

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
Faculdade de Tecnologia de Jundiaí – “Deputado Ary Fossen”
Curso Superior de Tecnologia em Ciência de Dados

Ariel Ladislau Reises
João Paulo Martins
Matheus Castro Alexandre
Thiago Macedo Vaz
Sofia Pena

RELATO TÉCNICO
ARQUITETURA DATA WAREHOUSE COM DATASET OLIST E DUCKDB

Jundiaí
2025

1 INTRODUÇÃO

Este relato aborda o desenvolvimento completo de um Data Warehouse (DW) utilizando dados reais de comércio eletrônico. O projeto segue a arquitetura clássica utilizada em ambientes corporativos de dados, estruturando claramente cada etapa do fluxo de processamento: *Staging Layer*, *OLTP*, *Data Warehouse*, validação, análises e otimização. A organização dessas camadas permite compreender como os dados evoluem desde sua forma bruta até modelos analíticos prontos para responder perguntas de negócio.

A compreensão dessa estrutura é fundamental para qualquer profissional de dados, pois o DW representa o núcleo de inteligência de uma organização. Os dados operacionais do sistema OLTP são altamente normalizados e voltados para transações, sendo inadequados para análises diretas. Por isso, é necessária a ingestão inicial no *staging*, onde os arquivos são padronizados e preparados; em seguida, os dados passam para o modelo operacional normalizado (OLTP), que simula as bases transacionais reais utilizadas por aplicações de e-commerce. Finalmente, eles são transformados para o modelo dimensional do DW, responsável por acelerar consultas, facilitar análises e garantir consistência histórica.

O processo apresentado neste relato evidencia a importância de entender todo o fluxo, desde a fonte dos dados até a camada analítica. Em projetos reais, erros em um estágio podem afetar os resultados estratégicos posteriores. Por isso, foram criadas rotinas de validação com métricas de controle, visualizações exploratórias e demonstração da ideia por trás de possíveis otimizações. Com isso, demonstra-se não só a construção de um DW funcional, mas também a abordagem profissional exigida para garantir confiabilidade, reprodutibilidade e eficiência.

Além disso, o trabalho cobre desde as transformações técnicas (SCD2, joins, carga de dimensões e fatos) até a criação de visualizações e tabelas agregadas voltadas ao negócio, reforçando a conexão entre engenharia de dados e análise de dados. O resultado é um pipeline ponta a ponta que reflete boas práticas utilizadas em ambientes reais.

2 DESENVOLVIMENTO INICIAL E STAGING

O primeiro passo para a construção do Data Warehouse foi a ingestão do *dataset* público do e-commerce brasileiro Olist, disponibilizado no *Kaggle*. A extração foi realizada diretamente no ambiente do *notebook*, garantindo que sempre fosse utilizada a versão mais recente dos arquivos. Após o *download*, iniciou-se a preparação da *Staging Layer*.

Nesta camada, não realizamos transformações ou limpezas; o objetivo é simplesmente disponibilizar os dados brutos de forma organizada, padronizada e acessível. Para isso, cada arquivo CSV original foi exposto como uma *view* no *DuckDB*, utilizando a função *read_csv_auto*. Essa abordagem traz diversas vantagens operacionais: garante reprodutibilidade, facilita auditoria, separa claramente dados brutos das próximas fases do *pipeline* e simplifica a modelagem *OLTP* que será construída posteriormente.

A criação da camada de *staging* começou com a definição das *views* responsáveis por espelhar cada tabela bruta do dataset (*orders*, *order_items*, *payments*, *reviews*, *products*, *customers*, *sellers* e *geolocation*). Todo esse processo foi consolidado no arquivo *00_staging.sql*, permitindo executar a etapa de ingestão de forma automatizada e reaplicável.

Por fim, o banco *DuckDB* foi inicializado e o arquivo SQL de staging foi executado. Isso resultou na criação imediata das *views* de referência, que passam a servir como a base para a etapa seguinte, a construção do modelo operacional (*OLTP*), responsável por normalizar e estruturar os dados antes que eles alimentem o *Data Warehouse*.

