Atividade 2 - Arquitetura Medalhão (Bronze e Silver Layer) 🚀

Você foi designado para estruturar os dados de um grande e-commerce. Seu trabalho consistirá em extrair os dados das fontes e carregá-los na infraestrutura **Data Lakehouse** do Databricks. Para isso, você implementará todo o processo de ETL para que os dados brutos fiquem prontos para gerar análises que possibilitam que as melhores decisões estratégicas sejam tomadas pela empresa.

Objetivo

Construir uma dois notebooks reprodutíveis que:

- Realize a ingestão dos arquivos fontes para a camada Bronze.
- Realize a limpeza, padronização e transformação para a camada Silver.

Passo a Passo para Resolução da Atividade

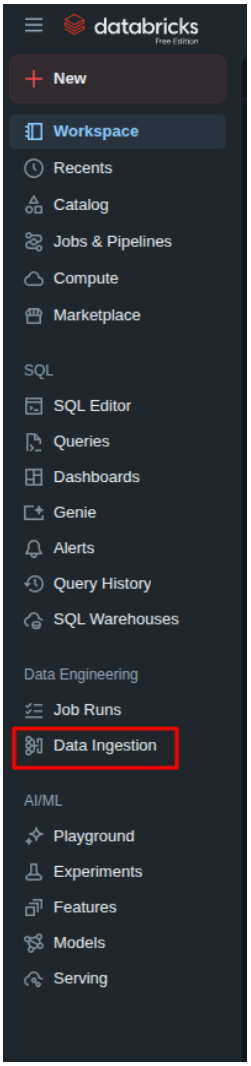
Fonte

- Dataset principal: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

Passo a passo

1) Preparação e upload dos dados

1. Baixe o ZIP do dataset Olist do Kaggle (link acima) e extraia os 9 arquivos CSV.
2. Carregue os arquivos **como Volume** no **Data Lakehouse** do Databricks



2) Notebook — camada Bronze

Crie um notebook no Databricks responsável por:

1. Criar dois databases: `bronze` e `silver`
2. Criar tabelas na camada bronze a partir de cada CSV com os nomes abaixo:
    a. Em cada uma dessas tabelas, deve haver uma coluna chamada: `ingestion_timestamp` , que deverá constar o timestamp do momento que o dado foi inserido na camada bronze

Arquivo	Nome da tabela (bronze)
olist_customers_dataset.csv	bronze.ft_consumidores
olist_geolocation_dataset.csv	bronze.ft_geolocalizacao
olist_order_items_dataset.csv	bronze.ft_itens_pedidos
olist_order_payments_dataset.csv	bronze.ft_pagamentos_pedidos
olist_order_reviews_dataset.csv	bronze.ft_avaliacoes_pedidos
olist_orders_dataset.csv	bronze.ft_pedidos
olist_products_dataset.csv	bronze.ft_produtos
olist_sellers_dataset.csv	bronze.ft_vendedores
product_category_name_translation.csv	bronze.dm_categoria_produtos_traducao

3. Extrair a cotação do dólar via API do Banco Central e salvar na tabela `bronze.dm_cotacao_dolar`

- Endpoint (modelo):

```
https://olinda.bcb.gov.br/olinda/servico/PTAX/versao/v1/odata/CotacaoDolarPeriodo(
  dataInicial=@dataInicial,dataFinalCotacao=@dataFinalCotacao
)?@dataInicial='{data_inicio_formatada}'&@dataFinalCotacao='{data_fim_formatada}'&$select=dataHoraCotacao,cotacaoCompra&$format=json
```

- Formato esperado de data: `MM-DD-AAAA`.
- `data_inicio_formatada` e `data_fim_formatada` devem ser parâmetros do notebook (variáveis que podem ser alteradas facilmente)

3) Notebook — camada Silver

Em um segundo notebook, implemente as transformações de limpeza, padronização ou enriquecimento dos dados. Cada transformação aplicada às tabelas da camada `bronze` deve gerar uma nova tabela correspondente na camada `silver`, mantendo o mesmo nome.

OBS: Nenhuma alteração será realizada em tabelas da camada bronze.

