Explicação dos códigos.

Funções para correção do banco de dados em Java Script:

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

Inicialmente foi importado o modulo fs(File System) para leitura e manipulação do arquivo JSON.

1. a) Ler os arquivos JSON:

```
async function lerArquivosJSON() {
    try {
        const conteudoArquivo1 = await
fs.promises.readFile('broken_database_1.json', 'utf-8');
        const conteudoArquivo2 = await
fs.promises.readFile('broken_database_2.json', 'utf-8');

        const dados1 = JSON.parse(conteudoArquivo1);
        const dados2 = JSON.parse(conteudoArquivo2);

        return { dados1, dados2 };
    } catch (error) {
        console.error('Erro ao ler os arquivos JSON:', error);
        throw error;
    }
}
```

Para realizar a leitura do arquivo json utilizei uma função assíncrona nomeada “lerArquivosJSON” , dentro da função, utilizei o bloco “try-catch”, onde será testado o código escrito no try, caso ocorra algum erro, o teste é interrompido e o catch captura o erro.

Nas constantes conteudoArquivo1 e conteudoArquivo2, trabalhei com promises, utilizando o file system para ler os arquivos, e armazenando o arquivo em suas respectivas constantes.

O JSON.parse foi utilizado para conversão do arquivo armazenado para um objeto JavaScript.

E o return para retornar o objeto com os dados.

1.b) Corrigir nomes de marcas e veículos

```
async function corrigirNomes() {
    try {
        const registros = await lerArquivosJSON();
        if (registros.dados1) {
            for (let i = 0; i < registros.dados1.length; i++) {
                if (registros.dados1[i].nome) {
                    registros.dados1[i].nome =
registros.dados1[i].nome.replace(/æ/g, 'a').replace(/ø/g, 'o');
                }
                if (registros.dados1[i].marca) {
```

```
                    registros.dados1[i].marca =
registros.dados1[i].marca.replace(/æ/g, 'a').replace(/ø/g, 'o');
                }
            }
        }
        if (registros.dados2) {
            for (let i = 0; i < registros.dados2.length; i++) {
                if (registros.dados2[i].nome) {
                    registros.dados2[i].nome =
registros.dados2[i].nome.replace(/æ/g, 'a').replace(/ø/g, 'o');
                }
                if (registros.dados2[i].marca) {
                    registros.dados2[i].marca =
registros.dados2[i].marca.replace(/æ/g, 'a').replace(/ø/g, 'o');
                }
            }
        }

        return registros;
    } catch (error) {
        console.error('Erro ao corrigir os nomes:', error);
        throw error;
    }
}
```

Nessa função assíncrona nomeada corrigirNomes, foi criada uma constante “registros” para obter os dados da função lerArquivosJSON.

Dentro do bloco try-catch fiz uma verificação para realizar a troca com (replace) dos nomes e marcas que estavam quebrados.

1.c) Corrigir vendas

```
async function corrigirVendas() {
    try {
        const registros = await corrigirNomes();

        for (let i = 0; i < registros.dados1.length; i++) {
            if (typeof registros.dados1[i].vendas === 'string') {
                registros.dados1[i].vendas =
parseFloat(registros.dados1[i].vendas);
            }
        }

        for (let i = 0; i < registros.dados2.length; i++) {
            if (typeof registros.dados2[i].vendas === 'string') {
                registros.dados2[i].vendas =
parseFloat(registros.dados2[i].vendas);
```

```
            }
        }

        return registros;
    } catch (error) {
        console.error('Erro ao corrigir as vendas:', error);
        throw error;
    }
}
```

Nessa função assíncrona nomeada corrigirVendas, foi criada uma constante “registros” para obter os registros corrigidos da função corrigirNomes.

Dentro do bloco try-catch fiz uma verificação para identificar o tipo dos dados das vendas, se o tipo fosse string seria transformado em numero através do parseFloat e retornar os registros corrigidos.