3 ETAPA DE OLTP

Após estruturar o ambiente de *staging* e disponibilizar os arquivos brutos em *views* padronizadas, avançamos para a etapa de construção do modelo *OLTP*. Essa camada tem como objetivo organizar os dados de forma normalizada, seguindo princípios clássicos de bancos transacionais. Diferentemente do *staging*, que apenas expõe os dados originais, o *OLTP* é a primeira camada onde de fato ocorre modelagem estruturada.

A transformação das *views* brutas em tabelas normalizadas permite criar uma representação mais fiel a um sistema operacional real, reduzindo redundâncias, corrigindo tipos de dados, garantindo integridade referencial e separando entidades de maneira clara. Isso é especialmente importante em pipelines de Data Warehouse, pois o OLTP funciona como uma camada intermediária entre os dados brutos e a modelagem dimensional. Manter essa separação reforça boas práticas de arquitetura, facilita auditoria e ajuda a evitar a propagação de inconsistências para as camadas analíticas.

Durante a construção do OLTP, criamos tabelas como clientes, vendedores, produtos, pedidos, itens de pedido, pagamentos e reviews. Cada tabela foi construída com base direta nas *views* de *staging*, aplicando *cast* de tipos adequados, extração de colunas relevantes e remoção de duplicidades. Além disso, asseguramos que cada entidade tivesse uma chave primária consistente e um formato alinhado com o padrão usado posteriormente no *Data Warehouse*.

Todo o processo foi consolidado no arquivo *01_oltp.sql*, que garante execução idempotente (permitindo *reruns* sem riscos) e cria automaticamente todas as tabelas normalizadas a partir das *views* de *staging*. Essa camada se torna, portanto, a base transacional organizada a partir da qual a modelagem dimensional será construída na etapa seguinte.

3 ETAPA DO DATA WAREHOUSE

Com a camada OLTP totalmente estruturada e normalizada, avançamos para a etapa central do projeto: a construção do *Data Warehouse*. Diferentemente do modelo transacional, o DW é projetado para análise, tendência, agregação e suporte a decisões estratégicas — exigindo um formato de dados otimizado e orientado a consultas.

Nesta fase implementamos o modelo estrela (*Star Schema*), padrão consolidado em ambientes de *Business Intelligence*, composto por tabelas dimensão e uma tabela fato. Essa estrutura torna operações analíticas significativamente mais eficientes, pois organiza os dados de forma intuitiva e altamente performática para agregações, *drill-downs* e cruzamentos multidimensionais.

As dimensões funcionam como "eixos descritivos" das análises, armazenando atributos textuais e categóricos que contextualizam os eventos da tabela fato.

Neste projeto, construímos dimensões para: clientes, produtos, vendedores, tipos de pagamento e datas. Essas dimensões utilizam chaves substitutas (*surrogate keys*), geradas por meio de sequências, garantindo estabilidade das relações mesmo em cenários onde os dados originais mudem ou sofram retrabalho. Para clientes, produtos e vendedores, implementamos o mecanismo de *Slowly Changing Dimension Type 2 (SCD2)*, assegurando histórico completo de alterações ao longo do tempo — essencial para análises consistentes e auditáveis.

Já a tabela fato concentra os eventos transacionais de venda no seu nível mais granular: um registro por item de pedido (*order_id* + *order_item_id*). Nela armazenamos: preços, valores de frete, valores pagos, reviews, quantidade e as chaves das dimensões.

Essa granularidade permite análises detalhadas e flexíveis, desde métricas operacionais até indicadores estratégicos. A fato também preserva as chaves naturais de origem para rastreabilidade (*data lineage*).

Toda a estrutura do DW é definida no arquivo `02_dw_model.sql`, que cria as sequências, dimensões e a tabela fato de forma idempotente, garantindo repetibilidade e consistência do pipeline.

