



Bootcamp: Engenheiro(a) de Dados

Documento Arquitetural

Autor: Francieli dos Santos Muniz

Data: 26/07/2025



Contexto do Projeto

Com o crescimento exponencial de dados nas organizações, torna-se essencial implementar soluções modernas de engenharia de dados que permitam a ingestão, processamento e disponibilização de informações de forma eficiente e escalável. Dentro desse cenário, foi proposto, como desafio final do Bootcamp de Engenharia de Dados, o desenvolvimento de uma arquitetura de dados em camadas utilizando tecnologias amplamente adotadas no mercado, como Apache Kafka, Apache Spark, PostgreSQL, Amazon S3 e Docker.

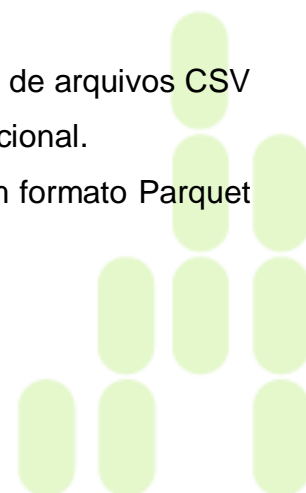
O projeto tem como objetivo simular uma esteira de dados real, com a ingestão de dados públicos do Tesouro Nacional (Tesouro Direto), aplicando o modelo de arquitetura em camadas (Bronze, Silver e Gold), com tratamento incremental dos dados e entrega final em um data lake para consumo analítico.

Objetivo Geral

Desenvolver uma pipeline ETL completa utilizando Apache Spark, Kafka, PostgreSQL e S3, estruturada segundo a arquitetura de dados em camadas (Bronze, Silver e Gold), capaz de consumir dados brutos de uma fonte externa (Tesouro Nacional), realizar o processamento e a transformação dos dados, e disponibilizá-los para consumo analítico em um data lake.

Objetivos Principais

1. Construir e orquestrar os serviços necessários utilizando Docker Compose, incluindo Kafka, Kafka Connect, PostgreSQL e Spark.
2. Realizar a ingestão bruta dos dados (camada Bronze) a partir de arquivos CSV disponibilizados por meio de uma URL pública do Tesouro Nacional.
3. Validar o schema dos dados e persistir os arquivos brutos em formato Parquet ou Delta no PostgreSQL e posteriormente no Kafka.



4. Processar os dados da camada Bronze com Apache Spark SQL para limpeza e transformação, formando a camada Silver, com tratamento de nulos, remoção de duplicados e normalização de colunas.
5. Gerar a camada Gold, contendo dados agregados e enriquecidos, prontos para análises, utilizando agregações e cálculos no Spark.
6. Configurar o Kafka Connect para integrar as diferentes camadas com o banco de dados PostgreSQL e com o data lake em Amazon S3.
7. Garantir a modularidade, clareza e eficiência do pipeline por meio de boas práticas de engenharia de dados.

