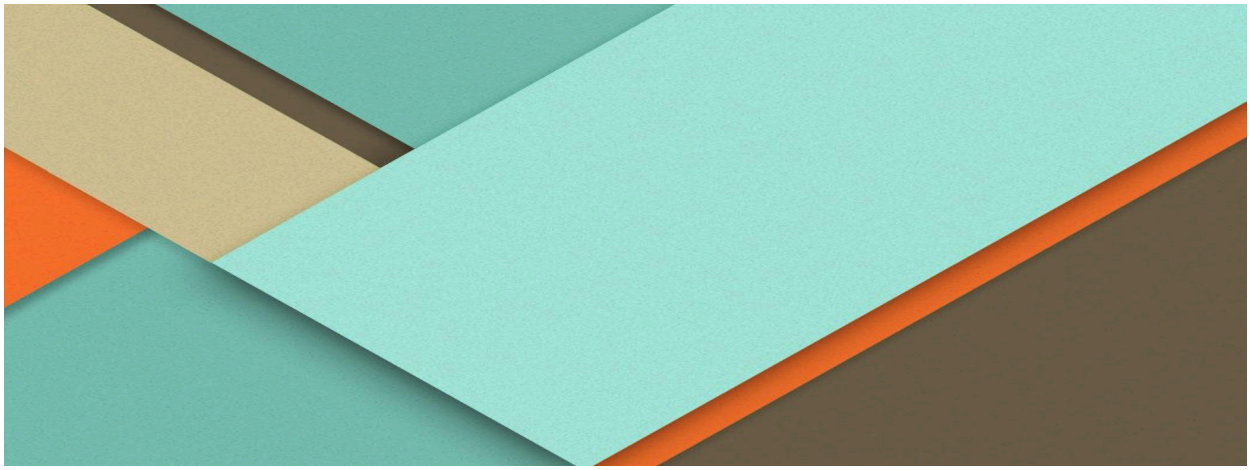


MVP CONCLUSÃO DE SPRINT - CRIAÇÃO DE UM PIPELINE DE DADOS COM DELTA LIVE TABLES

—Lucas de Oliveira Noronha

Visão geral

O notebook "vendas" prepara e organiza dados de vendas em um pipeline DLT. Inicia configurando o ambiente e copiando os dados necessários para o DBFS. Em seguida,

define uma tabela de fatos de vendas e tres dimensões (clientes, categorias e filial) para análise, garantindo a qualidade dos dados com expectativas DLT.

A base de dados é uma amostra de tres meses de vendas de uma empresa

https://raw.githubusercontent.com/LDONoronha/data_engineering/main/vendas.csv

Objetivos

1. Construir um data lake s para armazenar dados de emissões vendas de uma empresa.
2. Com essa base busco responder questões referentes a tendências de vendas e compras como por exemplo : Top 10 dos clientes com maior valor de compras em um respectivo mês; Qual departamento os clientes top 10 compram?

Plataforma

A plataforma escolhida foi a Databricks conforme orientações dos nossos professores.

Coleta de dados

O dataset escolhido foi uma base de vendas da empresa onde trabalho. Subi essa base em formato csv no meu repositório do github.

Link:
https://raw.githubusercontent.com/LDONoronha/data_engineering/main/vendas.csv

Modelagem e Carga

configuração do cluster:


```
{
   "cluster_name": "karolina kirst's Cluster",
   "spark_version": "14.3.x-scala2.12",
   "gcp_attributes": {
      "use_preemptible_executors": false,
```

```
        "availability": "ON_DEMAND_GCP",
        "zone_id": "auto"
    },
    "node_type_id": "e2-standard-4",
    "spark_env_vars": {
        "PYSPARK_PYTHON": "/databricks/python3/bin/python3"
    },
    "autotermination_minutes": 120,
    "single_user_name": "karolinakirst@gmail.com",
    "data_security_mode": "SINGLE_USER",
    "runtime_engine": "STANDARD",
    "autoscale": {
        "min_workers": 1,
        "max_workers": 3
    }
}
```


