Ficha técnica - Projeto 05

Estrutura de Dados (ETL) - Super Store

Ferramentas e tecnologias

- Google BigQuery
- SQL
- Google Colab
- Python
- ChatGPT e Gemini na validação de códigos
- Draw.io
- Loom
- Notion

Objetivo

Desenvolver um processo ETL (Extração, Transformação e Carga) eficiente e confiável para a base de dados da Super Store, com foco na criação de um data warehouse relacional otimizado. A proposta consiste em organizar os dados brutos e não estruturados em um modelo dimensional composto por tabelas fato e tabelas dimensão, permitindo consultas mais rápidas, redução de redundâncias e melhor desempenho no armazenamento.

O objetivo central é disponibilizar uma base analítica estruturada, capaz de fornecer informações relevantes para apoiar a tomada de decisão estratégica, reduzir a carga computacional sobre as fontes de dados originais e facilitar a criação de dashboards e relatórios gerenciais.

Equipe

Vanessa Santana do Amaral

Processamento dos dados

Foi criado um projeto no Google BigQuery com o nome `etlsuperstore05`. Em seguida, a base de dados original foi carregada para o ambiente, formando a

tabela bruta `superstorevendas.base_vendas`, composta por **51.290 registros**.

Essa base contém informações detalhadas sobre vendas, clientes, produtos, regiões e aspectos operacionais do processo de pedidos. Abaixo, a descrição das principais variáveis:

- **category**: Categoria principal do produto vendido.
- **sub_category**: Subcategoria do produto dentro da categoria principal.
- **city**: Cidade onde o pedido foi realizado.
- **state**: Estado correspondente à cidade do pedido.
- **country**: País de origem do pedido.
- **region**: Região geográfica do pedido.
- **market** e **market2**: Informações referentes à área de atuação de mercado.
- **customer_id**: Identificador único do cliente.
- **customer_name**: Nome do cliente.
- **segment**: Tipo de cliente (ex: consumidor, empresa, escritório doméstico).
- **product_id**: Identificador único do produto.
- **product_name**: Nome do produto vendido.
- **order_id**: Identificador único de cada pedido.
- **order_date**: Data em que o pedido foi realizado.
- **ship_date**: Data em que o pedido foi enviado.
- **ship_mode**: Modalidade de envio utilizada.
- **order_priority**: Prioridade atribuída ao pedido.
- **quantity**: Quantidade de itens vendidos.
- **sales**: Valor total da venda.
- **discount**: Desconto aplicado no pedido.
- **profit**: Lucro obtido com a venda.
- **shipping_cost**: Custo de envio do pedido.
- **weeknum**: Número da semana em que o pedido foi realizado.
- **year**: Ano da realização do pedido.

- **row_id**: Identificador único de cada linha no dataset.
- **unknown**: Coluna não especificada, de conteúdo irrelevante para o modelo.

Tratamento dos dados

Nesta fase inicial, foi adotada uma abordagem diagnóstica. Nenhuma modificação ou limpeza foi aplicada de imediato. O objetivo principal foi mapear a estrutura e qualidade dos dados, orientando decisões futuras durante a construção das tabelas.

Análises realizadas:

- **Valores nulos**: Nenhuma ocorrência identificada.
- **Valores duplicados**: Nenhuma duplicidade encontrada.
- **Dados discrepantes**:
  - **Categóricos**: Foi identificado que algumas colunas textuais necessitariam padronização futura, como remoção de espaços e conversão para letras minúsculas.
  - **Numéricos**: Todos os valores foram considerados coerentes com o contexto de negócio.
- **Exclusão de colunas irrelevantes** (planejada para a modelagem):
  - `row_id`: Campo sequencial sem valor analítico.
  - `unknown`: Coluna sem variação ou significado relevante (valor constante igual a 1).
- **Correção de tipos de dados**:
  - A conversão de `order_date` do tipo **TIMESTAMP** para **DATE** foi prevista, visto que a informação de horário estava zerada em todos os registros, sendo irrelevante para o modelo de análise temporal.