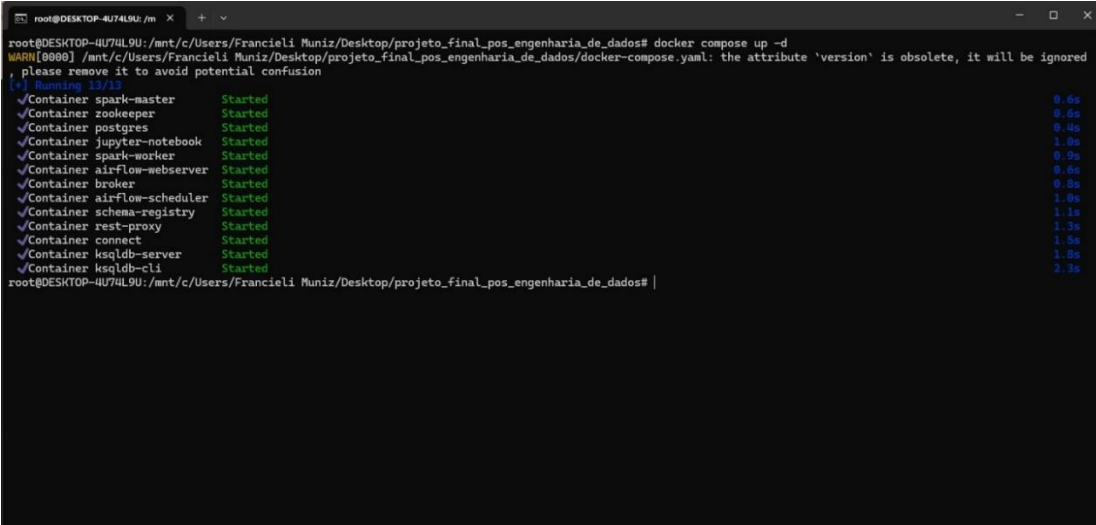
Entregáveis

Containers

Para a execução do projeto, foi criado um ambiente de desenvolvimento utilizando Docker Compose, responsável por orquestrar todos os serviços necessários, como Apache Kafka, Spark, PostgreSQL, Kafka Connect, Airflow, entre outros.

A imagem abaixo demonstra a execução bem-sucedida do comando `docker compose up -d`, com todos os containers iniciados corretamente. Esse ambiente simula uma arquitetura moderna de dados, permitindo a ingestão, o processamento e a entrega de dados em camadas (Bronze, Silver e Gold).

Imagem 1 – Ambiente de Desenvolvimento



```
root@DESKTOP-4U74L9U:/mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados# docker compose up -d
WARN[0000] /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/docker-compose.yaml: the attribute 'version' is obsolete, it will be ignored
, please remove it to avoid potential confusion
[*] Running 13/13
✓Container spark-master      Started      0.6s
✓Container zookeeper         Started      0.6s
✓Container postgres          Started      0.4s
✓Container jupyter-notebook   Started      1.0s
✓Container spark-worker       Started      0.9s
✓Container airflow-webserver  Started      0.6s
✓Container broker             Started      0.8s
✓Container airflow-scheduler  Started      1.0s
✓Container schema-registry    Started      1.1s
✓Container rest-proxy         Started      1.3s
✓Container connect            Started      1.5s
✓Container ksqldb-server      Started      1.8s
✓Container ksqldb-cli         Started      2.3s
root@DESKTOP-4U74L9U:/mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados#
```

Fonte: Própria autoria

Connectors

Para realizar a ingestão dos dados do banco PostgreSQL no Apache Kafka, foram utilizados Kafka Connectors do tipo JDBC Source, responsáveis por conectar o banco de dados relacional aos tópicos Kafka.

Imagem 2 – Kafka Connectors

```
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# curl -X POST -H "Content-Type: application/json" --data @connect_jdbc_postgres_ipca.config http://localhost:8083/connectors
{"name":"postg-connector-ipca","config":{"connector.class":"io.confluent.connect.jdbc.JdbcSourceConnector","tasks.max":"1","connection.url":"jdbc:postgresql://postgres:5432/postgres","connection.user":"postgres","connection.password":"postgres","mode":"timestamp","timestamp.column.name":"dt_update","table.whitelist":"public.dadostesouroipca","topic.prefix":"postgres-","validate.non.null":"false","poll.interval.ms":"500","name":"postg-connector-ipca"},"tasks":[],"type":"source"}
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# curl -X POST -H "Content-Type: application/json" --data @connect_jdbc_postgres_pre.config http://localhost:8083/connectors
{"name":"postg-connector-pre","config":{"connector.class":"io.confluent.connect.jdbc.JdbcSourceConnector","tasks.max":"1","connection.url":"jdbc:postgresql://postgres:5432/postgres","connection.user":"postgres","connection.password":"postgres","mode":"timestamp","timestamp.column.name":"dt_update","table.whitelist":"public.dadostesouropre","topic.prefix":"postgres-","validate.non.null":"false","poll.interval.ms":"500","name":"postg-connector-pre"},"tasks":[],"type":"source"}
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source#
```

Fonte: Própria autoria

Após a criação, o status dos conectores foi verificado com sucesso, apresentando o estado RUNNING para ambos, indicando que estão ativos e realizando a leitura e envio dos dados do banco para o Kafka.