1.d) Exportar um arquivo JSON com o banco corrigido.

```
async function exportarArquivosCorrigidos() {
    try {
        const registrosCorrigidos = await corrigirVendas();

        const dadosJSON1 = JSON.stringify(registrosCorrigidos.dados1,
null, 2);
        const dadosJSON2 = JSON.stringify(registrosCorrigidos.dados2,
null, 2);

        await fs.promises.writeFile('fixed_database_1.json', dadosJSON1,
'utf-8');
        await fs.promises.writeFile('fixed_database_2.json', dadosJSON2,
'utf-8');

        console.log('Arquivos exportados com sucesso:
fixed_database_1.json e fixed_database_2.json');
    } catch (error) {
        console.error('Erro ao exportar os arquivos corrigidos:', error);
    }
}
```

Nessa função assíncrona nomeada exportarArquivosCorrigidos, dentro do bloco try-catch criei uma constante para receber os dados da função corrigirVendas, nomeado registrosCorrigidos, usei JSON.stringify para converter os sados corrigidos em strings, usei o parâmetro 2 para tornar mais legível a leitura dos dados, usei a função writeFile da biblioteca fs promises e usei o await para que esperasse as operações assíncronas serem concluídas antes de das prosseguimento, usei o catch para imprimir o erro, e no final chamei a função para que a exportação fosse realizada, e s arquivos corrigidos foram nomeados: “fixed_database_1” e “fixed_database_2”.

Código SQL utilizado para tratamento dos dados

Para unir os dados dos bancos de dados fornecidos em uma tabela foi utilizado o seguindo cofigo dentro da ferramenta SQLite Online.

```
CREATE TABLE unify_table AS
SELECT
    fixed_database_1.data,
    fixed_database_1.vendas,
    fixed_database_1.valor_do_veiculo,
    fixed_database_1.nome,
    fixed_database_2.id_marca,
    fixed_database_2.marca
FROM
    fixed_database_1
JOIN
    fixed_database_2 ON fixed_database_1.id_marca_ =
fixed_database_2.id_marca;
```

Para criar a tabela unificada selecionei os dados de: data vendas, valor do veiculo, nome do banco de dados 1 e o id da marca e marca do banco de dados 2 para criar as colunas.

Utilizei o join para unir as tabelas do banco de dados 1 e banco de dados 2, com a condição da junção sendo o id da marca.

1. **Qual marca teve o maior volume de vendas?**

Pra responder essa questão utilizei o código:

```
SELECT marca, SUM(vendas) AS volume_de_vendas
FROM unify_table
GROUP BY marca
ORDER BY volume_de_vendas DESC
LIMIT 1;
```

Selecionando duas colunas da tabela unificada (marca e vendas), usei a função SUM para que fosse calculado a soma total de vendas por marca, e o resultado nomeei como "volume_de_vendas_", usei o GROUP BY para que o resultado fosse agrupado com base na coluna marca, sendo calculado o volume de vendas separadamente. O ORDERBY DESC ordenou os resultados de forma decrescente, e colocando o LIMIT1, foi impresso pra mim a marca que teve o maior volume de vendas.

marca	volume_de_vendas
Fiat	433

2.Qual veículo gerou a maior e menor receita?

```
SELECT nome, valor_do_veiculo
FROM unify_table
ORDER BY valor_do_veiculo DESC
LIMIT 1;
```

Selecionei as colunas nome e valor_do_veiculo da tabela unificada e pedi para que fosse ordenada pelo valor do veiculo de modo decrescrente, limitando por 1.

nome	valor_do_veiculo
Forester	360000

```
SELECT nome, valor_do_veiculo
FROM unify_table
ORDER BY valor_do_veiculo ASC
LIMIT 1;
```

Selecionei as colunas nome e valor_do_veiculo da tabela unificada e pedi para que fosse ordenada pelo valor do veiculo de modo crescente, limitando por 1.

nome	valor_do_veiculo
Palio	8000

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

```
SELECT marca, strftime('%Y', data) AS ano, AVG(vendas) AS media_de_vendas
FROM unify_table
GROUP BY marca, ano
ORDER BY marca, ano;
```

Selecionei 3 colunas da tabela (marca, data e vendas), sei a função strftime para extrair o ano da coluna data e nomear o resultado como ano, usei AVG para calcular a media de vendas nomeando o resultado como media_de_vendas, usei GROUPBY para agrupar com base nas colunas marca e ano, o pedi para ordenas pela marca e ano.

marca	ano	media_de_vendas
Chevrolet	2022	3.6666666666666665
Fiat	2022	19.681818181818183
JaC Motors	2022	2.1666666666666665
Kia	2022	23
Mitsubishi	2022	3.8
Nissan	2022	3.2857142857142856
Peugeot	2022	11.555555555555555
Renault	2022	4.75
Subaru	2022	7.428571428571429
Toyota	2022	7.875
Volkswagen	2022	18.80952380952381

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

```
SELECT marca, SUM(vendas) AS total_de_vendas, SUM(valor_do_veiculo) AS
receita_total
FROM unify_table
GROUP BY marca
ORDER BY receita_total DESC, total_de_vendas ASC;
```

