**PROJETO INTEGRADOR:**

**ANÁLISE DO PRAZO E CICLO DE ENTREGA DE PRODUTOS POLIMÉRICOS**

**DOCUMENTO DE ARQUITETURA DE DADOS**

**Autores:**

Gabriela Colovati de Almeida

Giovanna Nucci Matos

Mayara Aparecida de Camargo Miranda

Paloma Cristina Pinheiro

Sheron Marciliana Lopes de Castro Alves

Taiana She Mir Mui

**Histórico da Revisão**

|  |  |  |
| --- | --- | --- |
| **Data** | **Versão** | **Descrição** |
| 29/12/2023 | 0 | Definição do Template |
| 02/01/2024 | 1 | Preenchimento de Informações: Introdução, Visão Geral, Dataset e Ferramentas Utilizadas |
| 05/01/2024 | 2 | Preenchimento de Informações: KPI´s, Banco de Dados, Análise de Dados |
| 08/01/2024 | 3 | Preenchimento de Informações: KPI´s, Banco de Dados, Análise de Dados |
| 10/01/2024 | 4 | Preenchimento de Informações de Análise da Dados |
| 15/01/2024 | 5 | Formatação ABNT |
| 17/01/2024 | 6 | Revisão |

**Índice Analítico**

Sumário

[1.0 INTRODUÇÃO 4](#_Toc156203884)

[1.1 Finalidade 6](#_Toc156203885)

[1.2 Escopo 6](#_Toc156203886)

[1.3 Definições, Acrônimos e Abreviações 6](#_Toc156203887)

[1.4 Descrição de Variáveis das Tabelas do Dataset 12](#_Toc156203888)

[2.0 VISÃO GERAL E DATASET 22](#_Toc156203889)

[3.0 FERRAMENTAS UTILIZADAS 22](#_Toc156203890)

[4.0 KEY PERFORMANCE INDICATORS (KPI´S) 24](#_Toc156203891)

[5.0 BANCO DE DADOS 26](#_Toc156203892)

[5.1 Diagrama Entidade Relacionamento 26](#_Toc156203893)

[5.2 Triggers e Procedures 27](#_Toc156203894)

[5.3 Queries 48](#_Toc156203895)

[5.4 Processo de Integração de Modelo de Dados 65](#_Toc156203896)

[6.0 ANÁLISE DE DADOS 67](#_Toc156203897)

[6.1 Dashboards Power BI e Excel 67](#_Toc156203898)

[6.2 Funções e Filtros Excel e Power BI 68](#_Toc156203899)

[6.3 Criação do Notebook 68](#_Toc156203900)

[6.4 Bibliotecas 69](#_Toc156203901)

[7.0 CONCLUSÕES 70](#_Toc156203902)

[8.0 REFERÊNCIAS 71](#_Toc156203903)

1. INTRODUÇÃO

O Supply Chain Management (SCM) é a gestão da cadeia de fornecimento, um sistema pelo qual empresas entregam seus produtos e serviços aos seus consumidores, numa rede de organizações interligadas, lidando com problemas de planejamento e execução envolvidos no gerenciamento de uma cadeia de suprimentos (SILVA, 2010).

De acordo com Chagas e Segger (2021) a logística nas empresas atualmente é um sistema complexo que conta com a ajuda da tecnologia para manter a eficiência, envolvendo o uso de diversos softwares e equipamentos. Além das ferramentas tecnológicas, a competição no mercado exige que as empresas destaquem seus serviços para garantir a preferência dos clientes. Isso significa que é necessário inovar na gestão, com foco especial na qualidade, para se manter relevante. Nesse contexto, os indicadores de desempenho logístico aparecem como uma ferramenta valiosa para monitorar os processos logísticos, proporcionando uma base mais confiável para a tomada de decisões cotidianas, resultando em possíveis reduções de custos, aumento de qualidade, aumento de competitividade e valor agregado.

O Business Intelligence (BI) é concebido como a inteligência de negócios, consistindo em um conjunto de metodologias de gestão implementadas por meio de ferramentas de software. A essência do Business Intelligence é a transformação de dados em conhecimento, visando fornecer suporte ao processo decisório e, consequentemente, gerar vantagens competitivas para a organização (FREITAS, 2017). Na Figura 1 são apresentados os pilares do BI.

Figura 1. Os pilares do Business Intelligence.



Fonte: (SITEWARE, 2023).

Resumidamente, o BI pode ser dividido em:

* Coleta de dados: Inicialmente, são coletadas, armazenadas e analisadas as informações.
* Organização e análise: Os dados coletados são examinados, organizados em bancos de dados e apresentados visualmente em dashboards estratégicos.
* Ação e monitoramento: Com base nas informações, os responsáveis tomam decisões e monitoram os resultados obtidos.

Conforme destacado por Freitas (2017), o mercado oferece uma variedade de softwares destinados à inteligência de negócios, abrangendo opções tanto pagas quanto open source, com código livre e disponibilidade gratuita para utilização. Algumas dessas ferramentas incluem o Power BI, Tableau, SpagoBI, Pentaho e Jaspersoft.

* 1. Finalidade

Este projeto tem como finalidade apresentar a análise de dados de um banco de dados de materiais poliméricos para uso industrial. O objetivo é avaliar os indicadores de performance de prazos de entregas dos produtos.

* 1. Escopo

O escopo deste documento engloba desde a escolha do *dataset,* manipulação dos dados, criação de relacionamentos, triggers, procedures, queries, integração dos dados, análise de KPIs de prazo de entrega e criação de dashboards.

* 1. Definições, Acrônimos e Abreviações

Essa subseção contém as definições de todos os termos, acrônimos e abreviações necessárias.

1. **Modelagem de Dados****:** A modelagem de dados pode ser definida como um processo onde, através do levantamento dos requisitos de informação e regras de negócio, aplicando técnicas e mecanismos de abstração, construímos artefatos (modelos de dados) que tem por objetivo representar um conjunto de informações. Estes normalmente são geridos por uma aplicação.

1. **Modelo Lógico de Dados****:** É aquele em que os objetos, suas características e relacionamentos não seguem, necessariamente, as regras de implementação limitantes impostas por algum tipo de tecnologia de armazenamento, embora já seja possível definir algumas de suas propriedades físicas, levando-se em consideração, características específicas de um Sistema Gerenciador de Banco de Dados (SGBD) por exemplo. Neste nível, respeita-se conceitos de chaves de acesso, normalização, integridade referencial, formatos de campos, etc. O modelo lógico de dados deve ter uma disposição correta e bem distribuída, tanto das entidades quanto dos relacionamentos, sendo este um critério a ser observado na avaliação do modelo.
2. **Entidade****:** É a representação de “algo” que existe no mundo real. Uma entidade pode representar um elemento concreto, como uma pessoa ou um livro, ou um elemento abstrato, como um empréstimo, uma viagem ou um conceito. Um conjunto de entidades contém elementos que são similares e compartilham as mesmas características. Cada elemento deste conjunto é dito uma ocorrência da entidade e será identificado de forma unívoca em relação aos outros elementos. As entidades são classificadas em:

* **Entidade Associativa**: A entidade associativa surge de um relacionamento NxM, em que existe uma associação dos atributos identificadores das duas entidades relacionadas, caracterizando uma nova entidade. A nova entidade gerada possui, normalmente, atributos próprios do relacionamento. Também se utiliza esse tipo de entidade para representar relacionamentos múltiplos (envolvendo mais de 2 entidades) ou agregações (relacionamentos que se relacionam com outras entidades).
* **Entidade Forte (Independente ou Dominante):** uma entidade x é dita forte quando sua existência não depende da existência de outra entidade.
* **Entidade Fraca (Dependente ou Subordinada):** uma entidade x é chamada fraca quando sua existência depende hierarquicamente da existência de outra entidade y. A entidade y é chamada entidade dominante (forte) e a x é chamada entidade subordinada (fraca).
* **Superclasse (Supertipo):** quando diferentes entidades possuem características comuns, essas características podem ser agrupadas em uma entidade genérica de nível superior chamada superclasse. O processo de agrupamento é chamado generalização.
* **Subentidade (Subtipo):** quando a partir da subdivisão de uma entidade, segundo um critério de classificação, são obtidas outras entidades com características próprias. Cada entidade obtida com a subdivisão é chamada de subclasse e representa um subconjunto da entidade superior, a superclasse. O processo de subdivisão é dito Especialização.

1. **Atributo****:** Uma entidade é representada por um conjunto de atributos. Um atributo é uma propriedade que qualifica, descreve ou identifica um dado dentro da entidade. Os atributos descrevem cada membro de um conjunto de entidades. Um atributo pode ser:

* **Simples (Atômico):** quando o atributo possui um valor que não é divisível (ex.: sexo);
* **Composto**: quando o atributo possui um valor que pode ser dividido em partes menores (outros atributos) (ex.: endereço);
* **Monovalorado:** quando o atributo assume um único valor para cada elemento do conjunto de entidades (ex.: nome);
* **Multivalorado:** quando o atributo pode ter mais de um valor para uma única entidade (ex.: dependentes, telefones);
* **Derivado:** quando o valor do atributo é derivado de outros atributos da própria entidade ou a partir de atributos de entidades relacionadas (ex.: idade);
* **Nulo:** em alguns casos, uma determinada entidade pode não ter um valor aplicável para um atributo, ou este valor é desconhecido (ex.: número do apartamento).

