



PROJETO 1 – Domínio de BI

1. PERGUNTAS ANALÍTICAS ([Código SQL](#))

- Pergunta 1: Qual é a distribuição das notas de avaliação dos produtos pelos clientes?

```
CREATE OR REPLACE VIEW vw_distribuicao_notas_avaliacoes AS
SELECT
    nota,
    COUNT(*) AS quantidade
FROM tbl_avaliacoes
GROUP BY nota
ORDER BY nota;
```

- Finalidade: Analisar a distribuição das notas de avaliação dos produtos em um Histograma. (RS8)

- Pergunta 2: Qual é o perfil demográfico dos clientes (por sexo) que mais realizaram compras?

```
CREATE OR REPLACE VIEW vw_perfil_demografico_clientes_compras AS
SELECT
    c.sexo,
    COUNT(v.id_venda) AS quantidade_compras
FROM
    tbl_clientes c
JOIN
    tbl_vendas v ON c.id_cliente = v.ce_id_cliente
WHERE
    c.deleted = FALSE
GROUP BY
    c.sexo
ORDER BY
    quantidade_compras DESC;
```

- Finalidade: Analisar demograficamente as vendas para entender o público (masculino ou feminino) que mais realizou compras na TechStore em um gráfico de barras. (RS9)

- **Pergunta 3: Qual é a proporção de clientes que têm interesses em diferentes categorias de produtos?**

```
CREATE OR REPLACE VIEW vw_proporcao_interesses_categorias AS
SELECT
    c.nome AS categoria,
    COUNT(i.id_interesse) AS quantidade_clientes,
    ROUND(COUNT(i.id_interesse)::numeric / (SELECT COUNT(*) FROM tbl_clientes) * 100, 2)
    AS proporcao
FROM tbl_interesses i
JOIN tbl_categorias c ON c.id_categoria = i.ce_id_categoria
GROUP BY c.nome
ORDER BY proporcao DESC;
```

- Finalidade: Visualizar o interesse dos clientes nas diferentes categorias de produtos em um gráfico de barras. (RS10)

- **Pergunta 4: Qual o ticket médio de vendas por categoria?**

```
CREATE OR REPLACE VIEW vw_ticket_medio_por_categoria AS

SELECT
    c.nome AS categoria,
    AVG(v.valor_total) AS ticket_medio
FROM tbl_vendas v
JOIN tbl_produtos_fornecedores pf
    ON pf.id_produto_fornecedor = v.ce_id_produto_fornecedor
JOIN tbl_produtos p ON p.id_produto = pf.ce_id_produto
JOIN tbl_categorias c ON c.id_categoria = p.ce_categoria
GROUP BY c.nome
ORDER BY ticket_medio DESC;
```

- Finalidade: Visualizar o valor das vendas dos produtos distribuídos por cada uma das categorias em um gráfico de barras. (RS11)

- **Pergunta 5: Quantos clientes distintos fizeram compras em relação ao total de clientes?**

```
CREATE OR REPLACE VIEW vw_clientes_compraram_vs_total AS
SELECT
    COUNT(DISTINCT v.ce_id_cliente) AS clientes_compraram,
    COUNT(*) AS total_clientes,
    ROUND(COUNT(DISTINCT v.ce_id_cliente)::numeric / COUNT(*) * 100, 2)
    AS proporcao
FROM
    tbl_clientes c
LEFT JOIN
    tbl_vendas v ON c.id_cliente = v.ce_id_cliente
WHERE
    c.deleted = FALSE;
```

- Finalidade: Visualizar a proporção de clientes que fizeram compra em relação à quantidade total de clientes num cartão simples. (RS12)

- **Pergunta 6: Qual a quantidade em estoque de um determinado produto considerando todos os seus fornecedores ativos?**

```
CREATE OR REPLACE VIEW vw_quantidade_total_estoque_produto AS
SELECT
    p.nome AS produto,
    SUM(pf.quantidade_em_estoque) AS quantidade_total_estoque
FROM tbl_produtos p
JOIN tbl_produtos_fornecedores pf ON p.id_produto = pf.ce_id_produto
GROUP BY p.nome
ORDER BY quantidade_total_estoque DESC;
```

- Finalidade: Visualizar a quantidade de cada produto em estoque, considerando todos os seus fornecedores em um gráfico de barras com nome e quantidade. (RS5)

