**INTRODUÇÃO**

O Delivery Center, com seus diversos hubs operacionais espalhados pelo Brasil, é uma plataforma que integra lojistas e marketplaces, criando um ecossistema saudável para vendas de produtos e comidas no varejo brasileiro. Atualmente, possui um cadastro (catálogo + cardápio) com mais de 900 mil itens, e milhares de pedidos e entregas são operacionalizados diariamente com uma rede de milhares de lojistas e entregadores parceiros espalhados por todas as regiões do país.

Tudo isso gera dados continuamente. Diante desse cenário, o negócio está cada vez mais orientado por dados (data driven), ou seja, utiliza dados para tomar decisões. Com uma visão de futuro, o Delivery Center reconhece que o uso inteligente dos dados pode ser um grande diferencial no mercado.

Este é o contexto apresentado, juntamente com um desafio que propõe a aplicação de conhecimentos técnicos para resolver problemas cotidianos de uma equipe de dados. Os dados disponíveis são representados em um modelo de dados no padrão floco de neve (snowflake). Esse modelo representa, de forma fictícia, dados de pedidos e entregas processados pelo Delivery Center entre os meses de janeiro a abril de 2021.

[modelodedados.jpg]

Nota-se que este é um modelo lógico e está fisicamente disponível em datasets no formato CSV, ou seja, cada dataset disponível representa uma tabela no esquema mencionado. Os dados não possuem a completude de toda a operação do Delivery Center, e algumas informações foram anonimizadas devido ao tratamento conforme a Lei Geral de Proteção de Dados (LGPD).

O dataset pode ser encontrado em:

https://www.kaggle.com/datasets/nosbielcs/brazilian-delivery-center

**DESENVOLVIMENTO**

Antes de iniciar as querys e o schema da nova base de dados que seria criada, foi utilizado Google Collab com a biblioteca pandas para realizar uma análise exploratória dos dados para entender melhor a estrutura, conteúdo e a qualidade dos dados fornecidos. Apesar da possibilidade de realizar EDA pelo PostGres, por se tratar de um sistema de gerenciamento de banco de dados relacional e não uma biblioteca específica, as diversas consultas básicas e de estatística descritiva que seriam realizadas ocupariam muito espaço e tomariam muito tempo do projeto.

Funções utilizadas neste processo:

• pd.read\_csv

• df.info()

• df.describe()

• df.isnull().sum()

• df2.duplicated().sum()

Então uma tabela para cada arquivo csv foi criada, formando nosso schema:

* channels: Possui informações sobre os canais de venda (marketplaces) onde são vendidos os good e food dos lojistas.
* Deliveries: Possui informações sobre as entregas realizadas pelos entregadores parceiros.
* drivers: Possui informações sobre os entregadores parceiros. Eles ficam nos hubs e toda vez que um pedido é processado, são eles que realizam as entregas nas casas dos consumidores.
* hubs: Possui informações sobre os hubs do Delivery Center. Entenda que os Hubs são os centros de distribuição dos pedidos e é dali que saem as entregas.
* orders: Possui informações sobre as vendas processadas através da plataforma do Delivery Center.
* payments: Possui informações sobre os pagamentos realizados ao Delivery Center.
* stores: Possui informações sobre os lojistas. Eles utilizam a Plataforma do Delivery Center para vender seus itens (good e/ou food) nos marketplaces.
* Completando o schema, foram desenvolvidos views, índices e um trigger que foram adicionados e serão citados posteriormente.

Os dados presentes no arquivo csv foram inseridos nestas tabelas utilizando funções do pgAdmin 4.

Assim, o primeiro problema do projeto surgiu pois não era possível inserir os dados da tabela orders em algumas colunas (order\_moment) como TIMESTAMP devido aos seus valores que estavam dispostos no formato MM/DD/YYYY, sendo o correto YYYY/MM/DD.

A solução foi inserir os valores como strings utilizando VARCHAR para depois então serem alterados para TIMESTAMP. Este processo foi importante para o projeto pois as colunas ‘order\_moment’ foram utilizadas em diversas consultas e funções.

Após análise geral do dataset o projeto ficou dividido em 4 etapas:

Análise dos Pedidos e Entregas

Análise de Pagamentos

Análise dos Hubs

Análise dos Entregadores

Análise dos Pedidos e Entregas

Aqui foram elaboradas consultas simples e gerais como: calcular o total de entregas e pedidos. E também funções mais complexas que auxiliam a calcular o tempo levado para as entregas independente da categoria inserida pelo usuário.

As funções nesta etapa foram criadas com o objetivo de reduzir ao máximo as redundâncias das pesquisas, deixando o projeto mais limpo e organizado. Exemplo: Para calcular a quantidade de pedidos por canal e quantidade de pedidos por segmento das lojas as consultas seriam praticamente iguais.

Análise de Pagamentos

Esta etapa foi nomeada assim devido ao nome da tabela ser ‘pagamentos’ mas poderia facilmente ser substituída pelo nome ‘receitas’, pois além de avaliar os métodos de pagamentos mais utilizados o foco real foi aplicar consultas com o intuito de calcular o faturamento, comparando-o entre períodos e diferentes categorias.

Análise dos Hubs

O foco desta etapa foi analisar a capacidade de atendimento dos hubs juntamente com sua eficiência.

Análise dos Entregadores

O objetivo foi analisar a quantidade de entregas e também a eficiência de cada entregador.

Após as análises, foi verificada a possibilidade de otimizar o projeto para além das consultas e funções que foram criadas. Optou-se também pela inserção de ÍNDICES, VIEWs e um TRIGGER.