1. **Relacionamento****:** Trata-se da associação existente entre uma ou mais entidades. As modalidades de relacionamentos são as seguintes:

* **Relacionamento Identificador:** é aquele que faz parte da composição do identificador único de uma entidade.
* **Relacionamento Não Identificador:** é aquele que não faz parte da composição do identificador único de uma entidade.
* **Auto-relacionamento:** é aquele em que uma entidade se relaciona com ela mesma (relacionamento recursivo).

1. **Cardinalidade****:** É a quantidade de vezes em que um elemento de um conjunto de entidades pode, em um determinado instante, estar associado, em um dado relacionamento, a outros elementos de outras entidades. Quanto à cardinalidade, os relacionamentos podem ser classificados como:

* **Um para um (1:1):** Quando um elemento da entidade A pode estar associado no máximo a um elemento da entidade B, e um elemento da entidade B pode estar associado no máximo a um elemento da entidade A.
* **Um para muitos (1:N):** Quando um elemento da entidade A pode estar associado a vários elementos da entidade B. Entretanto, um elemento da entidade B pode estar associado no máximo a um elemento da entidade A.
* **Muitos para muitos (N:M):** Quando um elemento da entidade A pode estar associado a vários elementos da entidade B, e um elemento da entidade B pode estar associado a vários elementos da entidade A.

1. **Restrição de Integridade****:** É uma regra que deve ser obedecida para que se mantenha a consistência dos dados de um banco de dados. As restrições de integridade podem ser dos seguintes tipos:

* **Conteúdo:** quando um determinado atributo deve obrigatoriamente possuir um valor ou não. É o que definimos de atributo obrigatório ou opcional.
* **Domínio:** quando limita os valores que um atributo pode conter. Fica registrada quando se define o formato, tamanho, e valores possíveis de um atributo.
* **Referencial:** representada por um relacionamento entre duas entidades do modelo lógico, é usada para manter a consistência entre os dados relacionados nas duas entidades.
* **Negócio (Semântica):** quando representa uma regra característica do negócio em estudo, que não se enquadra nos demais tipos.

1. **Relação Normalizada****:** A normalização de dados consiste na realização de vários passos seguidos no projeto de um banco de dados, que permitem um armazenamento consistente e eficiente acesso aos dados em bancos de dados relacionais. Esses passos reduzem a redundância de dados e as chances dos dados se tornarem inconsistentes. A técnica de normalização prevê a existência de regras, chamadas de formas normais, que devem ser aplicadas com o objetivo de alcançar certas condições desejáveis que uma relação deve satisfazer. Uma relação normalizada é a que esteja pelo menos na terceira forma normal, ou seja:

* Não contenha colunas com elementos multivalorados ou compostos (1ª forma normal);
* Cada coluna que não faça parte da chave primária depende, funcionalmente, de todas as partes da chave primária (2ª forma normal);
* Cada coluna que não faça parte da chave primária depende, funcionalmente, apenas da chave primária, e de nenhuma outra coluna fora da chave primária (3ª forma normal).

1. **Modelo Físico de Dados****:** O modelo de dados físico deve ser criado a partir da transformação de um modelo de dados lógico. Nele a representação dos objetos, suas características e relacionamentos seguem as regras de implementação impostas pelo tipo de tecnologia de armazenamento utilizada para implementar as estruturas de dados deste. Neste nível, são introduzidos conceitos clássicos relacionados a modelos de dados físicos estruturados.

* **Tabela:** estrutura de armazenamento de dados, formada por um conjunto finito de colunas e ilimitado de linhas (tuplas). Uma tabela é equivalente a uma entidade no modelo lógico.
* **Linha (ou tupla):** representação equivalente à ocorrência de uma entidade. Cada linha representa um registro dentro da tabela.
* **Coluna:** são unidades básicas de armazenamento de dados em uma linha dentro de uma tabela. Uma coluna é similar a um atributo no modelo lógico.
* **Chave Primária:** é um atributo ou um conjunto de atributos cujos valores distinguem ou identificam de forma unívoca cada linha dentro de uma tabela.
* **Chave Única (Unique Key):** elemento que especifica que outros valores de coluna(s), além da chave primária, devem ser únicos.
* **Chave Estrangeira (Foreign Key):** representa um atributo, ou combinações de atributos, cujos valores aparecem na chave primária ou em uma chave única de outra entidade que se relaciona com esta, onde encontramos a chave estrangeira. A chave estrangeira é oriunda do relacionamento entre as entidades.
* **Índice:** recurso físico que visa otimizar a recuperação de uma informação, via um método de acesso, fornecendo acesso rápido a linhas específicas no banco de dados, eliminando a necessidade de varreduras completas de tabelas. Um índice é a seleção de uma ou mais colunas de uma tabela para já armazenar os seus dados ordenados, acompanhado de ponteiros que servirão como atalhos para a localização física dos registros.
* **Regra de Verificação (Check Constraint):** elemento que especifica regras para os valores que uma coluna poderá receber.
* **Sequência (Sequence):** elemento que gera uma numeração serial e única para colunas numéricas de uma tabela.
* **Visão (View):** Representa uma visão parcial ou completa de uma ou mais tabela.
* **Visão Materializada (Materialized View):** réplica de uma tabela ou visão a partir de um único ponto no tempo. Muito utilizada em modelos multidimensionais para compartilhar estruturas de dados.

* 1. Descrição de Variáveis das Tabelas do Dataset

1. **Tabela Família:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Familia\_numero | INT | Chave Primária da tabela Família |
|  | Nome | VARCHAR (45) | Nome da família do produto |

1. **Tabela Linha:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Linha\_numero | INT | Chave Primária da tabela Linha |
|  | Nome | VARCHAR (45) | Nome da Linha |

1. **Tabela Produto:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Produto\_numero | INT | Chave Primária da tabela Produto |
|  | Nome | VARCHAR (160) | Nome do produto |
|  | Codigo | VARCHAR(15) | Código de cadastro do produto |
|  | Grupo | INT | Grupo a que o produto pertence |
|  | Fabricante | VARCHAR(60) | Nome do fabricante |
|  | Fabricado\_por | VARCHAR(30) | Fabricado por terceiros |
|  | Conteudo\_unidade | DECIMAL(21,6) | Conteúdo da embalagem |
|  | Unidade\_medida | VARCHAR(15) | Unidade de medida |
|  | Peso\_liquido | DECIMAL(21,6) | Peso líquido do produto |
|  | Peso\_bruto | DECIMAL(21,6) | Peso bruto do produto |
|  | Preco\_compra | DECIMAL(21,6) | Preço de compra do produto |
|  | Preco\_venda | DECIMAL(21,6) | Preço de venda do produto |
|  | Estoque\_minimo | INT | Estoque mínimo do produto |
|  | Estoque\_maximo | INT | Estoque máximo do produto |
|  | Estoque\_ideal | INT | Estoque ideal do produto |
|  | Tempo\_reposicao | INT | Tempo para reposição do produto |
| Chave Estrangeira | Familia\_numero | INT | Chave Primária da tabela Familia |
| Chave Estrangeira | Linha\_numero | INT | Chave Primária da tabela Linha |

1. **Tabela PedidoVendaItem:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Seria | INT | Chave primária da tabela PedidoVendaItem |
| Chave Primária | Pedido\_numero | DECIMAL(21,6) | Chave primária da tabela PedidoVendaItem |
|  | Qtde | INT | Quantidade de itens |
|  | Preco\_venda | DECIMAL(21,6) | Preço de venda do item |
|  | Peso\_liquido | DECIMAL(21,6) | Peso líquido do item |
|  | Peso\_bruto | DECIMAL(21,6) | Peso bruto do item |
|  | Valor\_frete | DECIMAL(21,6) | Valor do frete |
| Chave Estrangeira | Produto\_numero | INT | Chave Primária da tabela Produto |
|  | Preco\_total | DECIMAL(21,6) | Preço total do item |
|  | Peso\_liquido | DECIMAL(21,6) | Peso líquido do item |
|  | Peso\_bruto | DECIMAL(21,6) | Peso bruto do item |

1. **Tabela Operacao:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Operacao\_numero | INT | Chave primária da tabela Operacao |
|  | Nome | VARCHAR(45) | Nome da operação |

1. **Tabela Uso:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Uso\_numero | INT | Chave primária da tabela Uso |
|  | Uso | VARCHAR(45) | Nome do Uso |

1. **Tabela Municipio:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Municipio\_numero | INT | Chave primária da tabela Município |
|  | Municipio | VARCHAR(45) | Nome do Município |
|  | Mun\_IBGE | INT | Código de cadastro do município no IBGE |
| Chave Estrangeira | UF | VARCHAR(2) | Chave Primária da tabela UF |

1. **Tabela UF**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | UF | VARCHAR(2) | Chave primária da tabela UF |
|  | Descricao | VARCHAR(50) | Nome do estado |
|  | Regiao | VARCHAR(50) | Região a que o estado pertence |