Requisitos principais:

- Padronização de nomes:
  - Todos os nomes de colunas devem estar em **português** (ex.: `customer_id` → `id_consumidor`)
  - Use convenção `snake_case` e prefira nomes descritivos.
  - Não será necessário levar todas as tabelas para a camada silver, e em algumas tabelas, não será necessário todas as suas colunas. Segue abaixo os requisitos, algumas transformações necessárias e os nomes cedidos pela área de negócio

  1. ft_consumidores

  - A coluna id_consumidor não deve conter valores duplicados, é necessário realizar essa verificação antes de salvar a tabela na camada Silver (Não utilize a customer_unique_id)
  - Nomes de Estado e Cidade devem estar em Upper Case (Em letras maiúsculas)

Coluna na Bronze	Coluna na Silver
customer_id	id_consumidor
customer_zip_code_prefix	prefixo_cep
customer_city	cidade
customer_state	estado

  2. ft_pedidos

  - Será necessário converter a coluna `order_status` do dataset da camada Bronze, que originalmente estavam em inglês, para seus equivalentes em português, de forma a padronizar e facilitar a interpretação dos dados na camada Silver.

    O mapeamento realizado foi o seguinte (**de/para**):

    - delivered → entregue
    - invoiced → faturado
    - shipped → enviado
    - processing → em processamento
    - unavailable → indisponível
    - canceled → cancelado
    - created → criado
    - approved → aprovado

- Será necessário criar novas colunas com informações derivadas, a fim de analisar as análises das áreas de negócio

1. **tempo_entrega_dias** → diferença em dias entre a data de entrega (pedido_entregue_timestamp) e a data de compra (pedido_compra_timestamp);
2. **tempo_entrega_estimado_dias** → diferença em dias entre a data estimada de entrega (pedido_estimativa_entrega_timestamp) e a data de compra (pedido_compra_timestamp);
3. **diferenca_entrega_dias** → diferença entre o tempo real e o tempo estimado de entrega;
4. **entrega_no_prazo** → indicador textual que deve conter:
   - "Sim" → quando a entrega ocorreu **no prazo** (diferença ≤ 0);
   - "Não" → quando ocorreu **fora do prazo**;
   - "Não Entregue" → quando o pedido **ainda não foi entregue**.

Coluna na Bronze	Coluna na Silver
order_id	id_pedido
customer_id	id_consumidor
order_status	status
order_purchase_timestamp	pedido_compra_timestamp
order_approved_at	pedido_aprovado_timestamp
order_delivered_carrier_date	pedido_carregado_timestamp
order_delivered_customer_date	pedido_entregue_timestamp
order_estimated_delivery_date	pedido_estimativa_entrega_timestamp
	tempo_entrega_dias
	tempo_entrega_estimado_dias
	diferenca_entrega_dias
	entrega_no_prazo

3. ft_itens_pedidos

Coluna na Bronze	Coluna na Silver
order_id	id_pedido
order_item_id	id_item
product_id	id_produto
seller_id	id_vendedor
price	preco_BRL
freight_value	preco_frete

4. ft_pagamentos

- Será necessário converter a coluna `order_status` do dataset da camada Bronze, que originalmente estavam em inglês, para seus equivalentes em português, de forma a padronizar e facilitar a interpretação dos dados na camada Silver.

O mapeamento aplicado (**de/para**) foi o seguinte:

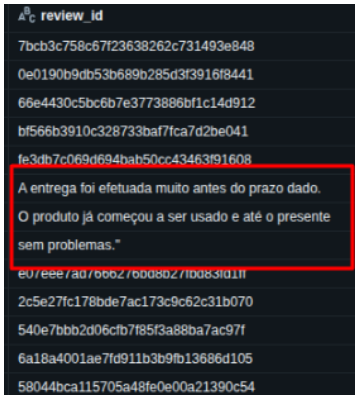
- credit_card → Cartão de Crédito
- boleto → Boleto
- voucher → Voucher
- debit_card → Cartão de Débito
- *(demais valores)* → Outro

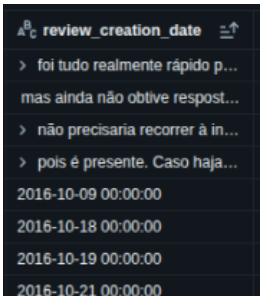
Coluna na Bronze	Coluna na Silver
order_id	id_pedido
payment_sequential	codigo_pagamento