configuração do pipeline:


```
{
    "id": "6be27b57-cca0-4df6-a59b-637ffbee7732",
    "pipeline_type": "WORKSPACE",
    "clusters": [
        {
            "label": "default",
            "node_type_id": "n2-standard-4",
```

```
        "autoscale": {
            "min_workers": 1,
            "max_workers": 3,
            "mode": "ENHANCED"
        }
    }
],
"development": true,
"notifications": [
    {
        "email_recipients": [
            "noronharm2@gmail.com"
        ],
        "alerts": [
            "on-update-fatal-failure",
            "on-flow-failure",
            "on-update-success"
        ]
    }
],
"continuous": false,
"channel": "PREVIEW",
"photon": false,
"libraries": [
    {
        "notebook": {
            "path": "/Users/karolinakirst@gmail.com/vendas"
        }
    }
```

```
    ],
    "name": "vendas",
    "edition": "ADVANCED",
    "catalog": "lucasdeoliveira",
    "target": "vendas",
    "data_sampling": false
}
```

Notebook

```
3
# Esta célula importa o módulo dlt, utilizado para definir tabelas e transformações em pipelines Delta Live Tables (DLT), e todas as funções do módulo pyspark.sql.functions, que são usadas para manipulação de colunas e dados em DataFrames do PySpark.

import dlt
from pyspark.sql.functions import *
```

```
4
#Esta célula configura o ambiente, definindo variáveis de ambiente para o caminho do volume no catálogo Unity, a URL de download do conjunto de dados e o nome do arquivo. Em seguida, copia o arquivo CSV de vendas da URL especificada para o caminho do volume no Databricks File System (DBFS).

import os

os.environ["UNITY_CATALOG_VOLUME_PATH"] = "/Volumes/lucasdeoliveira/default/fatovendas/"
os.environ["DATASET_DOWNLOAD_URL"] = "https://raw.githubusercontent.com/LDONoronha/data_engineering/main/vendas.csv"
os.environ["DATASET_DOWNLOAD_FILENAME"] = "vendas.csv"

dbutils.fs.cp(f"{os.environ.get('DATASET_DOWNLOAD_URL')}", f"{os.environ.get('UNITY_CATALOG_VOLUME_PATH')}{os.environ.get('DATASET_DOWNLOAD_FILENAME')}")
```