1. **Tabela PedidoVenda:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Serie | INT | Chave primária da tabela PedidoVenda |
| Chave Primária | Pedido\_numero | DECIMAL | Chave primária da tabela PedidoVenda |
|  | Ordem\_venda | VARCHAR(60) | Descrição do tipo de venda |
|  | Emitido\_em | DATE | Data da emissão do documento |
|  | Previsao\_entrega | DATE | Data da previsão de entrega |
|  | Pedido\_cliente | VARCHAR(45) | Código do pedido do cliente |
|  | DT\_Embarque | DATE | Data do embarque do pedido |
|  | DT\_entregue | DATE | Data da entrega do pedido |
|  | Ordem\_embarque | VARCHAR(45) | Código da ordem de embarque |
|  | Transporte | VARCHAR(60) | Tipo de transporte (CIF/FOB) |
|  | Volume | INT | Volume do pedido |
|  | Especie | VARCHAR(60) | Espécie de embalagem |
|  | Valor\_mercadoria | DECIMAL(21,6) | Valor da mercadoria |
|  | Frete | DECIMAL(21,6) | Valor do frete |
|  | Seguro | DECIMAL(21,6) | Valor do seguro |
|  | Acrescimos | DECIMAL(21,6) | Valor dos acréscimos |
|  | Peso\_liquido | DECIMAL(21,6) | Peso líquido do pedido |
|  | Peso\_bruto | DECIMAL(21,6) | Peso bruto do pedido |
|  | OnTime | INT | Pedido entregue no prazo |
|  | InFull | INT | Pedido entregue completo |
|  | OnTime\_InFull | INT | Pedido entregue no prazo e completo |
| Chave Estrangeira | Cliente\_numero | INT | Chave Primária da tabela Cliente |
| Chave Estrangeira | Operação\_numero | INT | Chave Primária da tabela Operacao |
| Chave Estrangeira | Uso\_numero | INT | Chave Primária da tabela Uso |
| Chave Estrangeira | Transportadora\_numero | INT | Chave Primária da tabela Transportadora |
| Chave Estrangeira | Cod\_log\_entrega | INT | Chave Primária da tabela Endereco |

1. **Tabela Endereco:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Cod\_log | INT | Chave Primária da tabela Endereco |
|  | Regiao | VARCHAR(50) | Região do endereço |
|  | Bairro | VARCHAR(50) | Bairro do endereço |
|  | Logradouro | VARCHAR(40) | Tipo de logradouro |
|  | Endereco | VARCHAR(100) | Nome do endereco |
|  | CEP | VARCHAR(10) | Código de endereçamento postal |
| Chave Estrangeira | Municipio\_numero | INT | Chave Estrangeira da tabela Municipio |

1. **Tabela Cliente:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Cliente\_numero | INT | Chave primária da tabela Cliente |
|  | Cliente | VARCHAR(80) | Razão social do cliente |
|  | CNPJ | VARCHAR(15) | Cadastro Nacional de Pessoas Jurídicas |
|  | IE | VARCHAR(25) | Inscrição Estadual |
|  | IM | VARCHAR(14) | Inscrição Municipal |
|  | Telefones | VARCHAR(60) | Números de telefones dos Clientes |
|  | Numero\_compra | INT | Número do endereço de compra |
|  | Numero\_entrega | INT | Número do endereço de entrega |
| Chave Estrangeira | Grupo\_numero | INT | Chave Primária da tabela Grupo |
| Chave Estrangeira | Cod\_log\_compra | INT | Chave Primária da tabela Endereco |
| Chave Estrangeira | Cod\_log\_entrega | INT | Chave Primária da tabela endereco |

1. **Tabela Transportadora:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Transportadora\_numero | INT | Chave Primária da tabela Transportadora |
|  | Nome | VARCHAR(100) | Nome da transportadora |
|  | CNPJ | VARCHAR(15) | Cadastro Nacional de Pessoas Jurídicas |
|  | IE | VARCHAR(25) | Inscrição Estadual |
|  | IM | VARCHAR(14) | Inscrição Municipal |
|  | Telefones | VARCHAR(60) | Números de telefones dos Clientes |
|  | Numero | INT | Numero do endereço |
| Chave Estrangeira | Grupo\_numero | INT | Chave Primária da tabela Grupo |
| Chave Estrangeira | Cod\_log | INT | Chave Primária da tabela Endereco |

1. **Tabela Grupo:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | Grupo\_numero | INT | Chave Primária da tabela Grupo |
|  | Nome | VARCHAR(45) | Nome do grupo |

1. **Tabela Log:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Armazenamento** | **Nome da Coluna** | **Tipo do Dado** | **Descrição da Variável** |
| Chave Primária | id | INT | Chave Primária da tabela Log |
|  | dt\_registro | DATE | Data de registro da modificação |
|  | user | VARCHAR(30) | Usuário que efetuou a modificação |
|  | tipo\_acao | VARCHAR(10) | Ação do registro (Insert, Update ou Delete) |
|  | nome\_tabela | VARCHAR(30) | Nome da tabela que foi modificada |
|  | id\_linha | VARCHAR(30) | id da linha que foi modificada |

1. VISÃO GERAL E DATASET

O dataset selecionado para este projeto é extraído de um banco de dados SQL Server integrado ao ERP AGE, desenvolvido pela Estratégia Software. Esse dataset foi escolhido por representar uma estrutura de banco de dados já em uso, proporcionando informações pertinentes ao tema em foco.

Os dados utilizados são provenientes do ambiente de desenvolvimento da empresa, apresentando informações genéricas que simulam uma indústria do setor de transformação de material plástico. A empresa concedeu autorização para a utilização desses dados, porém, realizamos ajustes na estrutura original com o intuito de simplificar o modelo de negócios. Optamos também por gerar novos dados de cliente e transportadora através de um sistema de inteligência artificial, evitando manter qualquer informação que possa identificar ou pertencer a empresas reais.

A estrutura do dataset é composta por 13 tabelas, sendo as principais: Cliente, Produto, PedidoVenda e PedidoVendaItem. Essas tabelas desempenham um papel fundamental ao oferecerem a base de informações necessárias para análises detalhadas e compreensão do processo de vendas. Elas possibilitam entender os relacionamentos entre clientes, produtos e os itens associados aos pedidos realizados na indústria

1. FERRAMENTAS UTILIZADAS

Para a execução deste projeto, abaixo estão descritas a função de cada ferramenta utilizada e seus respectivos links de acesso.

**a) Trello:** foi utilizado como ferramenta de gerenciamento do projeto, onde foram criadas listas de backlog do produto, a fazer, em andamento, fase de teste e concluído, conforme apresentado na Figura 2. Tarefas foram atribuídas aos membros da equipe com descrições e prazos de execução.

Figura 2. Trello do Projeto Integrador.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Link de acesso:

<https://trello.com/invite/b/xlXZtatS/ATTI087291b6ae6751d57828139d52cc84588267F55C/projeto-integrador-embraer>

**b) GitHub:** O GitHub foi utilizado como um repositório para armazenamento de arquivos dos projetos, facilitando o compartilhamento e a colaboração entre os membros da equipe. No Github foram criadas pastas e sub-pastas para organização dos arquivos (referências, apresentação powerpoint, base de dados, dashboards e notebook).

Link de acesso: <https://github.com/NuttyLee22/Block-Chain-Analysis>

**c) Discord:** O aplicativo Discord foi utilizado para comunicação através de chats, vídeos e canais organizados por temas. Facilitando o compartilhamento de informações de maneira assertiva para o projeto.

**d) MySQL Workbench:** A ferramenta foi utilizada para modelagem, criação, modificação, gerenciamento, definição das chaves primárias e estrangeiras e conexão do banco de dados.

**e)** [**Jupyter Notebook**](https://github.com/NuttyLee22/Block-Chain-Analysis/commit/a9ee2985a0a96f86589dc892903b782880294fa4)**:** Foi utilizado para criação de códigos em python para realização de análises exploratórias e visualização dos dados avançadas.

**f) Excel:** O Excel foi utilizado para normatizar os dados, criação de tabelas, formatação condicional, tabelas dinâmicas e criação de Dashboard com gráficos para análise dos dados.

**g) Power BI:** O Power Bi foi utilizado para criação de Dashboards com o intuito de transformar dados brutos em informações visuais importantes para tomadas de decisões. Foram criadas tabelas, gráficos e visualizações dinâmicas com relatórios interativos.

**H) Figma:** O Figma foi empregado como ferramenta para concepção do layout dos dashboards, caracterizando-se por uma interface intuitiva. Essas características possibilitam a eficiente criação de elementos visuais e interações. Além disso, a plataforma oferece recursos como prototipagem interativa e compartilhamento, facilitando o desenvolvimento de dashboards atrativos e funcionalmente robustos.

1. KEY PERFORMANCE INDICATORS (KPI´S)

De acordo com Cavalcanti (2015)os indicadores chaves de desempenho logísticos medem, avaliam e auxiliam o controle da performance do serviço, monitorando a qualidade das atividades.

Dentre os diversos tipos de indicadores chave de desempenho logísticos, foram escolhidas 5 para uma análise mais aprofundada, relacionada ao prazo e ciclo de entregas dos pedidos.

**a) Tempo de entrega:** Tempo decorrido da saída do armazém ou centro de distribuição até a chegada do pedido ao cliente, em dias.  Esse indicador mede o tempo necessário para entregar um produto ou pedido ao cliente. Portanto, dependendo da operação logística, pode ser medido em horas, dias ou semanas.

**b) Tempo de ciclo do pedido:** Tempo decorrido desde as efetivações das compras até as entregas desses pedidos ao cliente, em dias.

Cavalcanti (2015) afirma que o tempo exerce uma influência significativa no ciclo do pedido, levando as organizações a demonstrarem uma crescente preocupação em reduzir tanto o tempo dos ciclos quanto o prazo de entrega. Isso se deve à expectativa dos clientes em receberem seus pedidos de maneira rápida e completa. Esse KPI fornece e identifica se o ciclo de pedidos de uma empresa é curto, médio e longo. Possibilitando realizar previsões e tomar iniciativas para otimizar o tempo, até mesmo, obtendo maior rapidez no tempo de ciclo dos pedidos com ciclos mais curtos.

**c) On Time - In Full (OTIF - Pedidos Completos e no Prazo):** Número de pedidos recebidos pelo cliente no prazo e quantidades acordadas / número total de pedidos entregues, em porcentagem. O OTIF é um indicador de entregas perfeitas, ou seja, entregas eficientes (pedidos atendidos completamente) e entregas rápidas (no prazo estimado).