A decisão de criação dos índices se baseou nas consultas mais utilizadas e o auxílio da função EXPLAIN para observar onde performance estava sendo perdida em cada uma. Importante observar que em projetos grandes a criação dos índices deve levar em consideração a memória consumida.

As views foram criadas com a intenção de serem utilizadas como uma ferramenta de pesquisa de pedidos e entregas, contendo as informações gerais mais relevantes para tal.

O trigger elaborado simula a inserção de novos dados em um sistema real, especificamente no momento do pagamento, onde o cliente ao finalizar o pagamento alterando o status para ‘PAGO’, o status do pedido também seria alterado para ‘FINALIZADO’, impossibilitando com que o pedido seja concluído antes do pagamento ser realizado.

**INTRODUCTION**

With its various operational hubs scattered across Brazil, the Delivery Center is a platform that integrates retailers and marketplaces, creating a healthy ecosystem for the sale of products and food in the Brazilian retail market. Currently, it has a catalog (product catalog + menu) with over 900,000 items, and thousands of orders and deliveries are processed daily with a network of thousands of partner retailers and delivery people spread across all regions of the country.

All of this generates data continuously. In this scenario, the business is increasingly data-driven, meaning it uses data to make decisions. With a vision for the future, the Delivery Center recognizes that intelligent use of data can be a major market differentiator.

This is the context presented, along with a challenge that proposes the application of technical knowledge to solve everyday problems for a data team. The available data is represented in a data model based on the snowflake schema. This model represents, in a fictional way, data from orders and deliveries processed by the Delivery Center between January and April 2021.

[modelodedados.jpg]

Note that this is a logical model and is physically available in datasets in CSV format, meaning that each available dataset represents a table in the aforementioned schema. The data does not have the completeness of the entire Delivery Center operation, and some information has been anonymized due to treatment in accordance with the General Data Protection Law (LGPD).

The dataset can be found at:

<https://www.kaggle.com/datasets/nosbielcs/brazilian-delivery-center>

**DEVELOPMENT**

Before starting the queries and the schema of the new database that would be created, Google Collab with the pandas library was used to perform an exploratory data analysis to better understand the structure, content, and quality of the data provided. Despite the possibility of performing EDA through PostGres, since it is a relational database management system and not a specific library, the many basic and descriptive statistical queries that would be performed would take up too much space and time of the project.

Functions used in this process:

• pd.read\_csv

• df.info()

• df.describe()

• df.isnull().sum()

• df2.duplicated().sum()

Then a table was created for each csv file, forming our schema:

* channels: Contains information about the sales channels (marketplaces) where the retailers' goods and food are sold.
* Deliveries: Contains information about deliveries made by partner delivery people.
* drivers: Contains information about partner delivery people. They stay at the hubs and whenever an order is processed, they are the ones who make the deliveries to customers' homes.
* hubs: Contains information about the Delivery Center's hubs. Understand that the hubs are the order distribution centers and that's where the deliveries come from.
* orders: Contains information about sales processed through the Delivery Center platform.
* payments: Contains information about payments made to the Delivery Center.
* stores: Contains information about retailers. They use the Delivery Center platform to sell their items (goods and/or food) on marketplaces.
* Completing the schema, views, indexes, and a trigger were developed, which were added and will be mentioned later.

The data present in the csv file was inserted into these tables using pgAdmin 4 functions.

Thus, the first problem of the project arose because it was not possible to insert the data from the orders table into some columns (order\_moment) as TIMESTAMP due to their values being arranged in the MM/DD/YYYY format, the correct format being YYYY/MM/DD.

The solution was to insert the values as strings using VARCHAR and then change them to TIMESTAMP. This process was important for the project because the ‘order\_moment’ columns were used in several queries and functions.

After a general analysis of the dataset, the project was divided into 4 stages:

Analysis of Orders and Deliveries

Analysis of Payments

Analysis of Hubs

Analysis of Delivery People

Analysis of Orders and Deliveries

Simple and general queries were developed here, such as: calculating the total number of deliveries and orders. Also more complex functions that help calculate the time taken for deliveries, regardless of the category entered by the user.

The functions in this stage were created with the objective of minimizing redundancies in searches, making the project cleaner and more organized. Example: To calculate the number of orders per channel and the number of orders per store segment, the queries would be practically the same.

Analysis of Payments

This stage was named this way because the name of the table is ‘payments’ but could easily be replaced with the name ‘revenue’, because besides evaluating the most used payment methods, the real focus was to apply queries with the intention of calculating revenue, comparing it between periods and different categories.

Analysis of Hubs

The focus of this stage was to analyze the capacity of the hubs to serve customers, along with their efficiency.

Analysis of Delivery People

The goal was to analyze the number of deliveries and also the efficiency of each delivery person.

After the analyses, it was verified the possibility of optimizing the project beyond the queries and functions that were created. It was also decided to insert INDEXES, VIEWS, and a TRIGGER.

The decision to create the indexes was based on the most used queries and the help of the EXPLAIN function to observe where performance was being lost in each one. It is important to note that in large projects, the creation of indexes must take into account the memory consumed.

The views were created with the intention of being used as a tool for searching orders and deliveries, containing the most relevant general information for this purpose.

The trigger developed simulates the insertion of new data in a real system, specifically at the time of payment, where the customer, upon finalizing the payment by changing the status to ‘PAID’, the order status would also be changed to ‘COMPLETED’, preventing the order from being completed before the payment is made.