- **Pergunta 7: Qual o valor investido para aquisição dos produtos em estoque?**

```
CREATE OR REPLACE VIEW vw_custo_total_estoque AS
SELECT
    p.nome AS produto,
    SUM(pf.quantidade_em_estoque * pf.preco_fornecimento) AS custo_total_estoque
FROM tbl_produtos p
JOIN tbl_produtos_fornecedores pf ON p.id_produto = pf.ce_id_produto
GROUP BY p.nome
```

- Finalidade: Entender o valor investido que está parado no estoque. Exibição flexível, pode ser um gráfico de barras por custo de aquisição total de cada produto ou um card único ao somar o total investido no tratamento dos dados. **(RS13)**

- **Pergunta 8: Como a quantidade total em estoque está distribuída entre as diferentes categorias de produto?**

```
CREATE OR REPLACE VIEW vw_quantidade_estoque_por_categoria AS
SELECT
    c.nome AS categoria,
    SUM(pf.quantidade_em_estoque) AS quantidade_total_estoque
FROM tbl_categorias c
JOIN tbl_produtos p ON c.id_categoria = p.ce_categoria
JOIN tbl_produtos_fornecedores pf ON p.id_produto = pf.ce_id_produto
GROUP BY c.nome;
```

- Finalidade: Entender quantos produtos de cada categoria temos no estoque. Exibir num gráfico de pizza. **(RS14)**

- **Pergunta 9: Qual o total de vendas por cliente?**

```
CREATE OR REPLACE VIEW vw_total_vendas_por_cliente AS
SELECT
    c.nome AS nome_cliente,
    SUM(v.valor_total) AS total_vendas
FROM
    tbl_clientes c
JOIN
    tbl_vendas v ON c.id_cliente = v.ce_id_cliente
WHERE
    c.deleted = FALSE
GROUP BY
    c.nome
ORDER BY
    total_vendas DESC;
```

- Finalidade: Identificar em uma lista com os clientes que mais geraram receitas para a empresa em ordem decrescente. **(RS15)**

- **Pergunta 10: Qual é o custo médio de aquisição por produto?**

```
CREATE OR REPLACE VIEW vw_custo_medio_aquisicao_por_produto AS
SELECT
    p.nome AS produto,
    AVG(pf.preco_fornecimento) AS custo_medio_aquisicao
FROM
    tbl_produtos p
JOIN
    tbl_produtos_fornecedores pf ON p.id_produto = pf.ce_id_produto
GROUP BY
    p.nome
ORDER BY
    custo_medio_aquisicao DESC;
```

- Finalidade: Identificar o custo médio de aquisição dos produtos em estoque. Para exibição em gráfico de barras. **(RS16)**