Coluna na Bronze	Coluna na Silver
payment_type	forma_pagamento
payment_installments	parcelas
payment_value	valor_pagamento

5. ft_avaliacoes_pedidos

- Remover registros em `ft_avaliacoes_pedidos` que tenham `id_pedido` inválido ou datas incorretas (ex.: data nula, formato inconsistente, data futura fora do escopo).
- Documente as regras exatas de validação (o que é considerado "ID incorreto" e "data preenchida errada") no notebook e registre o número de linhas removidas.





Coluna na Bronze	Coluna na Silver
review_id	id_avaliacao
order_id	id_pedido
review_score	avaliacao
review_comment_title	titulo_comentario
review_comment_message	comentario
review_creation_date	data_comentario
review_answer_timestamp	data_resposta

6. ft_produtos

Coluna na Bronze	Coluna na Silver
product_id	id_produto
product_category_name	categoria_produto
product_weight_g	peso_produto_gramas
product_length_cm	comprimento_centimetros
product_height_cm	altura_centimetros
product_width_cm	largura_centimetros

7. ft_vendedores

- Nomes de estado e cidade devem estar em Upper Case (Em letras maiúsculas)

Coluna na Bronze	Coluna na Silver
seller_id	id_vendedor
seller_zip_code_prefix	prefixo_cep
seller_city	cidade
seller_state	estado

8. dm_categoria_produtos_traducao

Coluna na Bronze	Coluna na Silver
product_category_name	nome_produto_pt
product_category_name_english	nome_produto_en

6. dm_cotacao_dolar
   - A API não fornece cotação para finais de semana. Substitua faltas por **cotação de fechamento da sexta-feira anterior**.
   - Sugestão técnica: calcular a última cotação disponível usando *window functions* do spark.

Coluna na Bronze	Coluna na Silver
cotacaoCompra	cotacao_dolar
dataHoraCotacao	data

- Tipagem:
  - Converta tipos de dados para o tipo correto (strings que representam inteiros → `INT`, datas → `TIMESTAMP` / `DATE`, valores monetários → `DECIMAL(12,2)` ou `FLOAT` conforme necessidade).
- Validações:
  - Após o carregamento das tabelas da camada Silver (`ft_pedidos`, `ft_consumidores` e `ft_itens_pedidos`), realize uma **verificação de integridade referencial** entre os dados, garantindo que:
    1. **Todos os pedidos possuam um consumidor válido** (ou seja, não existam pedidos órfãos sem correspondência na tabela de consumidores).
    2. **Todos os itens de pedidos estejam associados a um pedido existente** (não existam itens órfãos sem pedido correspondente).
    - Ao final de cada uma das verificações, indique a quantidade de Pedidos e Itens órfãos. Caso eles existam, retire esses registros das tabelas
    - Dica: Utilize *joins* do tipo `left_anti` para identificar registros órfãos
- Por fim, será necessário a criação da tabela `silver.ft_pedido_total`
  - Junte as fontes necessárias: `bronze.ft_pedidos` com `bronze.ft_consumidores`, `bronze.ft_pagamentos_pedidos` e `bronze.dm_cotacao_dolar`.
- A tabela final deve conter as colunas:
  - `id_pedido`
  - `id_consumidor`
  - `status`
  - `valor_total_pago_brl` (soma dos pagamentos em BRL)
  - `valor_total_pago_usd` (soma dos pagamentos convertidos para USD usando cotação da data do pedido)
  - `data_pedido` (data do pedido)

  Segue um exemplo de dados da tabela final

data	id_pedido	id_consumidor	status	valor_total_pa
2017-10-02	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	38.71

Formato de Entrega

- Entregar **dois notebooks** em formato `.ipynb`: um notebook para a **camada bronze** e outro para a **camada silver**.
- Publicar ambos em um repositório **GitHub público**.
- **Atenção:** quaisquer commits realizados **após** a data de entrega serão desconsiderados para avaliação — confirme e finalize o repositório antes da entrega.