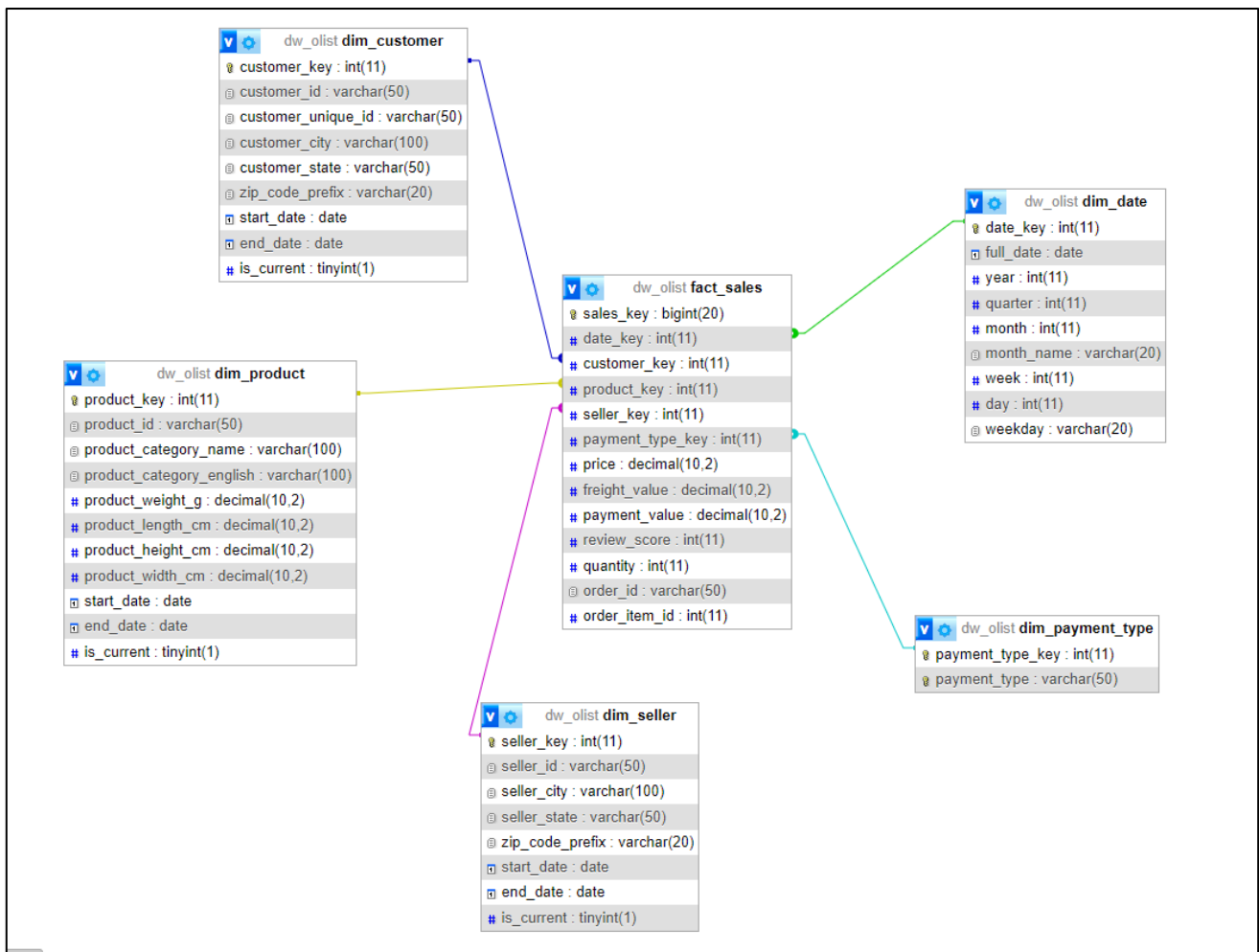


Figura 1 - Star Schema do Data Warehouse

4. ETAPA DE CARGA

Após a modelagem das tabelas de *Staging*, *OLTP* e *DW*, esta etapa executa o processo de ETL (*Extract, Transform, Load*) que efetivamente popula todas as estruturas construídas. O objetivo do ETL é transformar os dados normalizados do OLTP em dados analíticos consolidados no *Data Warehouse*, aplicando regras de negócio, controle de histórico e eliminação de duplicidades.