Imagem 3 – Connectors Status

```
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# docker exec -it connect curl localhost:8083/connectors/postg-connector-ipca/status
{"name":"postg-connector-ipca","connector":{"state":"RUNNING","worker_id":"connect:8083"},"tasks":[{"id":0,"state":"RUNNING","worker_id":"connect:8083"}],"type":"source"}
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# docker exec -it connect curl localhost:8083/connectors/postg-connector-pre/status
{"name":"postg-connector-pre","connector":{"state":"RUNNING","worker_id":"connect:8083"},"tasks":[{"id":0,"state":"RUNNING","worker_id":"connect:8083"}],"type":"source"}
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source#
```

Fonte: Própria autoria

Tópicos Kafka

Foram criados dois tópicos no Apache Kafka: postgres-dadostesouroipca e postgres-dadostesouropre, ambos com uma partição e um fator de replicação igual a 1. Após a criação, a listagem confirmou o registro correto dos tópicos no cluster.

Imagem 4 – Criação e listagem de tópicos Kafka

```
@broker:~
(.venv) root@DESKTOP-4U74L9U:/mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# docker exec -it broker bash
[appuser@broker ~]$ kafka-topics --create --bootstrap-server localhost:9092 \
> --partitions 1 --replication-factor 1 --topic postgres-dadosesouropre
Created topic postgres-dadosesouropre.
[appuser@broker ~]$ kafka-topics --create --bootstrap-server localhost:9092 \
> --partitions 1 --replication-factor 1 --topic postgres-dadosesouroipca
Created topic postgres-dadosesouroipca.
[appuser@broker ~]$ kafka-topics --list --bootstrap-server localhost:9092
__consumer_offsets
__transaction_state
__confluent-ksql-default__command_topic
__schemas
default_ksql_processing_log
docker-connect-configs
docker-connect-offsets
docker-connect-status
postgres-dadosesouroipca
postgres-dadosesouropre
[appuser@broker ~]$
```

Fonte: Própria autoria

Foi realizada a construção de uma imagem Docker personalizada utilizando confluentinc/cp-kafka-connect-base. A imagem inclui o conector JDBC, necessário para integração com bases PostgreSQL.

Imagem 5 – Build de imagem customizada para Kafka Connect

```
root@DESKTOP-4U74L9U:/mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/custom-kafka-connector-i
image# docker buildx build . -t connect-custom:1.0.0
[+] Building 0.6s (6/6) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 243B                             0.0s
=> [internal] load metadata for docker.io/confluentinc/cp-kafka-connect-base:7.0.0 0.5s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [1/2] FROM docker.io/confluentinc/cp-kafka-connect-base:7.0.0@sha256:5d4f484e219ad704a735058531e33a5c0cc6bec8 0.0s
=> CACHED [2/2] RUN confluent-hub install --no-prompt confluentinc/kafka-connect-jdbc:10.4.1 && confluent-hu 0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:d8f75784ce0d5ccaf4a282f1cbd199a68102992a8c399fc7ae700f5dfa9c9265 0.0s
=> => naming to docker.io/library/connect-custom:1.0.0        0.0s
root@DESKTOP-4U74L9U:/mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/custom-kafka-connector-i
image#
```

Fonte: Própria autoria

Foi feito o comando --describe nos dois tópicos criados anteriormente, confirmando que possuem uma partição e estão corretamente replicados com Leader e Isr ativos.

