Resultado da consulta:

.00 %

Resultados		Mensagens	
	nome	marca	receita_total
1	Mobi	Fiat	14747000

Retornará o veículo que gerou a menor receita:

```
select top 1 vv.nome, m.marca,  
            sum(vv.vendas * vv.valor_do_veiculo) as receita_total  
from vendas_veiculos vv  
INNER JOIN marcas m on vv.id_marca_ = m.id_marca  
group by vv.nome, m.marca  
order by receita_total asc;
```

Resultado da consulta:

.00 %

Resultados		Mensagens	
	nome	marca	receita_total
1	307	Peugeot	19000

3. Qual a média de vendas do ano por marca?

Retorna a média anual das vendas por marca.

```
select year(vv.data) as ano, m.marca,  
        avg(vv.vendas) as media_anual_vendas  
from vendas_veiculos vv  
inner join marcas m ON vv.id_marca_ = m.id_marca  
group by YEAR(vv.data), m.marca  
order by media_anual_vendas desc;
```

Resultado da consulta:

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

Esse select gera uma tabela das marcas que geraram a maior receita com o número menor de vendas.

```
select m.marca,
       sum(vv.vendas * vv.valor_do_veiculo) as receita_total,
       sum(vv.vendas) as total_vendas
from vendas_veiculos vv
INNER JOIN marcas m on vv.id_marca_ = m.id_marca
group by m.marca
order by receita_total desc, total_vendas asc;
```

Resultado:

	marca	receita_total	total_vendas
1	Subaru	16030000	52
2	Volkswagen	15540000	395
3	Fiat	15447000	433
4	Kia	13586000	345
5	Peugeot	8086000	104
6	Mitsubishi	6374000	38
7	Toyota	5493000	63
8	Renault	2403000	57
9	JaC Motors	1430000	26
10	Chevrolet	1418400	33
11	Nissan	601000	23

5.Existe alguma relação entre os veículos mais vendidos?

O select abaixo trás a lista e o preço médio dos veículos mais vendidos.

```
select id_marca_, nome,
       sum(vendas) AS quantidade_de_vendas,
       avg(valor_do_veiculo) as valor_medio_do_veiculo
from vendas_veiculos
group by id_marca_, nome
order by quantidade_de_vendas desc;
```

Resultado da consulta:

	id_marca_	nome	quantidade_de_vendas	valor_medio_do_veiculo
1	1	Mobi	414	38666
2	2	Up	373	41000
3	3	Pic...	338	41750
4	4	208	90	82400
5	5	Cor...	40	108000
6	9	onix	33	43355
7	11	Clio	30	30600
8	8	For...	28	335000
9	6	Mar...	23	26428
10	5	Yaris	23	53750
11	8	WRX	15	250000
12	7	Lan...	14	72000
13	10	J2	14	12111
14	7	Paj...	13	220000
15	11	Du...	11	47500
16	7	L200	10	270000
17	2	Gol	9	32666
18	11	Sa...	8	38000
19	1	Uno	8	22750
20	4	2008	7	85000
21	4	206	6	12000
22	3	Cer...	6	43000
23	10	E-J...	6	145000
24	2	Sa...	6	60000
25	10	IE	5	25000