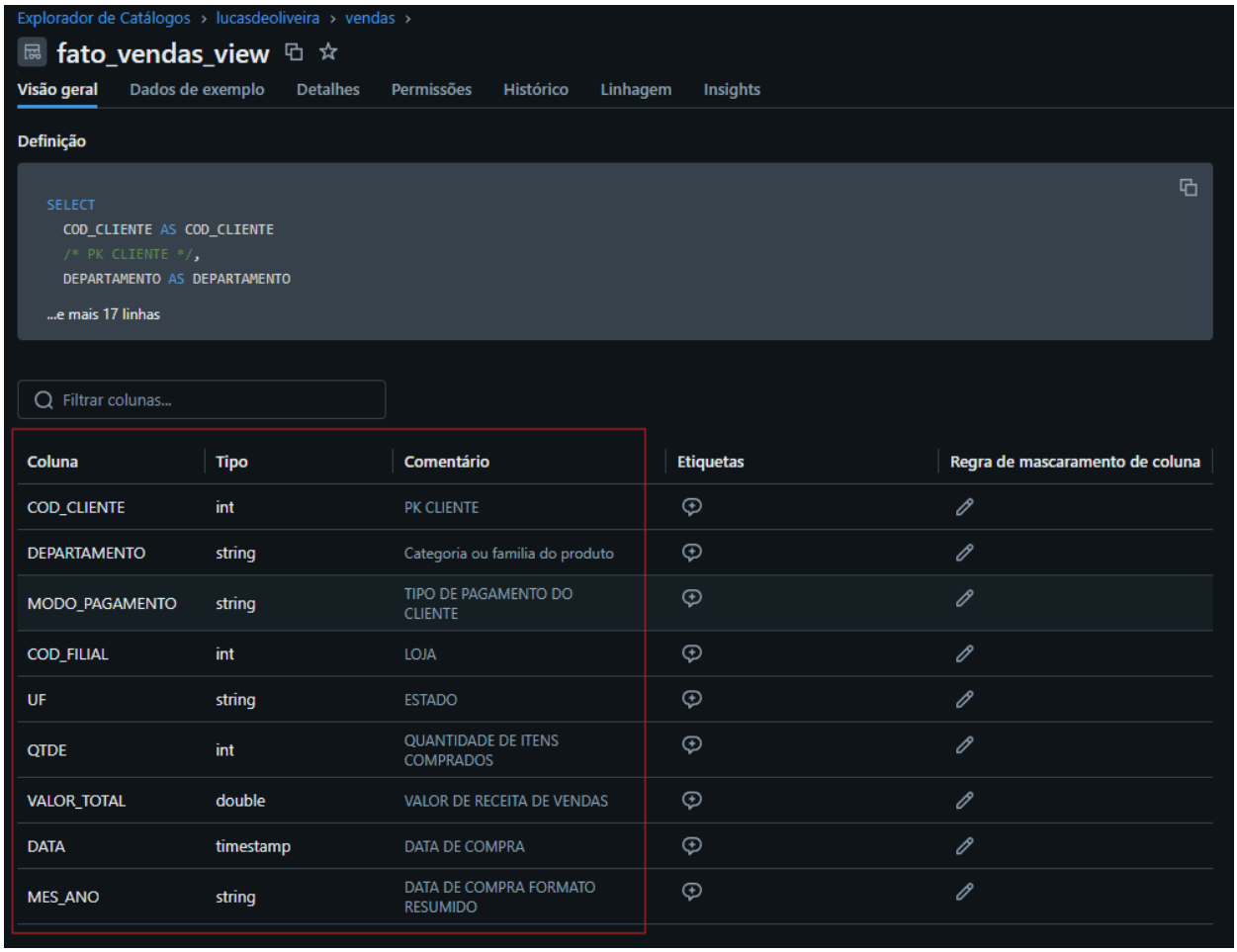
```
5
#Esta célula define uma tabela DLT chamada fato_vendas, contendo dados de vendas de um período de 3 meses. A função lê o arquivo CSV de vendas do caminho especificado, utilizando o Spark para inferir o esquema das colunas automaticamente e considerando a primeira linha como cabeçalho.

@dlt.table(
  comment="Base de dados com histórico de 3 meses de vendas de uma empresa."
)
def fato_vendas():
  df = spark.read.csv(f"{os.environ.get('UNITY_CATALOG_VOLUME_PATH')}{os.environ.get('DATASET_DOWNLOAD_FILENAME')}", header=True, inferSchema=True)
  return df
```

```
6                                                                                                                    Python
# Esta célula cria uma tabela DLT chamada dim_clientes, contendo uma lista única de códigos de clientes. Utiliza a expectativa dlt.expect para garantir que a coluna COD_CLIENTE não contenha valores nulos. A função seleciona a coluna COD_CLIENTE da tabela fato_vendas e remove duplicatas.

@dlt.table(
  comment="Base de Cliente."
)
@dlt.expect("valid_COD_CLIENTE", "COD_CLIENTE IS NOT NULL")
def dim_cliente():
  return (
    dlt.read("fato_vendas")
      .select("COD_CLIENTE")
      .distinct()
  )
```