Imagem 6 – Descrição dos tópicos Kafka

```
@broker:~  
root@DESKTOP-4U74U9U:/mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# docker exec -it broker bash  
[appuser@broker ~]$ kafka-topics \br/>  --bootstrap-server localhost:9092 \br/>  --describe \br/>  --topic postgres-dadostesouroipca  
Topic: postgres-dadostesouroipca TopicId: mJ2xMps6Rg-ULHHKmzqbwxw PartitionCount: 1 ReplicationFactor: 1 Configs:  
  Topic: postgres-dadostesouroipca Partition: 0 Leader: 1 Replicas: 1 Isr: 1  
[appuser@broker ~]$ kafka-topics \br/>  --bootstrap-server localhost:9092 \br/>  --describe \br/>  --topic postgres-dadostesouroipre  
Topic: postgres-dadostesouroipre TopicId: zqlmPrWIT-CtlqMvMWZacA PartitionCount: 1 ReplicationFactor: 1 Configs:  
  Topic: postgres-dadostesouroipre Partition: 0 Leader: 1 Replicas: 1 Isr: 1  
[appuser@broker ~]$
```

Fonte: Própria autoria

Visualização das mensagens trafegando no tópico postgres-dadostesouroipca, demonstrando os dados estruturados e serializados no formato JSON.

