Scripts e processo de normalização do banco de dados

Durante o processo de normalização da base de dados do Olist, utilizei principalmente o Power Query para identificar e eliminar redundâncias, além de reorganizar os dados em estruturas mais limpas, consistentes e alinhadas aos princípios da modelagem relacional. Abaixo explico o que foi feito em cada parte:

Tabela olist_customers

A tabela original de clientes armazenava repetidamente informações de localização (cidade e estado) junto com o cliente. Isso fazia com que muitos dados fossem duplicados desnecessariamente.

O que foi feito:

- Mantive na tabela customers apenas os campos: customer_id, customer_unique_id e customer_zip_code_prefix.
- Criei uma nova tabela chamada customers_location, contendo os campos customer_zip_code_prefix, customer_city e customer_state.
- Dessa forma, cada CEP passou a aparecer uma única vez na tabela de localização, evitando a repetição de cidade e estado em cada registro de cliente.

Tabela olist_order_items

Originalmente, a tabela de itens do pedido armazenava múltiplas linhas para cada combinação de order_id e product_id, com os valores de price e freight_value sendo repetidos por linha.

Normalização feita:

- Criei uma nova tabela order_items contendo uma linha para cada par order_id e product_id.
- Agrupei os dados por pedido e produto e calculei:
    - qtd_compra: total de itens daquele produto no pedido.
    - total_price: soma dos preços.
    - Mantive também seller_id, shipping_limit_date e order_item_id.

Além disso, criei uma nova tabela separada chamada freight_value:

- Nela, armazenei apenas o primeiro valor de frete (freight_value) por pedido (order_id), já que o valor do frete é o mesmo para todos os itens de um pedido.
- Isso eliminou repetições e somas incorretas do frete, tornando os dados mais precisos e organizados.

Tabela order_payments

A tabela de pagamentos possuía o campo payment_type escrito como texto em cada linha, o que gerava redundância.

O que foi feito:

- Criei uma nova tabela payment_types, que armazena os nomes únicos de tipos de pagamento com um identificador payment_type_id.
- Na tabela order_payments, substituí o campo de texto pelo payment_type_id, referenciando a nova tabela.
- Isso deixou os dados mais organizados, evitando erros de digitação e facilitando análises por tipo de pagamento.

Tabela sellers

Na tabela de vendedores (sellers), os campos de localização (seller_city e seller_state) eram repetidos diversas vezes, sempre que o seller_zip_code_prefix se repetia.

Normalização aplicada:

- Mantive na tabela sellers apenas os campos seller_id e seller_zip_code_prefix.
- Criei uma nova tabela sellers_location, com os campos seller_zip_code_prefix, seller_city e seller_state, onde cada prefixo de CEP aparece apenas uma vez.

Essa separação eliminou a redundância e deixou os dados de localização dos vendedores mais bem organizados.

Resultado

Com essa normalização:

- Reduzi a redundância nos dados.
- Organizei melhor os relacionamentos entre as entidades.
- Facilitei análises futuras.
- Melhorei a performance e a consistência da base.

Toda a normalização foi feita utilizando o Power Query e scripts SQL para refletir a estrutura final no banco PostgreSQL.