## **1. Introdução**

Este relatório documenta todo o processo de criação e ajustes de views no MySQL para integrar dados de rotas, fábricas e clientes, além da organização do projeto utilizando ferramentas de gestão de tarefas e controle de versão. O objetivo foi permitir a análise de produtividade dos veículos, evolução de custos por km e unidade transportada, tudo integrado com o Power BI. O documento também inclui o uso do **Jira Software** para a organização das tarefas e do backlog, bem como do **GitHub** para a estruturação da equipe e controle de versões do trabalho.

## **2. Estrutura Inicial das Tabelas**

As três tabelas principais utilizadas neste projeto foram:

* **Rotas**: Informações sobre as rotas realizadas pelos veículos, como pedidos, data, valor do frete e quantidade transportada.
* **Fabricas**: Informações sobre as fábricas, incluindo localizações (latitude e longitude).
* **Clientes**: Informações sobre os clientes, com dados de localização e identificação.

### **2.1. Colunas das Tabelas**

**Tabela Rotas:**

* Veiculo: Identificação do veículo.
* DtEmissao: Data de emissão do pedido.
* COFabrica: Código da fábrica.
* COCliente: Código do cliente.
* QtdTransp: Quantidade de carga transportada (em quilos).
* Qtdpallets: Quantidade de pallets transportados.
* VlrFrete: Valor total do frete.
* ï»¿Pedido: Identificador único de cada pedido.

**Tabela Fabricas:**

* Fabrica: Código da fábrica.
* NOMUN: Nome do município da fábrica.
* LAT: Latitude da fábrica.
* LONG: Longitude da fábrica.

**Tabela Clientes:**

* Cliente: Código do cliente.
* MUN: Nome do município do cliente.
* LAT: Latitude do cliente.
* LONG: Longitude do cliente.

## **3. Organização do Projeto e Ferramentas Utilizadas**

### **3.1. Utilização do Jira Software**

Durante o desenvolvimento deste projeto, utilizamos o **Jira Software** para organizar e gerenciar o backlog do produto. As atividades foram divididas em tarefas, e cada uma delas foi associada a um sprint específico para garantir o progresso contínuo do trabalho. O backlog foi utilizado para:

* Organizar as tarefas de criação das views SQL.
* Controlar a resolução de bugs e ajustes nas queries.
* Acompanhar o progresso de integração com o Power BI.
* Definir prazos e responsáveis por cada tarefa.

O uso do **Jira** permitiu que a equipe tivesse uma visão clara do progresso do projeto e facilitou a gestão de mudanças e melhorias ao longo do processo.

### **3.2. Utilização do GitHub**

Além da gestão de tarefas, o **GitHub** foi utilizado para a estruturação do código, permitindo controle de versão, colaboração entre os membros da equipe e documentação do trabalho. As principais funcionalidades utilizadas no GitHub foram:

* **Repositório do código SQL**: Todas as queries e scripts SQL foram versionados no GitHub para garantir rastreabilidade e controle de mudanças.
* **Documentação do projeto**: A estrutura e descrição das views, bem como as funcionalidades implementadas, foram documentadas diretamente no repositório.
* **Colaboração em equipe**: Utilizamos branches para desenvolvimento paralelo e pull requests para revisão de código e validação das alterações.

## **4. Descrição do Processo e Funções Utilizadas**

O processo foi dividido em várias etapas, cada uma com a sua respectiva função e cálculo. O principal foco foi criar views que pudessem ser facilmente utilizadas no Power BI para gerar relatórios dinâmicos.

### **4.1. Criação de Views para Análises**

Diversas views foram criadas para fornecer insights sobre a produtividade dos veículos, evolução dos custos por km e por unidade transportada. As principais métricas foram baseadas em cálculos como o custo por quilômetro, produtividade em relação ao frete e evolução dos custos ao longo do tempo.

Fórmula de Haversine para calcular a distância entre fábricas e clientes com base nas coordenadas de latitude e longitude:

scss

Copiar código

DISTÂNCIA = 6371 \* ACOS(COS(RADIANS(fabricas.LAT)) \* COS(RADIANS(clientes.LAT)) \*   
COS(RADIANS(clientes.LONG) - RADIANS(fabricas.LONG)) + SIN(RADIANS(fabricas.LAT)) \*   
SIN(RADIANS(clientes.LAT)))

### **4.2. Funções Utilizadas**

Abaixo, listamos algumas funções chave utilizadas nas queries:

* **SUM()**: Função de agregação para somar valores de frete, carga transportada, entre outros.
* **COUNT()**: Contou o número de pedidos realizados.
* **DATE\_FORMAT()**: Utilizada para agrupar as análises por mês e ano.
* **ACOS(), COS(), SIN(), RADIANS()**: Funções trigonométricas usadas no cálculo da distância entre as fábricas e clientes.

## **5. Problemas Encontrados e Soluções**

Durante o processo, diversos problemas foram encontrados e solucionados:

* **Erro GROUP BY**: Para resolver o problema de compatibilidade com o modo ONLY\_FULL\_GROUP\_BY, garantimos que todas as colunas não agregadas estavam no GROUP BY.
* **Erro Unknown column 'rotas.Pedido'**: Este erro foi resolvido identificando que a coluna correta era ï»¿Pedido e ajustando as consultas de acordo.
* **Uso de variáveis não permitidas em views**: O cálculo da distância utilizava a variável @earth\_radius\_km, que não é permitido em views. Isso foi resolvido substituindo a variável por seu valor fixo (6371 km).

## **6. Resultado Final**

Após a criação das views e resolução dos problemas, o resultado final foi uma série de views capazes de gerar as análises necessárias para o Power BI. Essas views incluem produtividade mensal dos veículos, custos por km e por unidade transportada, filtrando por fábricas. As views permitem a criação de gráficos dinâmicos e interativos no Power BI.

## **7. Conclusão**

O processo de criação e ajuste das views foi crucial para fornecer análises completas e interativas no Power BI. O uso do **Jira Software** para organizar o backlog e do **GitHub** para a estruturação e colaboração da equipe foram essenciais para garantir o sucesso do projeto. Com a resolução dos bugs e ajustes no agrupamento de dados, conseguimos garantir que as métricas geradas fossem precisas e adequadas às necessidades da análise.