Conforme Moraes (2019) esse indicador tem o propósito de avaliar a competência do ciclo de entregas de uma empresa, proporcionando uma mensuração da eficiência logística sob a perspectiva do consumidor. Ou seja, este indicador observa se o pedido foi entregue dentro do prazo (on-time) e se os demais processos de atendimento e logísticos foram eficientes (in-full). Os autores Silva e Silva (2019) afirmam que este indicador é considerado um dos mais relevantes na cadeia logística, pois verifica se o cliente recebeu a mercadoria que pediu, dentro do prazo e no endereço correto. Uma boa meta é alcançar cerca de 80% a 90% de OTIF pelo fato de ser um indicador estratégico para uma tomada de decisão.

**d) On time delivery (OTD):** Número de pedidos entregues ao cliente no prazo acordado / número total de pedidos entregues, em porcentagem.

O OTD é um indicador de entregas realizadas dentro do prazo, ou seja, é um tradutor que aponta quantas entregas de produtos foram realizadas na data e horário acordado com os clientes/consumidores, sendo considerado um dos elementos do OTIF. De acordo com Simões, Moura e Okano (2019) o consumidor tem atribuído crescente importância ao prazo de entrega ao tomar decisões de compra. Dessa forma, o indicador OTD (On-Time Delivery) torna-se crucial para garantir uma experiência satisfatória aos clientes. É possível conduzir uma análise estratégica que abrange desde o momento da separação e expedição do pedido até a entrega final do produto pela transportadora ao consumidor. Essa abordagem visa aprimorar tanto a qualidade quanto o nível do serviço de entrega, promovendo uma melhoria contínua. O OTD será positivo sempre que o consumidor receber o pedido antes ou no horário estipulado desde o momento da compra. Caso contrário, se o percentual de entregas realizadas depois do prazo for recorrente, gera-se uma série de consequências negativas para o negócio, como a perda de público (o negócio acaba ficando com a reputação baixa no mercado) e aumento dos índices de reclamações dos consumidores, sendo assim é importante avaliar o processo do delivery, melhorando a qualidade e o nível do serviço de entrega.

**e) Order Fill Rate (OFR):**  Número de pedidos atendidos completamente / Número total de pedidos entregues, em porcentagem. O OFR é indicador de desempenho de atendimento do pedido, além de ser um indicador que consegue mostrar se as operações podem a qualquer momento corresponder e satisfazer às necessidades dos clientes. Além disso, o Order Fill Rate é importante indicador que consegue medir a eficiência e performance de uma empresa, avaliando se as demandas foram cumpridas com precisão, considerando desde o tempo gasto no recebimento do pedido até a saída e entrega do mesmo, firmando uma vantagem competitiva nas execuções dos pedidos.

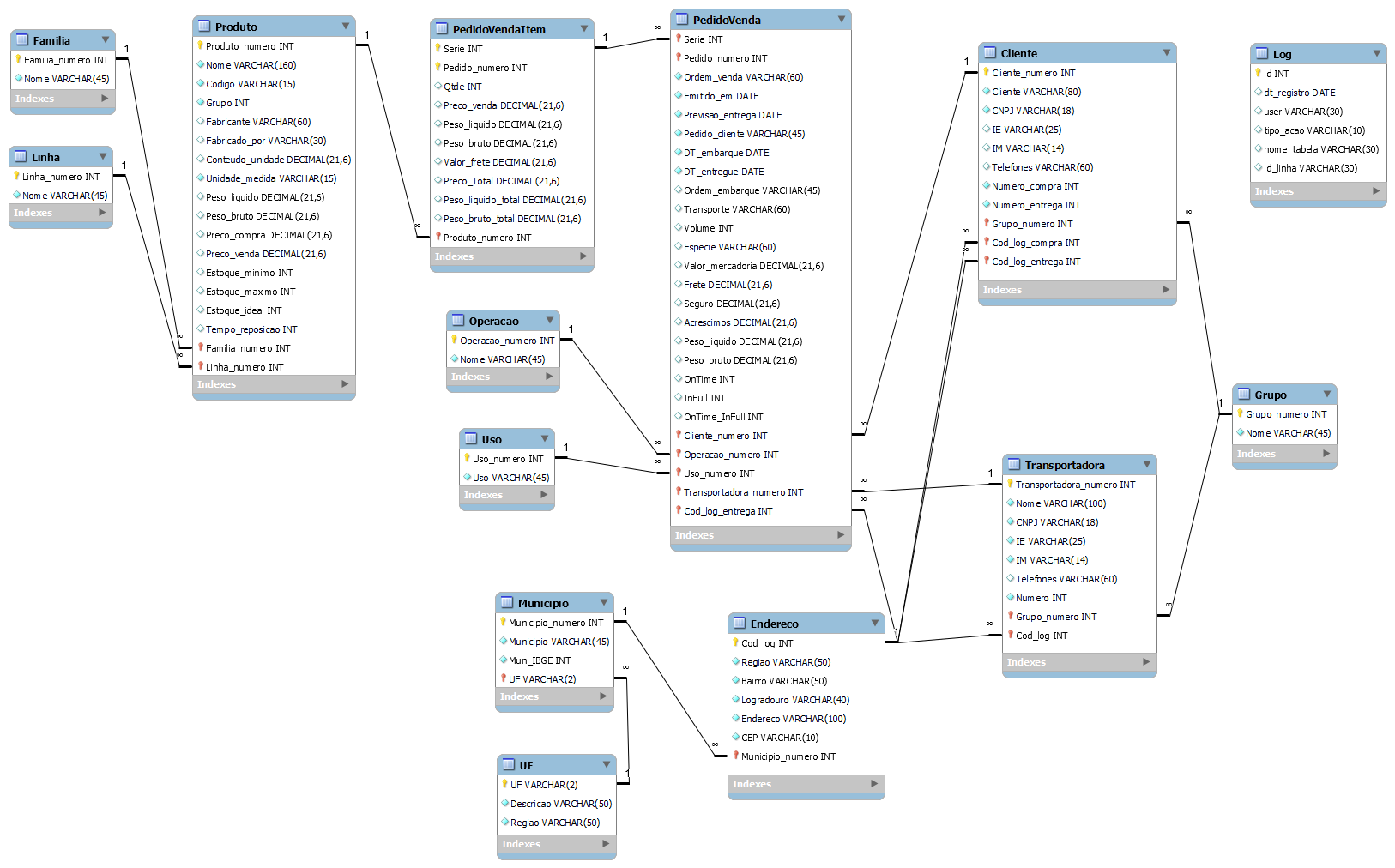
1. BANCO DE DADOS

O MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto, destacando-se pela gratuidade, confiabilidade e eficiência em lidar com grandes volumes de dados. Sua ampla compatibilidade com diversas plataformas e linguagens de programação destaca sua flexibilidade, permitindo integração suave com diferentes tecnologias. Essas características fazem do MySQL uma escolha acessível e versátil para organizações que buscam uma gestão robusta de dados.

* 1. Diagrama Entidade Relacionamento

O Diagrama Entidade-Relacionamento (DER) é uma representação visual que descreve as relações entre entidades em um sistema de informação. Utilizando entidades para representar objetos do mundo real e relacionamentos para descrever as conexões, o DER facilita a compreensão da estrutura lógica de um banco de dados. A modelagem através de DER facilita a compreensão e a documentação da estrutura lógica de um banco de dados, fornecendo uma visão clara e intuitiva das associações entre diferentes elementos, o que é crucial para o desenvolvimento e a manutenção eficientes de sistemas de informação. Na Figura 3 é presentada o DER do projeto.

Figura 3. Diagrama Entidade Relacionamento.



Fonte: próprios autores.

* 1. Triggers e Procedures

No contexto do sistema de gerenciamento de banco de dados MySQL, a utilização de triggers (gatilhos) e stored procedures (procedimentos armazenados) desempenha um papel crucial na otimização e automação de operações. Triggers são conjuntos de instruções que são automaticamente executadas em resposta a determinados eventos, como inserção, atualização ou exclusão de dados em uma tabela.

Por outro lado, stored procedures são blocos de código SQL que podem ser armazenados e reutilizados de forma eficiente. Esses procedimentos armazenados facilitam a execução de tarefas complexas, permitindo a encapsulação de lógica de negócios em uma unidade única e executável.

Dito isso, foi criado um conjunto de stored procedures e triggers destinados a manter um registro de atividades (log) relacionadas a diversas tabelas, como GRUPO, UF, MUNICIPIO, ENDERECO, CLIENTE, TRANSPORTADORA, FAMILIA, LINHA, PRODUTO, OPERACAO, USO, PEDIDOVENDAITEM E PEDIDOVENDA. A stored procedure PROC\_INSERE\_LOG foi criada para inserir informações relevantes em uma tabela de log sempre que é chamada. Cada tabela mencionada possui três triggers associados (AFTER INSERT, AFTER UPDATE e AFTER DELETE), que invocam a stored procedure para registrar as operações realizadas nessas tabelas, como inserções, atualizações e exclusões.

Por exemplo, ao ocorrer uma inserção, atualização ou exclusão na tabela 'GRUPO', o trigger TG\_LOG\_GRUPO\_AFTER\_INSERT, TG\_LOG\_GRUPO\_AFTER\_UPDATE ou TG\_LOG\_GRUPO\_AFTER\_DELETE é acionado, respectivamente, capturando detalhes da operação atual e chamando a stored procedure PROC\_INSERE\_LOG para registrar essas informações na tabela de log. Essa abordagem fornece uma maneira eficaz de rastrear alterações no banco de dados, auxiliando na auditoria e na solução de problemas relacionados à manipulação de dados. A seguir segue o código fonte dos triggers e procedures criados.