Imagem 7 – Dados trafegando no tópico IPCA

```
@broker:~  
["CompraManha":3.86,"VendaManha":3.94,"PUCompraManha":1453.65,"PUVendaManha":1449.85,"PUBaseManha":1448.22,"Data_Vencimento":1723680000000,"Data_Base":1360922400000,"Tipo":"IPCA","dt_update":1752928569709}  
["CompraManha":3.34,"VendaManha":3.4,"PUCompraManha":1829.28,"PUVendaManha":1822.67,"PUBaseManha":1821.9,"Data_Vencimento":1557878400000,"Data_Base":1360622400000,"Tipo":"IPCA","dt_update":1752928569710}  
["CompraManha":4.06,"VendaManha":4.16,"PUCompraManha":928.42,"PUVendaManha":980.85,"PUBaseManha":988.05,"Data_Vencimento":2062800000000,"Data_Base":1508022400000,"Tipo":"IPCA","dt_update":1752928569711}  
["CompraManha":3.27,"VendaManha":3.33,"PUCompraManha":1836.27,"PUVendaManha":1829.62,"PUBaseManha":1828.85,"Data_Vencimento":1557878400000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569711}  
["CompraManha":4.03,"VendaManha":4.13,"PUCompraManha":933.98,"PUVendaManha":914.27,"PUBaseManha":913.56,"Data_Vencimento":2062800000000,"Data_Base":1559936000000,"Tipo":"IPCA","dt_update":1752928569712}  
["CompraManha":2.36,"VendaManha":2.4,"PUCompraManha":2122.97,"PUVendaManha":2127.88,"PUBaseManha":2126.27,"Data_Vencimento":1431648000000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569712}  
["CompraManha":3.77,"VendaManha":3.85,"PUCompraManha":1467.58,"PUVendaManha":1451.62,"PUBaseManha":1453.88,"Data_Vencimento":1723680000000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569713}  
["CompraManha":3.98,"VendaManha":4.08,"PUCompraManha":942.58,"PUVendaManha":923.66,"PUBaseManha":922.71,"Data_Vencimento":2062800000000,"Data_Base":1559936000000,"Tipo":"IPCA","dt_update":1752928569714}  
["CompraManha":3.25,"VendaManha":3.31,"PUCompraManha":1837.72,"PUVendaManha":1831.07,"PUBaseManha":1829.24,"Data_Vencimento":1557878400000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569714}  
["CompraManha":3.68,"VendaManha":3.76,"PUCompraManha":1481.6,"PUVendaManha":1468.43,"PUBaseManha":1467.85,"Data_Vencimento":1723680000000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569715}  
["CompraManha":2.31,"VendaManha":2.35,"PUCompraManha":2139.52,"PUVendaManha":2128.63,"PUBaseManha":2126.59,"Data_Vencimento":1431648000000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569716}  
["CompraManha":4.32,"VendaManha":4.34,"PUCompraManha":2182.13,"PUVendaManha":2181.5,"PUBaseManha":2180.72,"Data_Vencimento":1431648000000,"Data_Base":1359936000000,"Tipo":"IPCA","dt_update":1752928569716}  
["CompraManha":6.0,"VendaManha":6.08,"PUCompraManha":1248.21,"PUVendaManha":1235.18,"PUBaseManha":1239.65,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569717}  
["CompraManha":5.86,"VendaManha":5.92,"PUCompraManha":1782.33,"PUVendaManha":1697.65,"PUBaseManha":1696.34,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569718}  
["CompraManha":6.05,"VendaManha":6.15,"PUCompraManha":660.41,"PUVendaManha":647.21,"PUBaseManha":646.93,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569718}  
["CompraManha":5.74,"VendaManha":5.8,"PUCompraManha":1912.26,"PUVendaManha":1786.22,"PUBaseManha":1786.21,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569719}  
["CompraManha":4.23,"VendaManha":4.25,"PUCompraManha":2184.2,"PUVendaManha":2183.57,"PUBaseManha":2182.79,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569719}  
["CompraManha":5.9,"VendaManha":5.98,"PUCompraManha":1257.35,"PUVendaManha":1247.21,"PUBaseManha":1246.68,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569720}  
["CompraManha":5.98,"VendaManha":6.08,"PUCompraManha":660.55,"PUVendaManha":655.15,"PUBaseManha":655.87,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569721}  