Selecionei marca, pedi a soma das vendas, nomeei como total_de_vendas, pedi a soma da coluna valor_do_veiculo e nomeei o resultado como receita_total, da tabela unificada, pedi para agrupar por marca e ordenar pela receita_total em modo decrescente e total_de_vendas em modo crescente.

marca	receita_total	total_de_vendas
Subaru	2150000	52
Mitsubishi	1464000	38
JaC Motors	549000	26
Chevrolet	390200	33
Toyota	647000	63
Renault	512000	57
Nissan	185000	23
Peugeot	613000	104
Volkswagen	1039000	395
Kia	676000	345
Fiat	747000	433

Código Python utilizado para geração de gráficos:

Gráfico vendas por carros:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Leitura do arquivo CSV
df = pd.read_csv('unify_table.csv')

# Agrupamento por marca e soma das vendas
vendas_por_marca =
df.groupby('marca')['vendas'].sum().sort_values(ascending=False)

# Cálculo das porcentagens
porcentagens = (vendas_por_marca / vendas_por_marca.sum()) * 100
```

```
# Configuração de cores e estilo
cores = sns.color_palette('pastel')[0:len(vendas_por_marca)]
sns.set_palette(cores)
sns.set(style="whitegrid")

# Gráfico de pizza
plt.figure(figsize=(10, 6))
plt.pie(vendas_por_marca, labels=None, autopct='', startangle=140,
colors=cores)

# Adiciona legenda ao lado
legend_labels = [f'{marca} ({percentagem:.1f}%)' for marca, percentagem
in zip(vendas_por_marca.index, porcentagens)]
plt.legend(legend_labels, title='Marcas', bbox_to_anchor=(1, 0.5),
loc="center left", fontsize='small')

# Adiciona título
plt.title('Distribuição de Vendas por Marca')

# Exibe o gráfico
plt.show()
```

Importei as bibliotecas “pandas, matplotlib e seaborn” para realizar a leitura do arquivo csv, manipular dados e criar um gráfico de pizza.

Média de vendas por veículo:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Leitura do arquivo CSV
df = pd.read_csv('unify_table.csv')

# Criação da coluna de receita
df['receita'] = df['vendas'] * df['valor_do_veiculo']

# Agrupamento por marca e cálculo da média de vendas
media_vendas_por_marca =
df.groupby('marca')['vendas'].mean().sort_values(ascending=False)
# Configurações do gráfico de barras
plt.figure(figsize=(12, 6))
sns.barplot(x=media_vendas_por_marca.index, y=media_vendas_por_marca,
palette='Purples')
plt.xlabel('Marca')
plt.ylabel('Média de Vendas')
plt.title('Média de Vendas por Marca no Ano')
```

```
# Adiciona os valores acima das barras
for i, valor in enumerate(media_vendas_por_marca):
    plt.text(i, valor + 0.2, f'{valor:.2f}', ha='center', va='bottom',
rotation=45, fontsize=8)

# Ajusta o layout para evitar o corte dos nomes
plt.tight_layout()

# Exibe o gráfico
plt.show()
```

Receita por veiculo

```
import pandas as pd
import matplotlib.pyplot as plt
# Leitura do arquivo CSV
df = pd.read_csv('unify_table.csv')

# Criação da coluna de receita
df['receita'] = df['vendas'] * df['valor_do_veiculo']

# Agrupamento por veículo e soma da receita, ordenado por receita
receita_por_veiculo = df.groupby('nome').agg({'vendas': 'sum', 'receita':
'sum'}).sort_values(by='receita', ascending=False)

# Configurações do gráfico de barras horizontal
plt.figure(figsize=(12, 8))
bars = plt.barh(receita_por_veiculo.index,
receita_por_veiculo['receita'], color='skyblue')

# Adiciona os valores de receita ao lado das barras
for bar in bars:
    plt.text(bar.get_width() + 5000, bar.get_y() + bar.get_height()/2,
             f'R${bar.get_width():,.2f}', va='center')
plt.xlabel('Receita')
plt.ylabel('Veículo')
plt.title('Receita por Veículo')

# Criação do quadro de legendas fora do gráfico
legendas = [f'{nome}: R${receita:,.2f}' for nome, receita in
zip(receita_por_veiculo.index, receita_por_veiculo['receita'])]
plt.legend(legendas, title='Legenda', bbox_to_anchor=(1, 1), loc='upper
left')
# Inverte a ordem dos veículos no eixo y para ter o maior valor no topo
plt.gca().invert_yaxis()

# Exibe o gráfico
plt.show()
```