-- PROCEDURE proc\_insere\_log

DELIMITER //

CREATE PROCEDURE proc\_insere\_log

(in nm\_usuario VARCHAR(255), in tp\_acao VARCHAR(255), in nm\_tabela VARCHAR(255), in id\_linha VARCHAR(255))

BEGIN

INSERT INTO log VALUES (0, NOW(), nm\_usuario, tp\_acao, nm\_tabela, id\_linha);

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_GRUPO

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_GRUPO\_AFTER\_INSERT

AFTER INSERT ON grupo

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'GRUPO', CAST(NEW.grupo\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_GRUPO\_AFTER\_UPDATE

AFTER UPDATE ON grupo

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'GRUPO', CAST(NEW.grupo\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_GRUPO\_AFTER\_DELETE

AFTER DELETE ON grupo

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'GRUPO', CAST(OLD.grupo\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_UF

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_UF\_AFTER\_INSERT

AFTER INSERT ON UF

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'UF', CAST(NEW.uf AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_UF\_AFTER\_UPDATE

AFTER UPDATE ON UF

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'UF', CAST(NEW.uf AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_UF\_AFTER\_DELETE

AFTER DELETE ON UF

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'UF', CAST(OLD.uf AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_MUN

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_MUN\_AFTER\_INSERT

AFTER INSERT ON MUNICIPIO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'MUNICIPIO', CAST(NEW.municipio\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_MUN\_AFTER\_UPDATE

AFTER UPDATE ON MUNICIPIO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'MUNICIPIO', CAST(NEW.municipio\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_MUN\_AFTER\_DELETE

AFTER DELETE ON MUNICIPIO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'MUNICIPIO', CAST(OLD.municipio\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_EDR

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_EDR\_AFTER\_INSERT

AFTER INSERT ON ENDERECO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'ENDERECO', CAST(NEW.cod\_log AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_EDR\_AFTER\_UPDATE

AFTER UPDATE ON ENDERECO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'ENDERECO', CAST(NEW.cod\_log AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_EDR\_AFTER\_DELETE

AFTER DELETE ON ENDERECO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'ENDERECO', CAST(OLD.cod\_log AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_CLT

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_CLT\_AFTER\_INSERT

AFTER INSERT ON CLIENTE

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'CLIENTE', CAST(NEW.cliente\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_CLT\_AFTER\_UPDATE

AFTER UPDATE ON CLIENTE

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'CLIENTE', CAST(NEW.cliente\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_CLT\_AFTER\_DELETE

AFTER DELETE ON CLIENTE

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'CLIENTE', CAST(OLD.cliente\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_TPR

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_TPR\_AFTER\_INSERT

AFTER INSERT ON Transportadora

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'TRANSPORTADORA', CAST(NEW.transportadora\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_TPR\_AFTER\_UPDATE

AFTER UPDATE ON Transportadora

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'TRANSPORTADORA', CAST(NEW.transportadora\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_TPR\_AFTER\_DELETE

AFTER DELETE ON Transportadora

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'TRANSPORTADORA', CAST(OLD.transportadora\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_FML

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_FML\_AFTER\_INSERT

AFTER INSERT ON FAMILIA

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'FAMILIA', CAST(NEW.familia\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_FML\_AFTER\_UPDATE

AFTER UPDATE ON FAMILIA

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'FAMILIA', CAST(NEW.familia\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_FML\_AFTER\_DELETE

AFTER DELETE ON FAMILIA

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'FAMILIA', CAST(OLD.familia\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_LINHA

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_LINHA\_AFTER\_INSERT

AFTER INSERT ON LINHA

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'LINHA', CAST(NEW.linha\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_LINHA\_AFTER\_UPDATE

AFTER UPDATE ON LINHA

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'LINHA', CAST(NEW.linha\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_LINHA\_AFTER\_DELETE

AFTER DELETE ON LINHA

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'LINHA', CAST(OLD.linha\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_PDT

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_PDT\_AFTER\_INSERT

AFTER INSERT ON PRODUTO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'PRODUTO', CAST(NEW.produto\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_PDT\_AFTER\_UPDATE

AFTER UPDATE ON PRODUTO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'PRODUTO', CAST(NEW.produto\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_PDT\_AFTER\_DELETE

AFTER DELETE ON PRODUTO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'PRODUTO', CAST(OLD.produto\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_OPR

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_OPR\_AFTER\_INSERT

AFTER INSERT ON OPERACAO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'OPERACAO', CAST(NEW.operacao\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_OPR\_AFTER\_UPDATE

AFTER UPDATE ON OPERACAO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'OPERACAO', CAST(NEW.operacao\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_OPR\_AFTER\_DELETE

AFTER DELETE ON OPERACAO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'OPERACAO', CAST(OLD.operacao\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_USO

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_USO\_AFTER\_INSERT

AFTER INSERT ON USO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'USO', CAST(NEW.uso\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_USO\_AFTER\_UPDATE

AFTER UPDATE ON USO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'USO', CAST(NEW.uso\_numero AS CHAR));

END //

CREATE TRIGGER TG\_LOG\_USO\_AFTER\_DELETE

AFTER DELETE ON USO

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'USO', CAST(OLD.uso\_numero AS CHAR));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_PVI

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_PVI\_AFTER\_INSERT

AFTER INSERT ON PedidoVendaItem

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'PEDIDOVENDAITEM', CONCAT(CAST(NEW.serie AS CHAR),',',CAST(NEW.pedido\_numero AS CHAR)));

END //

CREATE TRIGGER TG\_LOG\_PVI\_AFTER\_UPDATE

AFTER UPDATE ON PedidoVendaItem

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'PEDIDOVENDAITEM', CONCAT(CAST(NEW.serie AS CHAR),',',CAST(NEW.pedido\_numero AS CHAR)));

END //

CREATE TRIGGER TG\_LOG\_PVI\_AFTER\_DELETE

AFTER DELETE ON PedidoVendaItem

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'PEDIDOVENDAITEM', CONCAT(CAST(OLD.serie AS CHAR),',',CAST(OLD.pedido\_numero AS CHAR)));

END //

-- -----------------------------------------------------

-- Trigger TG\_LOG\_PV

-- -----------------------------------------------------

CREATE TRIGGER TG\_LOG\_PV\_AFTER\_INSERT

AFTER INSERT ON PedidoVenda

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'INSERT', 'PEDIDOVENDA', CONCAT(CAST(NEW.serie AS CHAR),',',CAST(NEW.pedido\_numero AS CHAR)));

END //

CREATE TRIGGER TG\_LOG\_PV\_AFTER\_UPDATE

AFTER UPDATE ON PedidoVenda

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'UPDATE', 'PEDIDOVENDA', CONCAT(CAST(NEW.serie AS CHAR),',',CAST(NEW.pedido\_numero AS CHAR)));

END //

CREATE TRIGGER TG\_LOG\_PV\_AFTER\_DELETE

AFTER DELETE ON PedidoVenda

FOR EACH ROW

BEGIN

DECLARE v\_usuario VARCHAR(30);

SELECT USER() INTO v\_usuario;

CALL proc\_insere\_log(v\_usuario, 'DELETE', 'PEDIDOVENDA', CONCAT(CAST(OLD.serie AS CHAR),',',CAST(OLD.pedido\_numero AS CHAR)));

END //

DELIMITER ;

* 1. Queries

Para aprimorar a eficiência na obtenção de dados essenciais para o projeto, foram desenvolvidas views no banco de dados MySQL, que armazenam de maneira mais compacta o resultado de queries destinadas a responder às principais Key Performance Indicators (KPIs) utilizadas no projeto.

A primeira view, chamada "tempo\_entrega," concentra informações relevantes para calcular o tempo de entrega, ou seja, o período decorrido desde a saída do armazém até a chegada no cliente, expresso em dias. Essa view inclui dados como operação, uso, transportadora, grupo, município, UF, data de emissão, previsão de entrega, data de embarque, data de entrega e o tempo de entrega propriamente dito.

A segunda view, intitulada "tempo\_ciclo\_pedido," abrange dados relacionados ao tempo de ciclo do pedido, medido desde a efetivação da compra até a entrega, também em dias. Ela compartilha características semelhantes à primeira view, apresentando informações como operação, uso, transportadora, grupo, município, UF, data de emissão, previsão de entrega, data de embarque, data de entrega e o ciclo do pedido.

A terceira view, denominada "otif," visa calcular o On Time In Full (OTIF), uma métrica que representa a porcentagem de pedidos recebidos pelo cliente no prazo e nas quantidades acordadas. Ela inclui dados como cliente, número de pedidos recebidos, porcentagem de pedidos recebidos e o total de pedidos.

A quarta view, "otd," é responsável por calcular o On Time Delivery (OTD), que expressa a porcentagem de pedidos entregues ao cliente no prazo acordado em relação ao número total de pedidos.

A quinta view, "order\_fill\_rate," aborda a métrica Order Fill Rate, representando a porcentagem de pedidos atendidos completamente em relação ao número total de pedidos.

Por fim, a sexta view, "ticket\_medio\_pedido," calcula o ticket médio por pedido, sendo o valor total das vendas dividido pelo número de pedidos atendidos. Esta view inclui informações sobre operação, uso, transportadora, grupo, município, UF, mês de entrega e o ticket médio por pedido.