Diferente das etapas anteriores, onde apenas definimos estrutura, aqui iniciamos o movimento real dos dados entre camadas. O processo é composto por extração das tabelas OLTP, aplicação de regras de transformação e carregamento das dimensões e da tabela fato de forma consistente e idempotente.

4.1 Geração da Tabela de Datas (dim_date)

A tabela de datas é construída de forma programática, cobrindo todo o intervalo existente no dataset original. Como DuckDB não possui funções como *generate_series* ou *to_char* em SQL padrão PostgreSQL, utilizamos uma combinação de *range()*, *date_diff()* e *strftime()* para gerar todas as datas entre o primeiro e o último pedido.

A *date dimension* é totalmente reconstruída a cada execução, garantindo idempotência e simplicidade operacional.

4.2 Carregamento da Dimensão de Tipo de Pagamento

A dimensão *dim_payment_type* é simples, estática e não requer controle de histórico. Aqui carregamos todos os tipos de pagamento distintos presentes na OLTP, como *credit_card*, *boleto*, *debit_card*, entre outros.

4.3 Implementação de SCD Tipo 2 (Clientes, Produtos e Vendedores)

As dimensões *dim_customer*, *dim_product* e *dim_seller* utilizam o mecanismo *Slowly Changing Dimension Type 2*, permitindo armazenar múltiplas versões de um mesmo registro ao longo do tempo. Esse controle é essencial para auditoria, análises fiéis ao momento da transação e preservação de histórico.

O ETL implementa SCD2 em três etapas:

1. Detecção de registros novos:
Registros presentes no OLTP e ausentes na dimensão são inseridos como novas entradas com *start_date* = *CURRENT_DATE*.
2. Detecção de alterações:
Comparações entre atributos do OLTP e dimensões correntes identificam mudanças em cidade, estado, categoria, peso, etc. Quando detectadas, a versão antiga é marcada com:
 - *end_date* = *CURRENT_DATE* - 1
 - *is_current* = FALSE
3. Criação da nova versão do registro:
A nova versão é inserida com *start_date* = *CURRENT_DATE* e *is_current* = TRUE.

Essa abordagem preserva o histórico completo e garante que a tabela fato sempre se relacione com a versão vigente no momento da venda.

4.4 Carregamento da Tabela Fato

A tabela fato é o ponto final do fluxo de informação no DW. Cada registro representa um item de pedido, com seus valores financeiros, relacionamento com dimensões e informações analíticas consolidadas.

Durante o carregamento:

- realizamos JOINS entre OLTP e dimensões com filtragem `is_current = TRUE`
- associamos corretamente clientes, produtos, vendedores e meios de pagamento
- trata-se a questão de reviews duplicados usando uma CTE com `ROW_NUMBER()`, garantindo uma única review por pedido

5 VALIDAÇÃO DO DW

Com o *Data Warehouse* totalmente carregado, é fundamental garantir que os dados estejam corretos, íntegros e coerentes com as fontes originais. A etapa de validação é crítica em qualquer pipeline analítico, pois assegura que as análises, gráficos e KPIs produzidos posteriormente sejam confiáveis.

Aqui, implementamos uma rotina completa de validação automatizada, verificando pontos-chave de consistência, integridade e conformidade estrutural entre *OLTP* e *DW*.

O processo de validação inicia verificando a contagem de registros nas tabelas de OLTP e DW. Como regra fundamental, a tabela fato não pode possuir mais registros do que o número total de itens presentes no modelo transacional; uma diferença significativa indicaria duplicidades ou problemas no processo de carga. Em seguida, avaliamos a presença de chaves estrangeiras essenciais na tabela fato, como referências a clientes, produtos, vendedores e datas. A ausência dessas chaves comprometeria o uso do DW em análises e relatórios, além de violar a integridade do modelo estrela.