["CompraManha":4.21,"VendaManha":4.23,"PUCompraManha":2184.86,"PUVendaManha":2183.43,"PUBaseManha":2182.65,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569721}  
["CompraManha":5.94,"VendaManha":6.04,"PUCompraManha":674.7,"PUVendaManha":661.19,"PUBaseManha":660.91,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569722}  
["CompraManha":5.69,"VendaManha":5.76,"PUCompraManha":1715.86,"PUVendaManha":1710.65,"PUBaseManha":1709.94,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569723}  
["CompraManha":5.86,"VendaManha":5.94,"PUCompraManha":1261.93,"PUVendaManha":1251.74,"PUBaseManha":1251.22,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569724}  
["CompraManha":5.75,"VendaManha":5.83,"PUCompraManha":1275.56,"PUVendaManha":1265.25,"PUBaseManha":1264.25,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569724}  
["CompraManha":5.68,"VendaManha":5.76,"PUCompraManha":683.4,"PUVendaManha":671.7,"PUBaseManha":671.16,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569725}  
["CompraManha":4.11,"VendaManha":4.13,"PUCompraManha":2186.48,"PUVendaManha":2185.84,"PUBaseManha":2184.25,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569726}  
["CompraManha":5.88,"VendaManha":5.94,"PUCompraManha":1725.12,"PUVendaManha":1719.74,"PUBaseManha":1718.39,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569726}  
["CompraManha":5.33,"VendaManha":5.42,"PUCompraManha":706.82,"PUVendaManha":721.61,"PUBaseManha":721.46,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569727}  
["CompraManha":3.72,"VendaManha":3.74,"PUCompraManha":2159.21,"PUVendaManha":2158.48,"PUBaseManha":2158.14,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569728}  
["CompraManha":4.99,"VendaManha":5.05,"PUCompraManha":1741.88,"PUVendaManha":1736.2,"PUBaseManha":1735.84,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569728}  
["CompraManha":5.24,"VendaManha":5.32,"PUCompraManha":1331.71,"PUVendaManha":1303.8,"PUBaseManha":1303.52,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569729}  
["CompraManha":3.82,"VendaManha":3.84,"PUCompraManha":2155.22,"PUVendaManha":2154.49,"PUBaseManha":2154.11,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569729}  
["CompraManha":5.35,"VendaManha":5.45,"PUCompraManha":743.59,"PUVendaManha":728.05,"PUBaseManha":728.28,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569730}  
["CompraManha":5.3,"VendaManha":5.39,"PUCompraManha":1306.26,"PUVendaManha":1326.4,"PUBaseManha":1326.89,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569731}  
["CompraManha":5.03,"VendaManha":5.09,"PUCompraManha":1777.73,"PUVendaManha":1732.86,"PUBaseManha":1731.66,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569731}  
["CompraManha":3.9,"VendaManha":3.92,"PUCompraManha":2151.91,"PUVendaManha":2151.18,"PUBaseManha":2150.83,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569732}  
["CompraManha":5.07,"VendaManha":5.13,"PUCompraManha":1723.85,"PUVendaManha":1727.69,"PUBaseManha":1727.53,"Data_Vencimento":1557878400000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569733}  
["CompraManha":5.33,"VendaManha":5.41,"PUCompraManha":1301.86,"PUVendaManha":1291.05,"PUBaseManha":1290.77,"Data_Vencimento":1723680000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569734}  
["CompraManha":5.38,"VendaManha":5.48,"PUCompraManha":738.84,"PUVendaManha":723.8,"PUBaseManha":723.64,"Data_Vencimento":2062800000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569734}  
["CompraManha":4.03,"VendaManha":4.05,"PUCompraManha":2146.82,"PUVendaManha":2146.09,"PUBaseManha":2145.73,"Data_Vencimento":1431648000000,"Data_Base":1383696000000,"Tipo":"IPCA","dt_update":1752928569735}
```