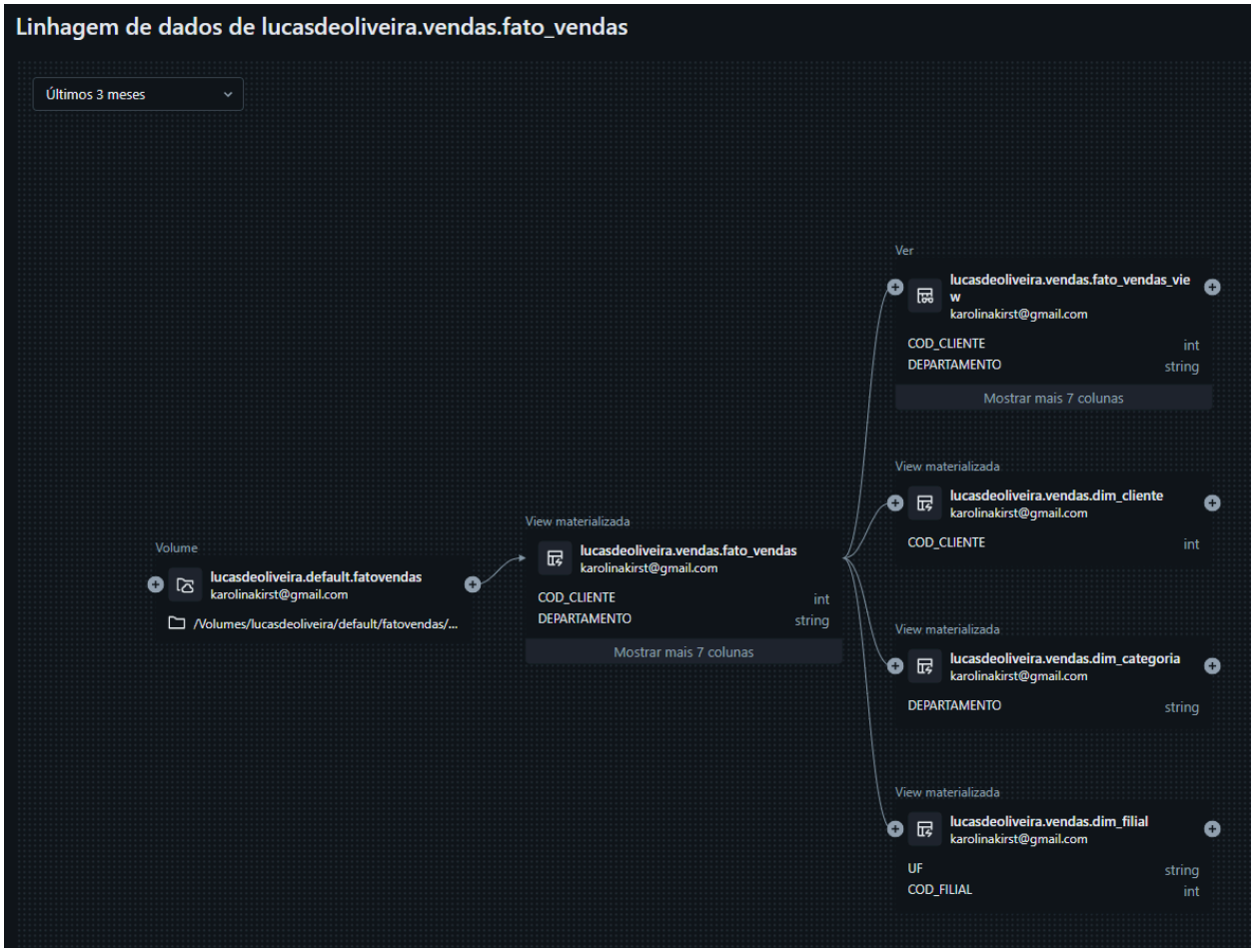
```
7
# Esta célula define outra tabela DLT, dim_categoria, contendo categorias únicas de produtos vendidos. Assim como na célula anterior, uma expectativa é definida para garantir que a coluna DEPARTAMENTO não tenha valores nulos. A função seleciona a coluna DEPARTAMENTO da tabela fato_vendas e aplica o método distinct() para remover duplicatas.

@dlt.table(
  comment="Categoria dos produtos vendidos."
)
@dlt.expect("valid_DEPARTAMENTO", "DEPARTAMENTO IS NOT NULL")
def dim_categoria():
  return (
    dlt.read("fato_vendas")
      .select("DEPARTAMENTO")
      .distinct()
  )
```

```
8

# Esta célula define outra tabela DLT, dim_filial, contendo filiais únicas (lojas de vendasd) por estados. Assim como na célula anterior, uma expectativa é definida para garantir que a coluna COD_FILIAL não tenha valores nulos. A
função seleciona as colunas UF e cod_filial da tabela fato_vendas e aplica o método distinct() para remover duplicatas.

@dlt.table(
  comment="Região e loja de compra."
)
@dlt.expect("valid_COD_FILIAL", "COD_FILIAL IS NOT NULL")
def dim_filial():
  return (
    dlt.read("fato_vendas")
      .select("UF","COD_FILIAL")
      .distinct()
  )
```