Outra verificação essencial consiste em garantir que cada fato possua uma correspondência real nas dimensões. Essa checagem identifica possíveis registros órfãos, situações em que uma linha da fato não encontra um membro correspondente em uma dimensão específica, o que tornaria o DW inconsistente e inadequado para consultas analíticas. Avaliamos também o intervalo de datas

da tabela `dim_date`, comparando-o com o período real coberto pelas transações no OLTP. Essa verificação garante que o calendário do DW contempla todo o histórico do dataset, evitando lacunas em análises temporais.

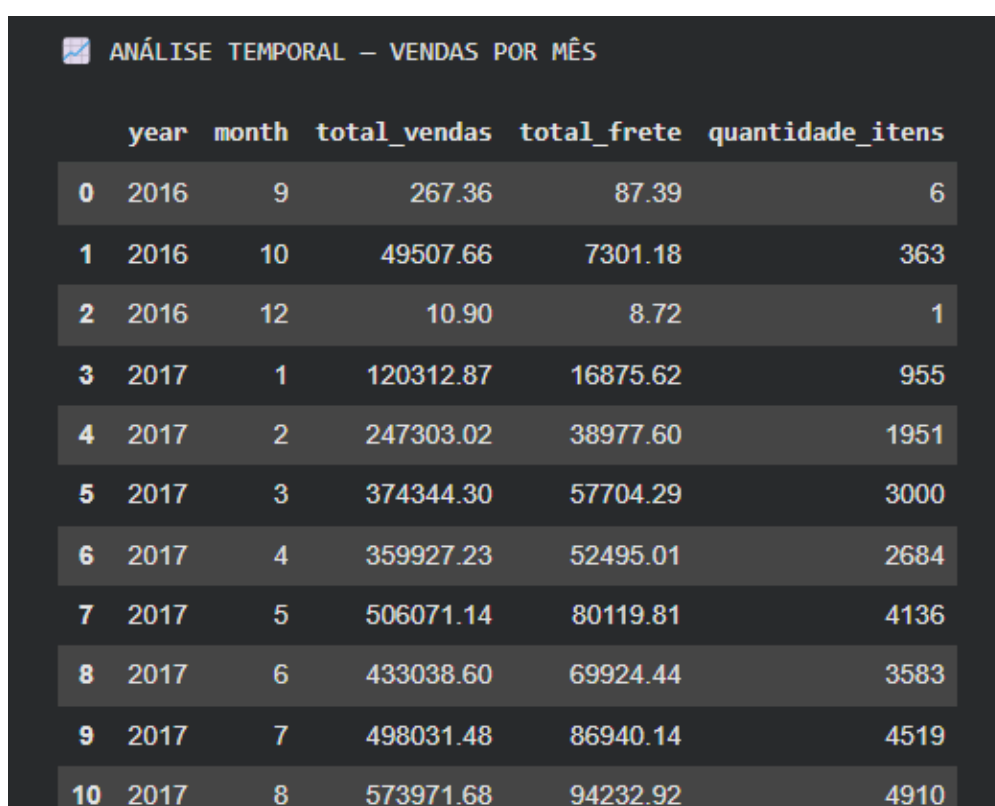
Por fim, realizamos uma verificação completa das dimensões com controle de histórico (SCD Tipo 2). Essa validação verifica se existe mais de uma versão marcada como vigente (`is_current = TRUE`) para o mesmo identificador natural, o que indicaria falhas no processo de versionamento. A correta implementação do SCD2 é crucial para assegurar análises temporais consistentes e alinhadas com o contexto de cada evento registrado na tabela fato.

6 CONSULTAS ANALÍTICAS

Com o Data Warehouse devidamente estruturado e validado, avançamos para a etapa de Consultas Analíticas, momento em que o modelo dimensional demonstra plenamente seu propósito: transformar dados transacionais brutos em insights acionáveis. Esta fase marca a transição do pipeline técnico para a exploração analítica, permitindo responder perguntas típicas de um ambiente corporativo de *Business Intelligence*.