A última view, "info\_gerais," foi projetada para fornecer informações gerais para análises no PowerBI e notebooks. Ela abrange detalhes específicos sobre cada pedido, incluindo cliente, produto, operação, uso, família, linha, ordem de venda, transportadora, tipo de transporte, grupo, volume, valor da mercadoria, frete, seguro, acréscimos, peso líquido, peso bruto, datas de emissão, previsão de entrega, embarque e entrega, além de indicadores On Time, In Full e On Time/In Full, município, UF e região.

Essas views otimizam a recuperação de dados-chave para a análise de desempenho do projeto, facilitando a interpretação das KPIs e contribuindo para decisões mais informadas. Abaixo segue código fonte da criação das views.

-- 1-Tempo de entrega: Tempo decorrido da saída do armazém até a chegada no cliente em dias

ALTER VIEW tempo\_entrega AS

SELECT

opr.Nome as Operacao

,uso.uso as Uso

,tpd.nome as Transportadora

,grupo.nome as Grupo

,mun.municipio as Municipio

,uf.uf as UF

,pv.emitido\_em as 'Emitido em'

,pv.previsao\_entrega as 'Previsao Entrega'

,pv.dt\_embarque as 'Data Embarque'

,pv.dt\_entregue as 'Data Entregue'

,CONCAT(CONVERT((pv.dt\_entregue-pv.dt\_embarque),CHAR)," dia(s)")

as 'Tempo Entrega'

FROM

pedidovenda pv

LEFT JOIN operacao opr

ON pv.Operacao\_numero = opr.Operacao\_numero

LEFT JOIN uso

ON pv.Uso\_numero = uso.uso\_Numero

LEFT JOIN transportadora tpd

ON pv.transportadora\_numero = tpd.transportadora\_numero

LEFT JOIN endereco edc

ON pv.Cod\_log\_entrega = edc.Cod\_log

LEFT JOIN grupo

ON tpd.Grupo\_numero = grupo.Grupo\_numero

LEFT JOIN municipio mun

ON edc.Municipio\_numero = mun.Municipio\_numero

LEFT JOIN uf

ON mun.uf = uf.uf

ORDER BY pv.emitido\_em;

-- 2-Tempo de ciclo do pedido: Tempo decorrido desde a efetivação da compra

até a entrega em dias

ALTER VIEW tempo\_ciclo\_pedido AS

SELECT

opr.Nome as Operacao

,uso.uso as Uso

,tpd.nome as Transportadora

,grupo.nome as Grupo

,mun.municipio as Municipio

,uf.uf as UF

,pv.emitido\_em as 'Emitido em'

,pv.previsao\_entrega as 'Previsao Entrega'

,pv.dt\_embarque as 'Data Embarque'

,pv.dt\_entregue as 'Data Entregue'

,CONCAT(CONVERT((pv.dt\_entregue-pv.emitido\_em),CHAR)," dia(s)") as

'Ciclo Pedido'

FROM

pedidovenda pv

LEFT JOIN operacao opr

ON pv.Operacao\_numero = opr.Operacao\_numero

LEFT JOIN uso

ON pv.Uso\_numero = uso.uso\_Numero

LEFT JOIN transportadora tpd

ON pv.transportadora\_numero = tpd.transportadora\_numero

LEFT JOIN endereco edc

ON pv.Cod\_log\_entrega = edc.Cod\_log

LEFT JOIN grupo

ON tpd.Grupo\_numero = grupo.Grupo\_numero

LEFT JOIN municipio mun

ON edc.Municipio\_numero = mun.Municipio\_numero

LEFT JOIN uf

ON mun.uf = uf.uf

Order by pv.emitido\_em;

-- 3-On Time In Full (OTIF - Pedidos Completos e no Prazo): Número de

pedidos recebidos pelo cliente no prazo e quantidades acordadas / número

total de pedidos, em porcentagem.

ALTER VIEW otif AS

SELECT

clt.cliente as 'Cliente'

,count(pv.pedido\_numero) as 'Pedidos Recebidos'

,ROUND((count(pv.pedido\_numero)\*100)/totalPed.total\_pedidos,2) as

'Pedidos Recebidos %'

,totalPed.total\_pedidos as 'Total Pedidos'

FROM

pedidovenda pv

LEFT JOIN cliente clt

ON pv.cliente\_numero = clt.cliente\_numero

LEFT JOIN

(SELECT

clt.cliente\_numero as NumeroCliente

,clt.cliente as Cliente

,count(pv.pedido\_numero) as total\_pedidos

FROM

pedidovenda pv

LEFT JOIN cliente clt

ON pv.cliente\_numero = clt.cliente\_numero

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero = pvi.pedido\_numero

GROUP BY clt.cliente\_numero, clt.cliente) as totalPed

ON clt.cliente\_numero = totalPed.NumeroCliente

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero = pvi.pedido\_numero

WHERE pv.dt\_entregue <= pv.previsao\_entrega

AND pv.OnTime\_InFull = 0

GROUP BY clt.cliente, totalPed.total\_pedidos

ORDER BY clt.cliente;

-- 4-On time delivery (OTD): Número de pedidos entregues ao cliente no prazo acordado / número total de pedidos, em porcentagem.

CREATE VIEW otd AS

SELECT

clt.cliente as 'Cliente'

,count(pv.pedido\_numero) as 'Pedidos Recebidos'

,ROUND((count(pv.pedido\_numero)\*100)/totalPed.total\_pedidos,2) as 'Pedidos Recebidos %'

,totalPed.total\_pedidos as 'Total Pedidos'

FROM

pedidovenda pv

LEFT JOIN cliente clt

ON pv.cliente\_numero = clt.cliente\_numero

LEFT JOIN

(SELECT

clt.cliente\_numero as NumeroCliente

,clt.cliente as Cliente

,count(pv.pedido\_numero) as total\_pedidos

FROM

pedidovenda pv

LEFT JOIN cliente clt

ON pv.cliente\_numero = clt.cliente\_numero

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero =

pvi.pedido\_numero

GROUP BY clt.cliente\_numero, clt.cliente) as totalPed

ON clt.cliente\_numero = totalPed.NumeroCliente

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero = pvi.pedido\_numero

WHERE pv.dt\_entregue <= pv.previsao\_entrega

-- AND pv.OnTime\_InFull = 0

GROUP BY clt.cliente, totalPed.total\_pedidos

ORDER BY clt.cliente;

-- 5-Order Fill Rate: Número de pedidos atendidos completamente / Número total de pedidos, em porcentagem

CREATE VIEW order\_fill\_rate AS

SELECT

Cliente

,Pedidos\_Recebidos

,Pedidos\_Recebidos\_Porc

,TotalPedidos

FROM

(SELECT

clt.cliente as 'Cliente'

,count(pv.pedido\_numero) as Pedidos\_Recebidos

,ROUND((count(pv.pedido\_numero)\*100)/totalPed.total\_pedidos,2) as Pedidos\_Recebidos\_Porc

,totalPed.total\_pedidos as TotalPedidos

FROM

pedidovenda pv

LEFT JOIN cliente clt

ON pv.cliente\_numero = clt.cliente\_numero

LEFT JOIN

(SELECT

clt.cliente\_numero as NumeroCliente

,clt.cliente as Cliente

,count(pv.pedido\_numero) as total\_pedidos

FROM

pedidovenda pv

LEFT JOIN cliente clt

ON pv.cliente\_numero = clt.cliente\_numero

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero =

pvi.pedido\_numero

GROUP BY clt.cliente\_numero, clt.cliente) as totalPed

ON clt.cliente\_numero = totalPed.NumeroCliente

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero = pvi.pedido\_numero

WHERE pv.dt\_entregue <= pv.previsao\_entrega

AND pv.OnTime\_InFull = 0

GROUP BY clt.cliente, totalPed.total\_pedidos

ORDER BY clt.cliente) as result

WHERE

Pedidos\_Recebidos = TotalPedidos;

-- Order Fill Rate (OFR): o tempo gasto no processamento interno de um pedido de compra, desde a separação até a expedição.

ALTER VIEW ofr\_sep\_exp AS

SELECT

opr.Nome as Operacao

,uso.uso as Uso

,tpd.nome as Transportadora

,grupo.nome as Grupo

,mun.municipio as Municipio

,uf.uf as UF

,pv.emitido\_em as 'Emitido em'

,pv.dt\_embarque as 'Data Embarque'

,CONCAT(CONVERT((pv.dt\_entregue-pv.emitido\_em),CHAR)," dia(s)") as 'Tempo Processamento'

FROM

pedidovenda pv

LEFT JOIN operacao opr

ON pv.Operacao\_numero = opr.Operacao\_numero

LEFT JOIN uso

ON pv.Uso\_numero = uso.uso\_Numero

LEFT JOIN transportadora tpd

ON pv.transportadora\_numero = tpd.transportadora\_numero

LEFT JOIN endereco edc

ON pv.Cod\_log\_entrega = edc.Cod\_log

LEFT JOIN grupo

ON tpd.Grupo\_numero = grupo.Grupo\_numero

LEFT JOIN municipio mun

ON edc.Municipio\_numero = mun.Municipio\_numero

LEFT JOIN uf

ON mun.uf = uf.uf

ORDER BY pv.emitido\_em;

-- Ticket médio por pedido: valor total das vendas dividido pelo numero de pedidos atendidos

ALTER VIEW ticket\_medio\_pedido AS

SELECT

opr.Nome as Operacao

,uso.uso as Uso

,tpd.nome as Transportadora

,grupo.nome as Grupo

,mun.municipio as Municipio

,uf.uf as UF

,CASE MONTH(pv.dt\_entregue)

WHEN 1 THEN 'Janeiro'

WHEN 2 THEN 'Fevereiro'

WHEN 3 THEN 'Março'