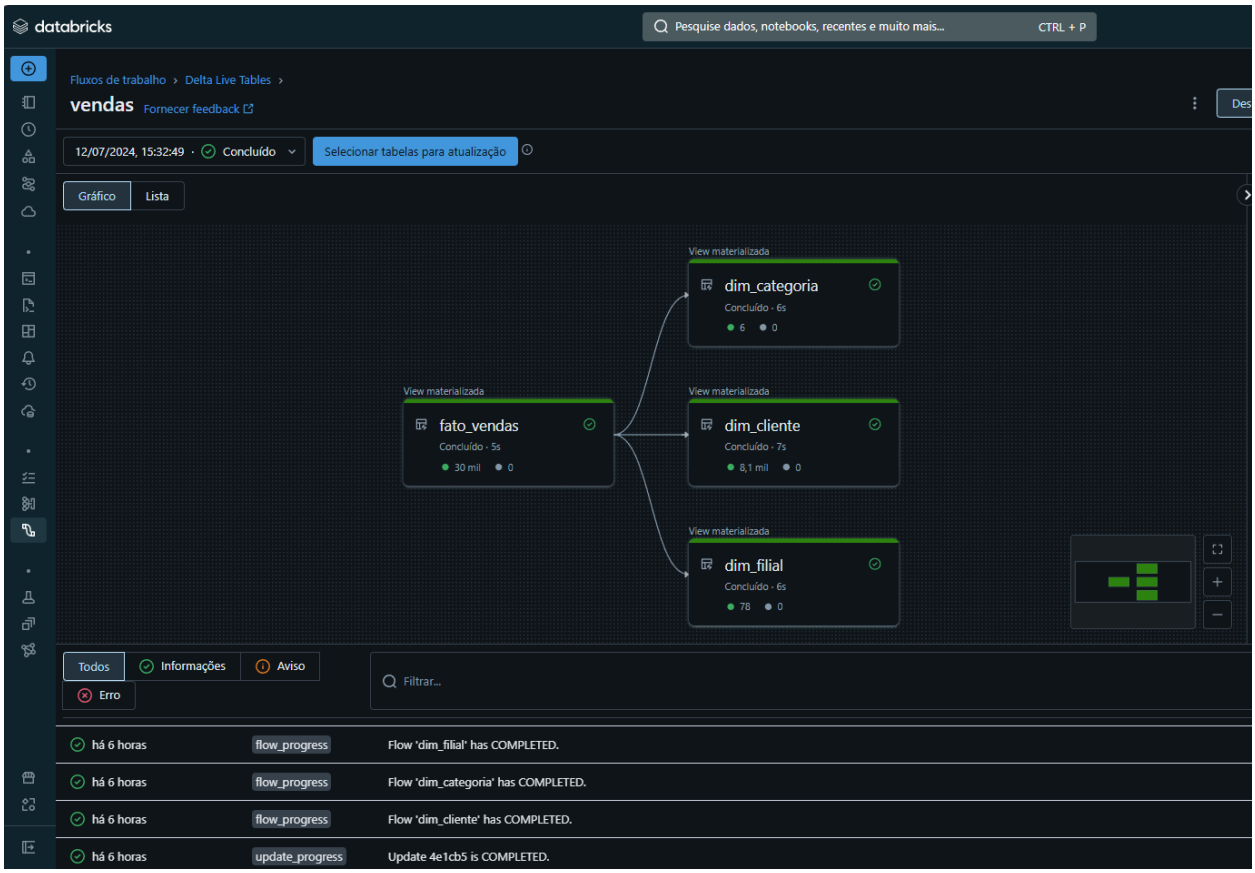
Metadados



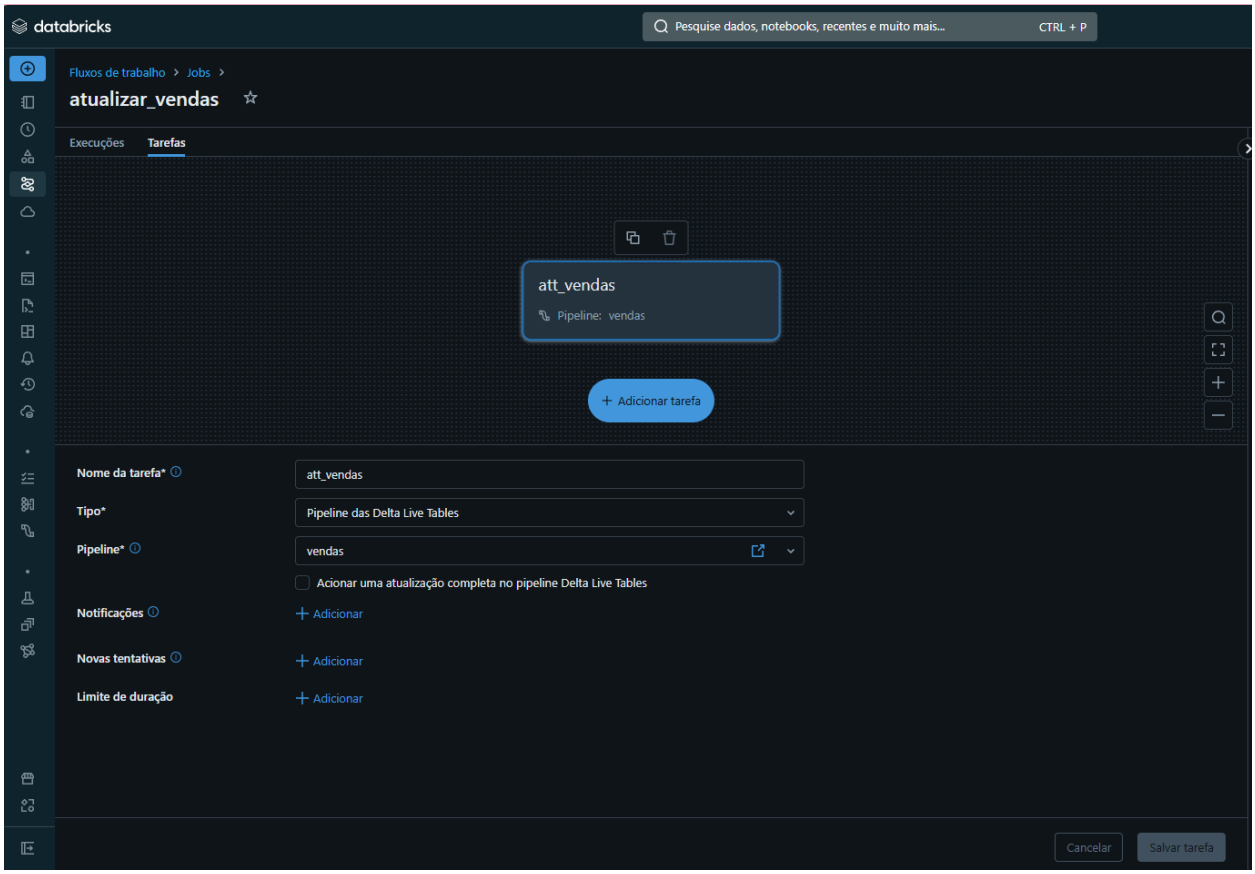
Linhagem dos dados

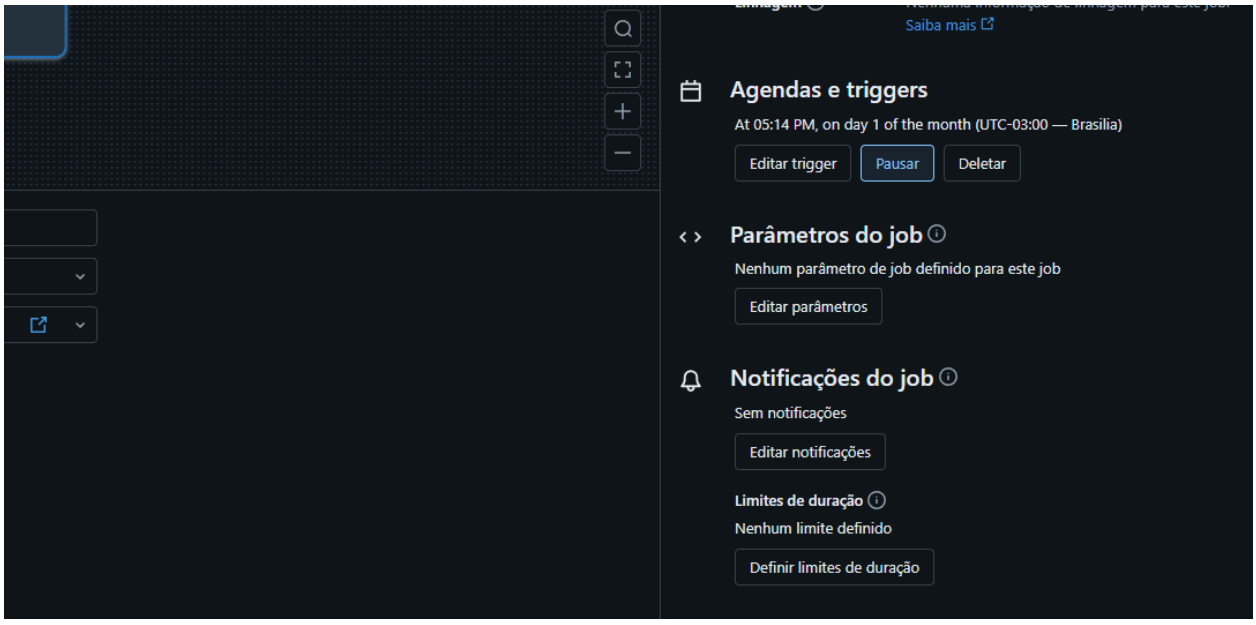


Esquema do Delta Live Tables

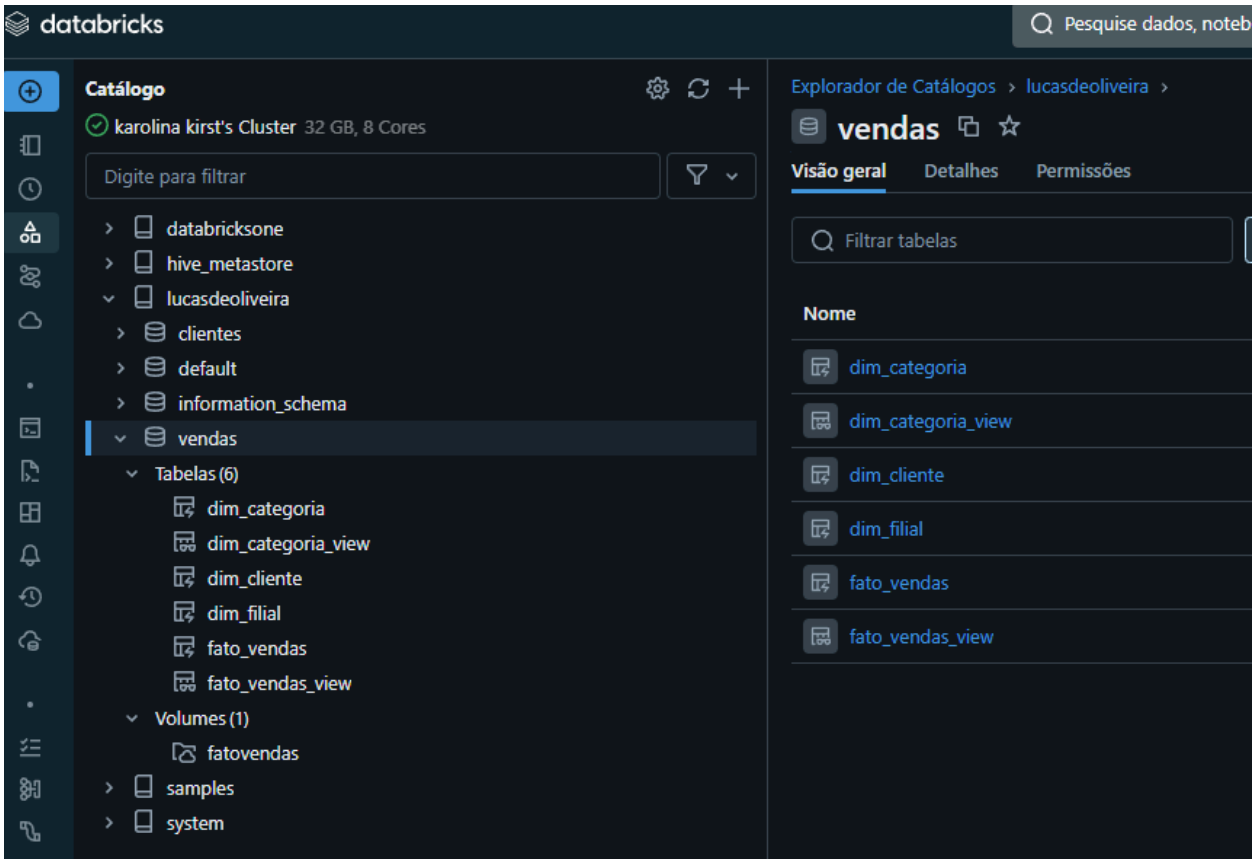