As consultas analíticas utilizam principalmente a tabela fato (*fact_sales*) combinada com suas dimensões, permitindo examinar vendas, comportamento de clientes, desempenho de produtos e padrões operacionais sob múltiplas perspectivas.


Entre as análises realizadas estão avaliações temporais que demonstram a evolução das vendas ao longo dos meses, revelando sazonalidade e tendências de demanda; rankings que identificam os produtos mais vendidos ou mais rentáveis, fundamentais para gestão de portfólio, marketing e estoque; análises multidimensionais que combinam diferentes eixos (como categoria × estado), permitindo observar preferências regionais e perfis de consumo em nível granular; *cohort analysis* para investigar retenção e fidelização de clientes ao longo do tempo; além do cálculo de KPIs essenciais, como ticket médio por estado, receita agregada e volume total de itens vendidos. Alguns exemplos:



ANÁLISE TEMPORAL – VENDAS POR MÊS

	year	month	total_vendas	total_frete	quantidade_itens
0	2016	9	267.36	87.39	6
1	2016	10	49507.66	7301.18	363
2	2016	12	10.90	8.72	1
3	2017	1	120312.87	16875.62	955
4	2017	2	247303.02	38977.60	1951
5	2017	3	374344.30	57704.29	3000
6	2017	4	359927.23	52495.01	2684
7	2017	5	506071.14	80119.81	4136
8	2017	6	433038.60	69924.44	3583
9	2017	7	498031.48	86940.14	4519
10	2017	8	573971.68	94232.92	4910

Figura 2 - Análise Temporal de vendas por mês



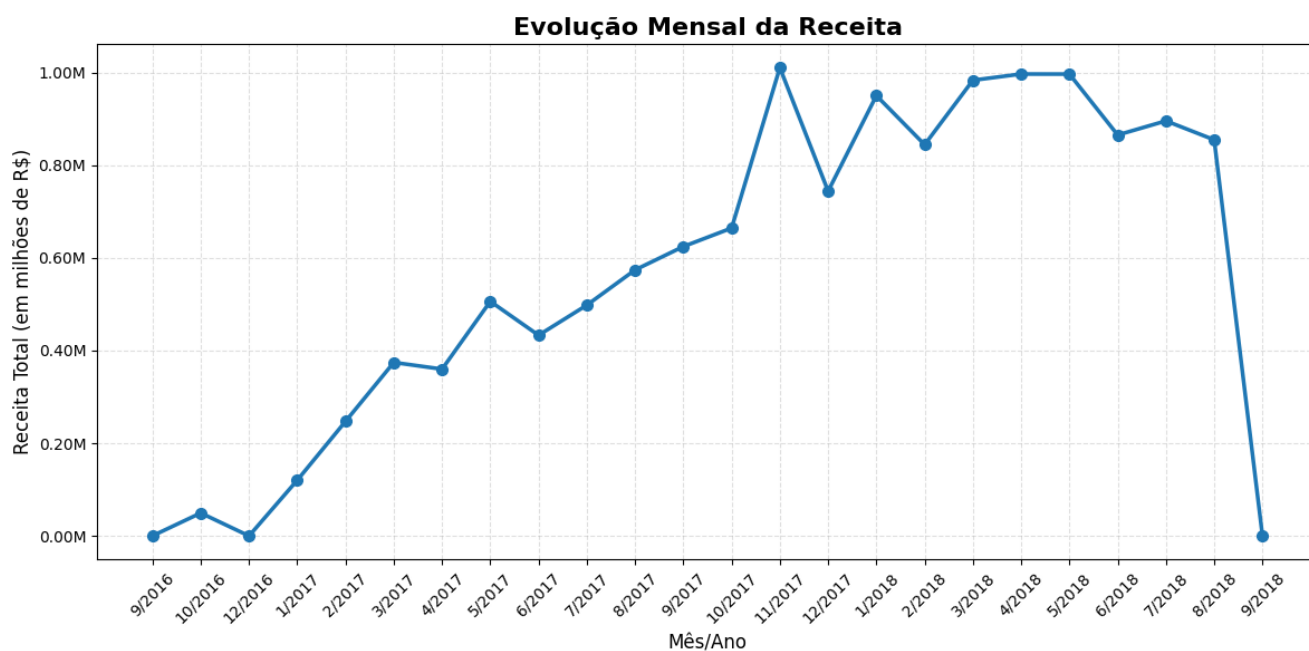
	categoria	estado	qtd_vendas	receita	ticket_medio
0	bed_bath_table	SP	5235	478284.52	91.362850
1	health_beauty	SP	4204	462305.22	109.967940
2	watches_gifts	SP	2281	435009.92	190.710180
3	sports_leisure	SP	3667	386357.01	105.360515
4	computers_accessories	SP	3170	350747.88	110.646019
...
1377	drinks	AL	1	15.49	15.490000
1378	home_comfort_2	ES	1	12.90	12.900000
1379	home_comfort_2	BA	1	12.90	12.900000
1380	home_comfort_2	PR	1	12.90	12.900000
1381	costruction_tools_tools	MS	1	6.80	6.800000