WHEN 4 THEN 'Abril'

WHEN 5 THEN 'Maio'

WHEN 6 THEN 'Junho'

WHEN 7 THEN 'Julho'

WHEN 8 THEN 'Agosto'

WHEN 9 THEN 'Setembro'

WHEN 10 THEN 'Outubro'

WHEN 11 THEN 'Novembro'

WHEN 12 THEN 'Dezembro' END as Mes\_Entregue

-- ,pvi.preco\_total as 'Valor Total Venda'

,ROUND(SUM(pvi.preco\_total/total\_ped\_atendidos.Pedido),2) as

'Ticket Medio por Pedido'

FROM

pedidovenda pv

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero = pvi.pedido\_numero

INNER JOIN

(SELECT

pv.pedido\_numero as ID

,count(pv.pedido\_numero) as Pedido

FROM

pedidovenda pv

WHERE

pv.dt\_entregue <= pv.previsao\_entrega

AND pv.OnTime\_InFull = 0

GROUP BY

pv.pedido\_numero) as total\_ped\_atendidos

ON pv.pedido\_numero = total\_ped\_atendidos.ID

LEFT JOIN operacao opr

ON pv.Operacao\_numero = opr.Operacao\_numero

LEFT JOIN uso

ON pv.Uso\_numero = uso.uso\_Numero

LEFT JOIN transportadora tpd

ON pv.transportadora\_numero = tpd.transportadora\_numero

LEFT JOIN endereco edc

ON pv.Cod\_log\_entrega = edc.Cod\_log

LEFT JOIN grupo

ON tpd.Grupo\_numero = grupo.Grupo\_numero

LEFT JOIN municipio mun

ON edc.Municipio\_numero = mun.Municipio\_numero

LEFT JOIN uf

ON mun.uf = uf.uf

GROUP BY

opr.Nome

,uso.uso

,tpd.nome

,grupo.nome

,mun.municipio

,uf.uf

,Mes\_Entregue;

-- ORDER BY MONTH(Mes\_Entregue);

-- Info gerais : Para PowerBI e analises notebook

ALTER VIEW info\_gerais AS

SELECT

pv.pedido\_numero as 'Pedido Numero'

,cli.cliente as Cliente

,prd.nome as Produto

,opr.Nome as Operacao

,uso.uso as Uso

,fam.nome as Familia

,linha.nome as Linha

,pv.ordem\_venda as 'Ordem Venda'

,tpd.nome as Transportadora

,pv.transporte as 'Tipo Transporte'

,grupo.nome as Grupo

,pv.volume as Volume

,ROUND(pv.valor\_mercadoria,2) as 'Valor Mercadoria (R$)'

,ROUND(pv.frete,2) as 'Frete (R$)'

,ROUND(pv.seguro,2) as 'Seguro (R$)'

,ROUND(pv.acrescimos,2) as 'Acrescimos (R$)'

,ROUND(pv.peso\_liquido,2) as 'Peso Liquido (Kg)'

,ROUND(pv.peso\_bruto,2) as 'Peso Bruto (Kg)'

,pv.emitido\_em as 'Emitido em'

,pv.previsao\_entrega as 'Previsao Entrega'

,pv.dt\_embarque as 'Data Embarque'

,pv.dt\_entregue as 'Data Entregue'

,pv.ontime as 'On Time'

,pv.infull as 'In Full'

,pv.ontime\_infull as 'On Time/In Full'

,mun.municipio as Municipio

,uf.uf as UF

,uf.regiao as Regiao

FROM

pedidovenda pv

LEFT JOIN cliente cli

ON pv.cliente\_numero = cli.cliente\_numero

LEFT JOIN operacao opr

ON pv.Operacao\_numero = opr.Operacao\_numero

LEFT JOIN uso

ON pv.Uso\_numero = uso.uso\_Numero

LEFT JOIN transportadora tpd

ON pv.transportadora\_numero = tpd.transportadora\_numero

LEFT JOIN endereco edc

ON pv.Cod\_log\_entrega = edc.Cod\_log

LEFT JOIN grupo

ON tpd.Grupo\_numero = grupo.Grupo\_numero

LEFT JOIN municipio mun

ON edc.Municipio\_numero = mun.Municipio\_numero

LEFT JOIN uf

ON mun.uf = uf.uf

LEFT JOIN pedidovendaitem pvi

ON pv.serie = pvi.serie AND pv.pedido\_numero = pvi.pedido\_numero

LEFT JOIN produto prd

ON pvi.produto\_numero = prd.produto\_numero

LEFT JOIN familia fam

ON prd.familia\_numero = fam.familia\_numero

LEFT JOIN linha

ON prd.linha\_numero = linha.linha\_numero

Order by cli.cliente,pv.emitido\_em;

describe info\_gerais;

* 1. Processo de Integração de Modelo de Dados

As bases de dados de um cliente devem ser entendidas como peças de um modelo de dados maior, o qual busca representar o negócio do cliente, independente da tecnologia de armazenamento adotada. A construção de modelos de dados ou processo de modelagem de dados, como também pode ser chamado, privilegiando a construção de modelos de dados altamente integrados apresenta os seguintes benefícios:

* 1. Entrada de dados em pontos únicos, evitando a necessidade de sincronização cruzada entre modelos de dados;
  2. A redundância de dados é evitada, economizando espaço de armazenamento;
  3. Os modelos de dados são mantidos coesos, coerentes e responsáveis por um subconjunto dos dados que fazem parte do negócio do cliente.

Desta forma, prega-se que a integração dos modelos de dados deve ser um dos principais objetivos no processo de construção de cada modelo de dado. De fato, a análise dos dados que comporão um novo modelo de dados e como ele pode ser incorporado ao modelo de dados do cliente, deve ser uma das primeiras atividades do processo de modelagem de dados. Dentre as principais atividades quanto ao processo de integração na construção de modelos de dados, destacam-se:

(1) Novos atributos e entidades devem ser avaliadas quanto a que assunto ou negócio do cliente estes pertencem e se estes já são tratados em alguns dos modelos de dados já existentes.

(2) Uma vez entendido que de fato representam um conceito não abordado em algum dos modelos de dados já existentes, as novas entidades e atributos devem ser avaliados quanto ao seu uso pelos demais modelos de dados existentes do cliente. Caso possam ser utilizados por vários dos demais modelos, estes devem ser incluídos em um dos modelos de dados compartilhados do cliente ou até cogitada sua incorporação ao Modelo de Dados Corporativo caso sejam de interesse de outros clientes.

Uma exceção a ser observada quanto a evitar a redundância de dados, diz respeito às aplicações onde a comunicação entre as fontes de dados seja um gargalo tecnológico e o desempenho seja um requisito crítico. Nesta situação se considera a clonagem dos dados como uma técnica adequada. No entanto, este artifício deve ser utilizado com cautela, não permitindo que se crie outro ponto de entrada de informações para os dados clonados e jamais permitindo que estes sofram atualizações.

No processo de integração do modelo de dados, foi adotada uma abordagem que alinha a representação do negócio do cliente em um modelo coeso e integrado. Para efetuar essa integração, utilizamos arquivos CSV como meio para carregar dados nas tabelas do banco MySQL, aproveitando a flexibilidade desse formato amplamente suportado. A escolha de CSVs como fonte de dados proporcionou uma solução eficiente e versátil para a entrada de dados, permitindo uma fácil manipulação e transformação das informações antes de serem carregadas no banco.

Após a carga dos dados no MySQL, estabelecemos uma conexão entre o banco e duas ferramentas essenciais para análises avançadas: o Power BI e o Jupyter Notebook. No Power BI, exploramos a conectividade direta ao banco MySQL para criar visualizações dinâmicas e relatórios interativos. Essa integração permitiu uma análise em tempo real dos dados, possibilitando uma compreensão profunda das métricas e indicadores essenciais.

Além disso, para análises mais avançadas e modelagem estatística, o Jupyter Notebook foi integrado ao banco MySQL. Utilizamos bibliotecas como Pandas e SQLalchemy para manipular e analisar dados diretamente do banco, proporcionando flexibilidade na execução de consultas complexas e na aplicação de algoritmos de machine learning.

A integração dessas ferramentas, desde a carga de dados via arquivos CSV até a exploração visual no Power BI e análises avançadas no Jupyter Notebook, proporcionou uma visão abrangente e integrada do modelo de dados. Essa abordagem não apenas atendeu aos requisitos de coesão e integração do modelo, mas também facilitou a análise de KPIs e métricas essenciais para o sucesso do projeto.

1. ANÁLISE DE DADOS
   1. Dashboards Power BI e Excel

O tratamento de dados no Excel iniciou-se com a formalização do formato específico de cada coluna (como número, moeda, etc) e estruturação de todos os dados em tabelas com nomes descritivos para facilitar as associações. Foi utilizado fórmulas como PROCV para fazer a relação entre diferentes tabelas, relacionando códigos com a descrição; a fórmula CONCAT para fazer a junção de informações como: endereço, logradouro, município e dados; a fórmula SE para identificar os pedidos entregues no prazo ou em atraso; adicionamos colunas calculadas para calcular o lucro do produto, tempo de entrega para o cliente e etc); foi utilizada formatação condicional, alterando a aparência das células (cores) com o intuito de tornar os valores mais fáceis de identificar; utilizamos diversas tabelas dinâmicas para consolidar e calcular os dados, estabelecendo relações entre diferentes campos como região, produto, área de trabalho do cliente, etc), foi implementado segmentação de dados para facilitar a filtragem de dados, permitindo uma análise mais detalhada por período, categoria, etc).