Webscraping

Foi realizado um experimento de web scraping utilizando Python no ambiente Google Colab, com o objetivo de explorar a coleta de dados externos relacionados a concorrentes e contexto de mercado.

Essa extração não foi integrada ao projeto principal, pois o foco da iniciativa era o desenvolvimento do pipeline ETL e a estruturação de um modelo dimensional. A atividade teve caráter exploratório e de aprendizado, contribuindo para o entendimento complementar de dados, mas sem impacto direto na base final.

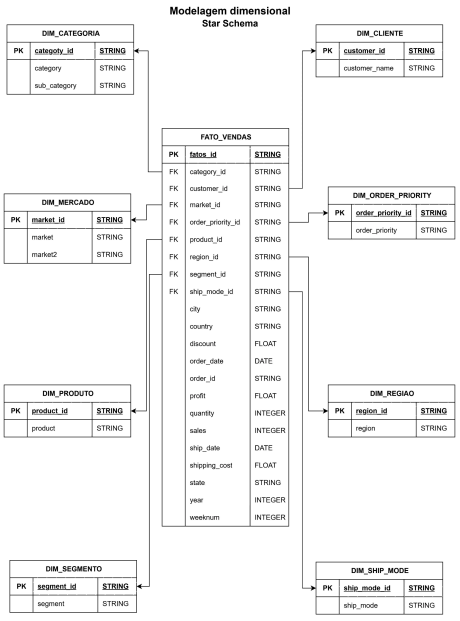
Modelagem dimensional

Antes da criação das tabelas no BigQuery, foi desenhado o modelo dimensional do projeto. Foi escolhido o Esquema em Estrela devido à sua simplicidade e alta performance em consultas analíticas. A estrutura evita redundâncias, mas mantém os dados acessíveis e bem organizados para futuras explorações.

O objetivo era garantir clareza na estrutura, padronização das entidades e melhor desempenho nas consultas. A modelagem foi feita utilizando a ferramenta Draw.io, onde foram definidos:

1 tabela fato_vendas: Contém as métricas quantitativas como `sales`, `profit`, `discount`, `quantity` e `shipping_cost`.

8 tabelas dimensão: dim_categoria, dim_cliente, dim_mercado, dim_order_priority, dim_produto, dim_regiao, dim_segmento, dim_ship_mode



Criação das tabelas dimensão

Após a definição da modelagem dimensional, foram criadas as tabelas de dimensão conforme o planejado.

Ao todo, foram desenvolvidas 8 tabelas dimensão, já citadas no tópico anterior.

Durante o processo de criação, foi utilizado o comando `ROW_NUMBER()` para controle de duplicidade nas junções (`JOINs`), garantindo a integridade dos dados. Além disso, foram aplicadas chaves artificiais (UUIDs) nas situações em que não havia uma chave natural única, assegurando a existência de chaves primárias (PK) em todas as tabelas dimensão.

Criação da tabela fato

Na criação da tabela fato, optou-se por gerar um identificador único (`fato_id`) aleatório, que serve como chave primária (PK) da tabela.

Foram selecionadas as variáveis quantitativas que representam o panorama geral da Superstore, como `quantity`, `profit`, `discount`, entre outras métricas financeiras e operacionais.

Além disso, foram incluídas as chaves estrangeiras (FKs) que referenciam as tabelas dimensão, estabelecendo as relações por meio de operações `LEFT JOIN` para garantir que todos os registros da base principal fossem mantidos, mesmo que não houvesse correspondência em alguma dimensão.

Validação das tabelas geradas

Após a criação das tabelas, foram realizadas consultas para garantir que as tabelas estavam corretamente integradas e que a estrutura final entregava informações consistentes e corretas.

- **Contagem de registros:**
  - `base_vendas_limpa`: 51.290 registros
  - `fato_vendas`: 51.290 registros (igual à base original, indicando que não houve duplicação)
- **Verificação de integridade:**
  - Nenhum valor `NULL` foi encontrado nas chaves estrangeiras, confirmando a integridade referencial entre fato e dimensões.
- **Análise e resolução de duplicidades:**
  - Inicialmente, a tabela fato apresentava 55.996 registros devido a múltiplos matches na junção com dimensões.