Figura 3 - Agregação Multidimensional de categorias mais vendidas por estado

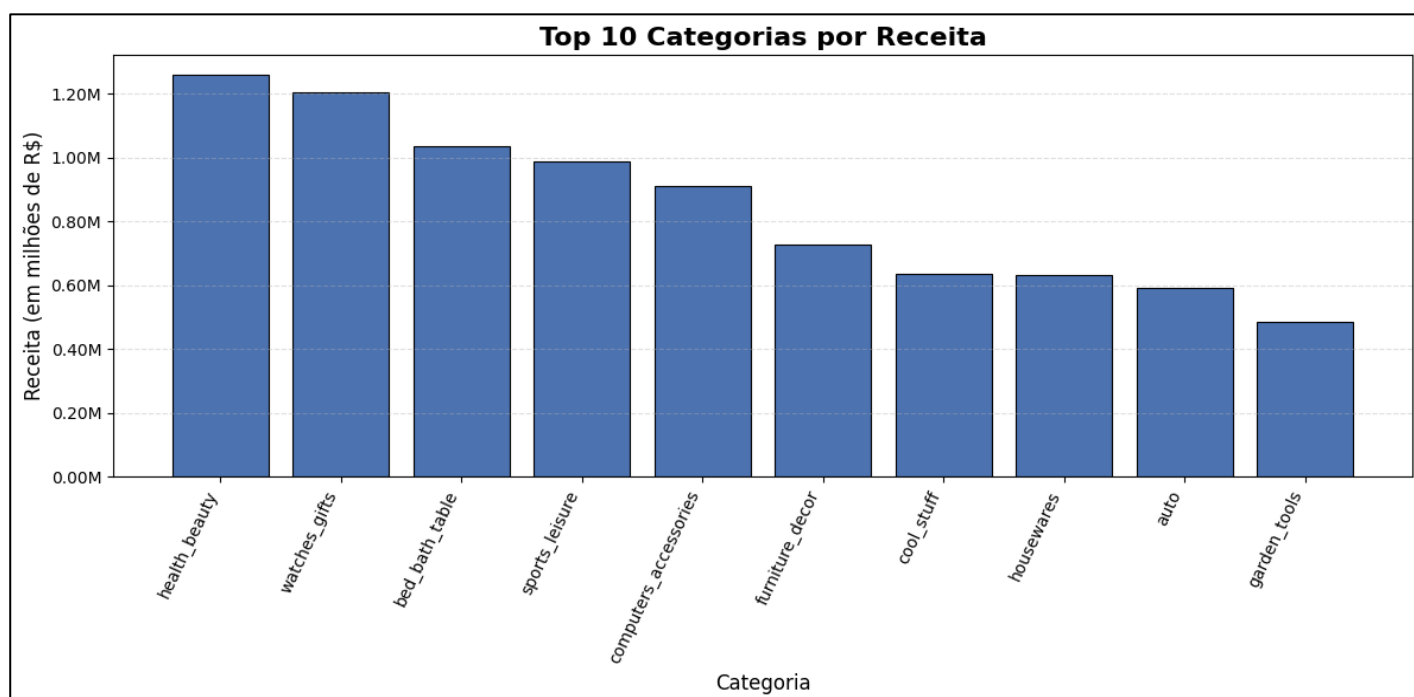
7 VISUALIZAÇÃO

Após executar as consultas analíticas no Data Warehouse, avançamos para a etapa de visualização, onde transformamos os resultados obtidos antes apresentados em tabelas e métricas numéricas em gráficos e representações visuais mais intuitivas. Essa fase é essencial para tornar a análise mais acessível e revelar comportamentos que muitas vezes não são facilmente perceptíveis apenas por meio de consultas SQL.

Por meio de gráficos temporais, rankings visuais, distribuições, comparações entre categorias e mapas de calor, é possível identificar tendências, padrões de consumo, sazonalidades e possíveis anomalias de forma muito mais clara e direta. A visualização cumpre o papel de traduzir o modelo dimensional em insights de negócio, permitindo que decisões estratégicas sejam tomadas com base em evidências visuais, enriquecendo substancialmente a interpretação dos dados e a comunicação dos resultados.



A série temporal revela o comportamento das vendas ao longo dos meses, permitindo identificar tendências, sazonalidade e picos como o aumento típico na Black Friday. É útil para planejamento de estoque e marketing.



O gráfico mostra quais categorias geram mais receita, permitindo entender onde o negócio tem maior força e quais segmentos poderiam receber mais investimento em marketing ou estoque.

8 OTIMIZAÇÃO DE PERFORMANCE

Essa fase tem como objetivo simular o comportamento de um ambiente corporativo de Business Intelligence, no qual grandes volumes de dados demandam respostas rápidas para alimentar dashboards, relatórios e análises operacionais. Assim, buscamos aplicar técnicas que reduzam o custo computacional das consultas, garantindo maior eficiência e escalabilidade do pipeline analítico.