Por último, foi desenvolvido um dashboard dinâmico que destaca KPI´s como Entregas por Transportadoras e Pedidos OTIF por Transportadora, Custo Total das Entregas (Valor Total dos Pedidos), Custo das Mercadorias (Valor Total dos Produtos nos Pedidos), Custo do Frete, Custo do Seguro, Ticket Médio das Entregas e Porcentagem Gasto com Transporte, com representações de gráficos de pizza, de colunas, de barras e gráfico de combinação (colunas e linha).

Os filtros utilizados no desenvolvimento do Dashboard do Excel foram: uma linha do tempo do pedido emitido (data de emissão do documento), tipo do pedido, UF da entrega, tipo de custo do transporte, nome da transportadora, tipo da operação, tipo do grupo do cliente e tipo de uso.

Essas alterações fortaleceram a arquitetura de dados no Excel, proporcionando insights valiosos e uma visualização mais clara e atualizada das informações. A Figura 3 abaixo representa o Dashboard em Excel voltado para uma análise financeira das entregas de produtos poliméricos.

Figura 3.Dashboard - Gráficos gerados no Excel.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Fonte: próprios autores.

Escrever 1 parágrafo do Power BI

* 1. Funções e Filtros Excel e Power BI

**Filtros do excel no tópico 6.1**

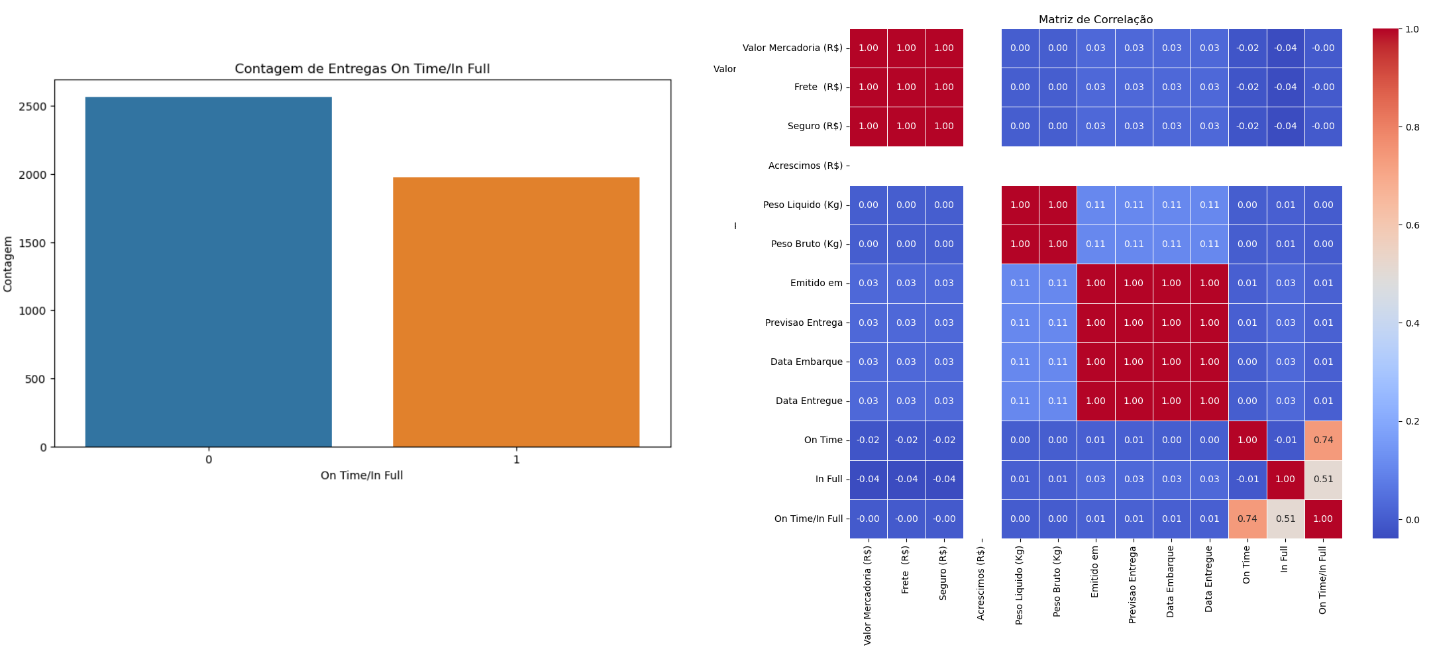
* 1. Criação do Notebook

O notebook foi criado no Jupyter em código Python para a realização de análises exploratórias em um banco de dados MySQL chamado "polimero". O intuito do uso do notebook foi a criação de uma análise de dados avançadas com gráficos e visualizações incorporados de uma maneira rápida e intuitiva para avaliação dos dados utilizados.

As bibliotecas utilizadas foram pandas, matplotlib, seaborn e scikit-learn. Foram realizadas análises de estatística descritiva e outras análises mais complexas com o uso de algoritmos de machine learning, como Random Forest Classifier e Regressão Linear, para explorar e compreender melhor os dados armazenados no banco de dados.

Além disso, diversas visualizações foram realizadas usando Seaborn, explorando a distribuição de operações, correlações entre variáveis, contagem de entregas, histograma de lucro por mês, gráfico de dispersão entre quantidade de vendas e grupos por operação, boxplot do valor das mercadorias por tipo de transporte e matriz de Na Figuraxxx são apresentados alguns dos gráficos gerados dentro do notebook.

Figura 4. Gráficos gerados no Notebook.



Fonte: próprios autores.

* 1. Bibliotecas

Para a criação do notebook foram utilizadas as seguintes bibliotecas:

1. **pandas (pd):** para manipulação e análise de dados.
2. **matplotlib.pyplot (plt):** biblioteca gráfica utilizada para criar visualizações estáticas e interativas em Python.
3. **seaborn (sns):** visualização de dados baseada no Matplotlib, fornece uma interface de alto nível para criar gráficos estatísticos com visual diferenciado.
4. **sklearn.model\_selection:** Faz parte da biblioteca scikit-learn (sklearn) oferece ferramentas para dividir conjuntos de dados em subconjuntos de treino e teste
5. **sklearn.ensemble:** Dentro do scikit-learn, a subbiblioteca ensemble oferece métodos para construção de modelos de conjunto. Neste caso, é utilizado o RandomForestClassifier, que implementa um classificador de floresta aleatória.
6. **sklearn.linear\_model:** Esta parte do scikit-learn fornece ferramentas para modelagem linear. O LinearRegression é um modelo de regressão linear, utilizado para análise de regressão.
7. **sklearn.metrics:** Esta parte do scikit-learn contém métricas de avaliação de modelos. As funções accuracy\_score, classification\_report e confusion\_matrix são usadas para avaliar o desempenho de modelos de classificação.
8. CONCLUSÕES
9. REFERÊNCIAS

CAVALCANTI, A.S.B. **VERIFICAÇÃO DO NÍVEL DO SERVIÇO DE ENTREGA DE PRODUTOS DO SETOR DE DISTRIBUIÇÃO DE UMA EMPRESA DE BEBIDAS**. Trabalho de Conclusão de Curso (TCC) do curso de Administração, Universidade Federal da Paraíba, João Pessoa, 2015.

CHAGAS, R.C.C.; SEGGER, R.L.P. **Indicadores de desempenho logístico: Estudo de caso em uma empresa transportadora de carga.** Trabalho de Conclusão de Curso (TCC) do curso de Engenharia de Transportes, Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Goiânia, 2021.

FREITAS, D.F. **ANÁLISE DE FERRAMENTAS DE BUSINESS INTELLIGENCE: COMO FERRAMENTAS OPEN SOURCE PODEM REDUZIR CUSTOS EM NEGÓCIOS.** Monografia do curso de Especialização em Gestão da Tecnologia da Informação e Comunicação, Universidade Tecnológica Federal do Paraná, Curitiba, 2017.

MORAES, M.C. IM**PLEMENTAÇÃO E GESTÃO DE UM INDICADOR DE DESEMPENHO DE NÍVEL DE SERVIÇO PARA ATENDIMENTO DE PEDIDOS EM UMA DISTRIBUIÇÃO DE MEDICAMENTOS E MATERIAIS HOSPITALARES NO ESTADO DA PARAÍBA**.Trabalho de Conclusão de Curso (TCC) do curso de Engenharia da Produção, João Pessoa, 2019.

SILVA, S. F. P. **Materiais e logística.** Curso Superior de Tecnologia em Gestão Pública. Florianópolis: IF-SC, 2010. 2. ed. Programa Universidade Aberta do Brasil (UAB).

SILVA, A.A.A.; SILVA, J.A. A **UTILIZAÇÃO DO INDICADOR OTIF COMO FERRAMENTA PARA A SATISFAÇÃO DO CLIENTE: UM ESTUDO EM UMA EMPRESA DA INDÚSTRIA CIMENTEIRA**. Trabalho de Conclusão de Curso (TCC) do curso de Administração, Universidade Federal Rural da Amazônia,Capanema, 2019.

SIMÕES, E. A.; MOURA, V.A.; OKANO, M.T. Tecnologia aplicada à logística de distribuição para melhoria do otd (on time delivery). **Braz. J. of Develop**., Curitiba, v. 5, n. 9, p. 16574-16603, 2019.

SITEWARE.[**GESTÃO ESTRATÉGICA**](https://www.siteware.com.br/blog/categoria/gestao-estrategica/): Saiba o que é BI Business Intelligence e para que serve!

Disponível em: <https://www.siteware.com.br/blog/gestao-estrategica/o-que-e-bi-business-intelligence/>

Acesso em: 10 Dez 2023.