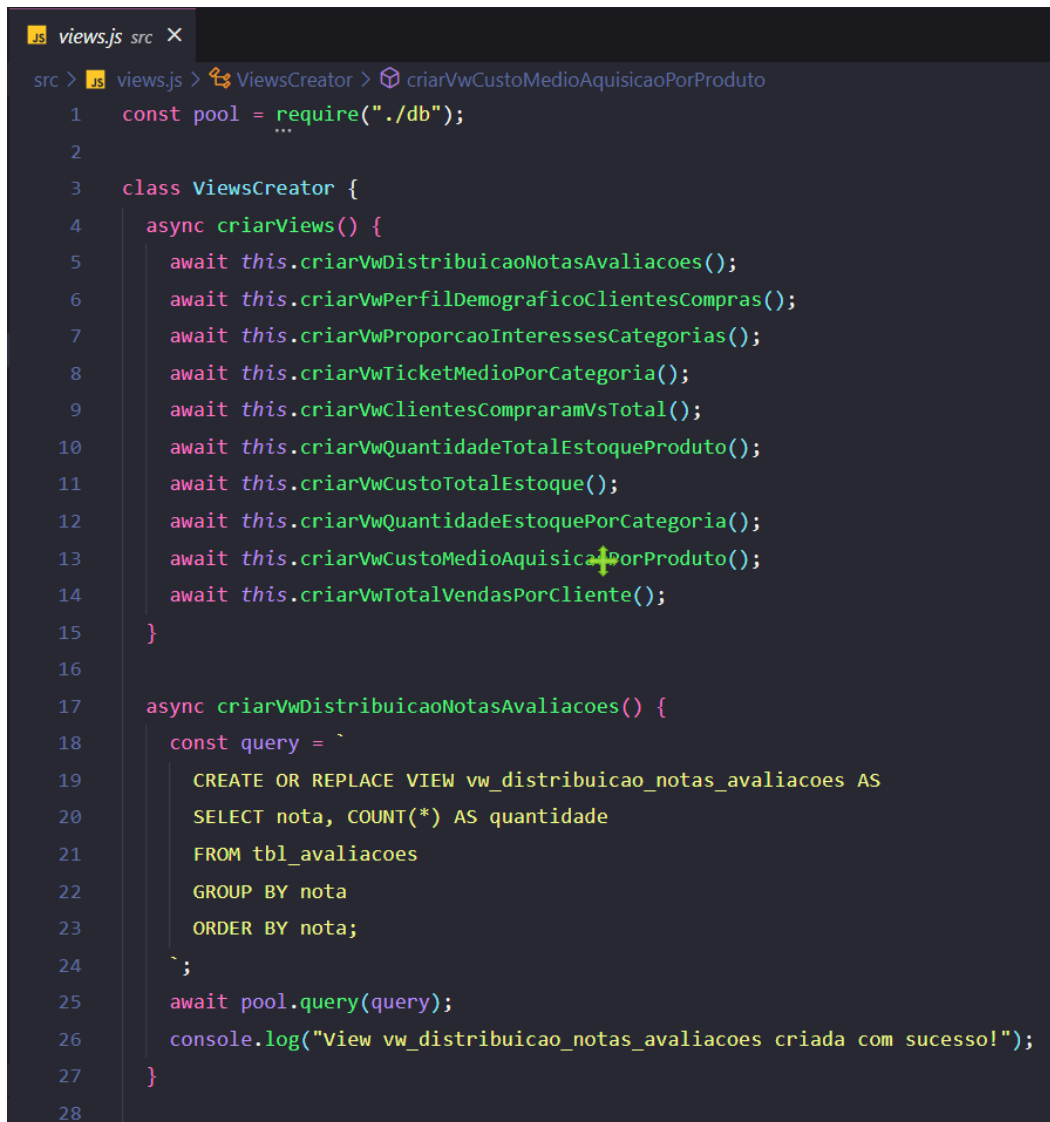
2. APLICAÇÃO

No projeto em NodeJS foi criada uma classe “ViewsCreator” que vai controlar a criação de views, ou seja, não é necessário rodar cada um dos comandos manualmente.

Sendo assim, ao seguir os passos da configuração do sistema (disponível em: [\[MATA60\] Configurando o Projeto Localmente](#)) será possível verificar no DBeaver os resultados gerados por cada uma das views com as respostas das perguntas.

Para demonstrar a efetividade dos comandos SQL associados a cada uma das respostas, fiz um vídeo sobre o domínio de BI disponível em:

[\[MATA60\] Dominio BI](#)



```
views.js src X
src > JS views.js > ViewsCreator > criarVwCustoMedioAquisicaoPorProduto
1  const pool = require("../db");
2
3  class ViewsCreator {
4    async criarViews() {
5      await this.criarVwDistribuicaoNotasAvaliacoes();
6      await this.criarVwPerfilDemograficoClientesCompras();
7      await this.criarVwProporcaoInteressesCategorias();
8      await this.criarVwTicketMedioPorCategoria();
9      await this.criarVwClientesCompraramVsTotal();
10     await this.criarVwQuantidadeTotalEstoqueProduto();
11     await this.criarVwCustoTotalEstoque();
12     await this.criarVwQuantidadeEstoquePorCategoria();
13     await this.criarVwCustoMedioAquisicaoPorProduto();
14     await this.criarVwTotalVendasPorCliente();
15   }
16
17   async criarVwDistribuicaoNotasAvaliacoes() {
18     const query = `
19       CREATE OR REPLACE VIEW vw_distribuicao_notas_avaliacoes AS
20       SELECT nota, COUNT(*) AS quantidade
21       FROM tbl_avaliacoes
22       GROUP BY nota
23       ORDER BY nota;
24     `;
25     await pool.query(query);
26     console.log("View vw_distribuicao_notas_avaliacoes criada com sucesso!");
27   }
28 }
```