O primeiro passo dessa etapa consistiu na criação de uma tabela agregada mensal (*agg_monthly_sales*), responsável por armazenar métricas já consolidadas por ano e mês. Essa estrutura elimina a necessidade de processar novamente toda a tabela fato sempre que uma análise temporal é executada, evitando repetições de somas e contagens. Com isso, reduzimos drasticamente o trabalho de leitura, agregação e junção de dados, permitindo que consultas recorrentes se tornem praticamente instantâneas.

Em seguida, realizamos uma comparação direta entre duas versões da mesma consulta: a execução original sobre a tabela fato *fact_sales*, que exige *joins* e agregações em tempo real, e a execução otimizada, que lê diretamente a tabela pré-agregada. Ambas as variações foram materializadas em tabelas auxiliares (*results_original* e *results_optimized*), para permitir análise comparativa e validação de consistência entre os resultados, etapa importante para assegurar que a otimização não compromete a integridade das métricas.

Para aprofundar a avaliação do desempenho, utilizamos o comando `EXPLAIN ANALYZE` nas duas versões da consulta. Essa ferramenta nos permitiu observar o tempo total de execução, a quantidade de linhas processadas e os operadores utilizados pelo mecanismo de execução do *DuckDB*. A consulta original apresentou um tempo aproximado de 0.0119s, envolvendo *hash join* entre fato e dimensão, leitura de mais de 112 mil linhas, agrupamento em tempo de execução e uma etapa de ordenação final. Já a versão otimizada, ao operar sobre apenas 24 linhas pré-agrupadas, eliminou completamente *joins* e agregações, resultando em um tempo de apenas 0.0014s.

Mesmo considerando que o dataset utilizado não é massivo, o ganho de performance foi superior a oito vezes, demonstrando a relevância da utilização de estruturas agregadas em *Data Warehouses*. Em ambientes reais, com tabelas fato contendo milhões de registros, esse tipo de abordagem tende a potencializar a eficiência dos dashboards, reduzir o consumo de infraestrutura, diminuir a carga sobre o sistema e assegurar respostas mais rápidas a usuários finais e ferramentas analíticas.