Agendamento dos jobs





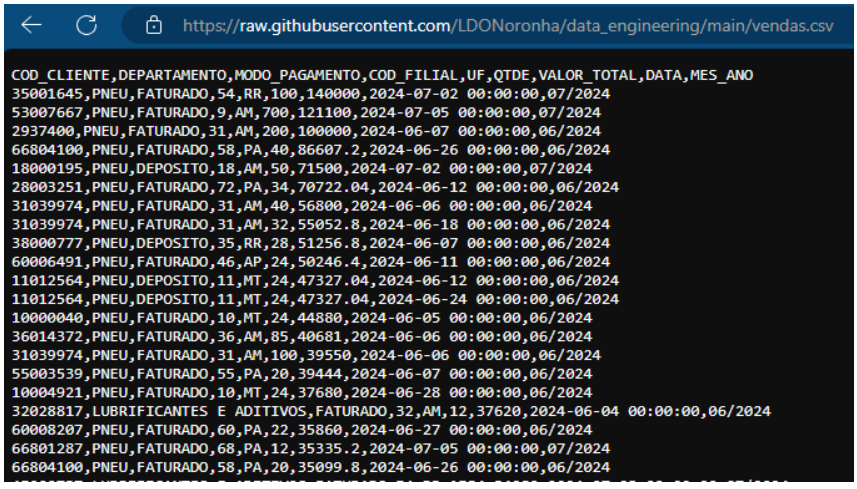
Catalágo de dados



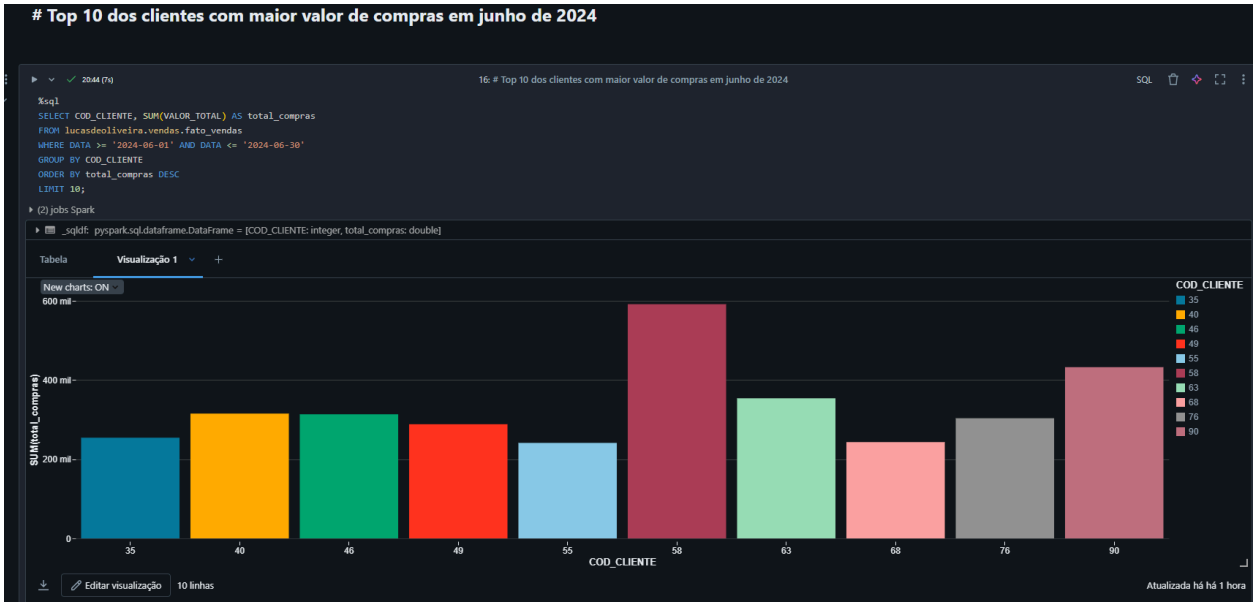
Análise

a. Qualidade de dados

No conjunto de dados não enfrentei problemas em relação a sua qualidade como com valores nulos e caracteres estranhos, pois o dataset estava bem limpo desde a sua origem.



b. Solução do problema



Autoavaliação

Esse projeto foi bem desafiador e bem empolgante de fazer, tive muitas dificuldades porém consegui atingir os objetivos e realizar o pipe line no databricks ajustando cotas e limites dos sistemas do google cloud. Sem dúvidas irei me aprofundar ainda mais nessa plataforma.