- O problema foi solucionado aplicando `ROW_NUMBER()` nas dimensões para garantir unicidade e filtragem dos registros, evitando multiplicidades e garantindo a consistência da tabela fato.

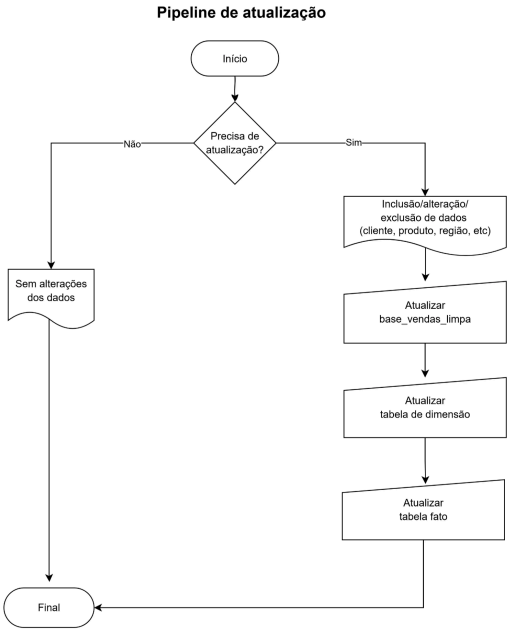
Pipeline de atualização

Nesta etapa, o foco foi mais descritivo e de aprendizado. Não foi implementado um pipeline automatizado em ferramentas específicas, mas sim desenhada a estrutura e o fluxo ideal para atualização dos dados.

O fluxo determinado para atualização é o seguinte:

1. **Atualização da base limpa (`superstore_base_limpa`)**: Esta etapa deve ser executada primeiro, pois as tabelas de dimensão dependem diretamente dos dados limpos e padronizados presentes nesta base.
2. **Atualização das tabelas dimensão**: Após a base limpa estar atualizada, as tabelas de dimensão são reconstruídas ou atualizadas com os dados provenientes da base limpa. Isso garante que todas as dimensões estejam atualizadas antes da criação da tabela fato.
3. **Atualização da tabela fato**: Por fim, a tabela fato é gerada a partir das junções das tabelas dimensão com a base limpa. Este passo deve ocorrer após a atualização completa das dimensões para garantir integridade e consistência nos dados relacionais.

Assim, a ordem lógica da atualização é fundamental para garantir que todas as dependências sejam respeitadas, evitando inconsistências ou erros no modelo dimensional.



Tratamento de Slowly Changing Dimensions (SCD)

Durante o desenvolvimento deste projeto, foi considerado a importância do tratamento de Slowly Changing Dimensions (SCD) para garantir que alterações nos dados dimensionais fossem gerenciadas adequadamente ao longo do tempo.

Foi optado por implementar o **SCD tipo 1**, que consiste em sobrescrever os dados antigos pelas novas informações, sem manter o histórico de alterações. Essa decisão foi tomada com base no contexto e nas necessidades atuais da Super Store, onde o foco principal é a análise do estado atual dos dados, e não o rastreamento histórico detalhado.

Essa abordagem simplifica a manutenção das tabelas de dimensão, evitando a complexidade adicional de armazenar múltiplas versões de registros. No entanto, recomendamos que futuros projetos avaliem a possibilidade de implementar SCD Tipo 2 ou Tipo 3 caso seja necessário acompanhar mudanças históricas para análises mais profundas.

Assim, o tratamento de SCD está alinhado com a simplicidade, performance e objetivo analítico deste modelo dimensional.

Recomendações

- Automatizar o pipeline para agendamento e monitoramento das atualizações.
- Criar testes automatizados para validar a integridade e consistência dos dados após cada execução.
- Avaliar uso de Slowly Changing Dimensions (SCD) para controlar histórico de mudanças importantes nas dimensões.
- Manter documentação atualizada dos processos e regras aplicadas.
- Otimizar consultas e estrutura das tabelas para garantir boa performance com o crescimento dos dados.
- Estabelecer governança de dados para segurança, privacidade e compliance.