Fonte: Própria autoria

Assim como no tópico IPCA, foram exibidos dados provenientes do conector de dados pré-fixados, com estrutura similar e organizados em mensagens JSON.

Imagem 8 – Dados trafegando no tópico PRÉ-FIXADOS

```

[Comprafanha]-14.65, 'VendaFanha'-14.7, 'PUComprafanha'-783.8, 'PUVendaFanha'-783.2, 'PUBaseFanha'-782.77, 'Data_Vencimento'-1199155600000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651551
[Comprafanha]-15.12, 'VendaFanha'-15.16, 'PUComprafanha'-834.76, 'PUVendaFanha'-834.52, 'PUBaseFanha'-831.62, 'Data_Vencimento'-1167699600000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651552
[Comprafanha]-14.56, 'VendaFanha'-14.62, 'PUComprafanha'-734.24, 'PUVendaFanha'-733.64, 'PUBaseFanha'-733.05, 'Data_Vencimento'-1212487040000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651552
[Comprafanha]-15.91, 'VendaFanha'-15.94, 'PUComprafanha'-997.57, 'PUVendaFanha'-997.49, 'PUBaseFanha'-995.85, 'Data_Vencimento'-1197112000000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651553
[Comprafanha]-15.91, 'VendaFanha'-15.94, 'PUComprafanha'-997.57, 'PUVendaFanha'-997.49, 'PUBaseFanha'-995.85, 'Data_Vencimento'-1197112000000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651553
[Comprafanha]-15.4, 'VendaFanha'-15.44, 'PUComprafanha'-1924.55, 'PUVendaFanha'-1924.38, 'PUBaseFanha'-1923.85, 'Data_Vencimento'-1159668800000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651554
[Comprafanha]-15.61, 'VendaFanha'-15.65, 'PUComprafanha'-884.07, 'PUVendaFanha'-884.36, 'PUBaseFanha'-881.33, 'Data_Vencimento'-1176315600000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651555
[Comprafanha]-15.91, 'VendaFanha'-15.94, 'PUComprafanha'-997.57, 'PUVendaFanha'-997.49, 'PUBaseFanha'-995.85, 'Data_Vencimento'-1197112000000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651555
[Comprafanha]-14.84, 'VendaFanha'-14.89, 'PUComprafanha'-860.17, 'PUVendaFanha'-860.57, 'PUBaseFanha'-857.13, 'Data_Vencimento'-1191368000000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651556
[Comprafanha]-14.72, 'VendaFanha'-14.77, 'PUComprafanha'-782.53, 'PUVendaFanha'-781.92, 'PUBaseFanha'-781.49, 'Data_Vencimento'-1199155600000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651557
[Comprafanha]-14.72, 'VendaFanha'-14.77, 'PUComprafanha'-782.53, 'PUVendaFanha'-781.92, 'PUBaseFanha'-781.49, 'Data_Vencimento'-1199155600000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651557
[Comprafanha]-14.61, 'VendaFanha'-14.67, 'PUComprafanha'-732.33, 'PUVendaFanha'-732.52, 'PUBaseFanha'-732.12, 'Data_Vencimento'-1212487040000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651558
[Comprafanha]-15.92, 'VendaFanha'-15.95, 'PUComprafanha'-956.98, 'PUVendaFanha'-956.91, 'PUBaseFanha'-956.34, 'Data_Vencimento'-1197112000000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651559
[Comprafanha]-15.92, 'VendaFanha'-15.95, 'PUComprafanha'-956.98, 'PUVendaFanha'-956.91, 'PUBaseFanha'-956.34, 'Data_Vencimento'-1197112000000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651559
[Comprafanha]-15.45, 'VendaFanha'-15.49, 'PUComprafanha'-923.81, 'PUVendaFanha'-923.63, 'PUBaseFanha'-921.31, 'Data_Vencimento'-1159668800000, 'Data_Base'-1142239400000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651560
[Comprafanha]-16.04, 'VendaFanha'-16.09, 'PUComprafanha'-775.26, 'PUVendaFanha'-771.65, 'PUBaseFanha'-770.21, 'Data_Vencimento'-1199155600000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651561
[Comprafanha]-16.04, 'VendaFanha'-16.09, 'PUComprafanha'-775.26, 'PUVendaFanha'-771.65, 'PUBaseFanha'-770.21, 'Data_Vencimento'-1199155600000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651561
[Comprafanha]-16.3, 'VendaFanha'-16.34, 'PUComprafanha'-889.15, 'PUVendaFanha'-889.75, 'PUBaseFanha'-886.79, 'Data_Vencimento'-1183248000000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651562
[Comprafanha]-16.46, 'VendaFanha'-16.5, 'PUComprafanha'-828.61, 'PUVendaFanha'-828.72, 'PUBaseFanha'-828.21, 'Data_Vencimento'-1175356000000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651563
[Comprafanha]-16.46, 'VendaFanha'-16.5, 'PUComprafanha'-828.61, 'PUVendaFanha'-828.72, 'PUBaseFanha'-828.21, 'Data_Vencimento'-1175356000000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651563
[Comprafanha]-17.04, 'VendaFanha'-17.07, 'PUComprafanha'-963.03, 'PUVendaFanha'-963.95, 'PUBaseFanha'-960.83, 'Data_Vencimento'-1183248000000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651564
[Comprafanha]-17.04, 'VendaFanha'-17.07, 'PUComprafanha'-963.03, 'PUVendaFanha'-963.95, 'PUBaseFanha'-960.83, 'Data_Vencimento'-1183248000000, 'Data_Base'-1136160000000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651564
[Comprafanha]-17.54, 'VendaFanha'-17.57, 'PUComprafanha'-1821.01, 'PUVendaFanha'-1821.61, 'PUBaseFanha'-1818.24, 'Data_Vencimento'-1135990800000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651565
[Comprafanha]-17.07, 'VendaFanha'-17.1, 'PUComprafanha'-925.95, 'PUVendaFanha'-925.84, 'PUBaseFanha'-925.26, 'Data_Vencimento'-1153171200000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651566
[Comprafanha]-17.07, 'VendaFanha'-17.1, 'PUComprafanha'-925.95, 'PUVendaFanha'-925.84, 'PUBaseFanha'-925.26, 'Data_Vencimento'-1153171200000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651566
[Comprafanha]-16.29, 'VendaFanha'-16.31, 'PUComprafanha'-799.9, 'PUVendaFanha'-799.9, 'PUBaseFanha'-788.82, 'Data_Vencimento'-1135990800000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651567
[Comprafanha]-16.62, 'VendaFanha'-16.68, 'PUComprafanha'-745.1, 'PUVendaFanha'-744.33, 'PUBaseFanha'-742.84, 'Data_Vencimento'-1159915600000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651568
[Comprafanha]-15.93, 'VendaFanha'-16.03, 'PUComprafanha'-680.3, 'PUVendaFanha'-680.3, 'PUBaseFanha'-677.8, 'Data_Vencimento'-1135990800000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651569
[Comprafanha]-16.25, 'VendaFanha'-16.25, 'PUComprafanha'-823.21, 'PUVendaFanha'-823.21, 'PUBaseFanha'-823.71, 'Data_Vencimento'-1175356000000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651568
[Comprafanha]-16.64, 'VendaFanha'-16.68, 'PUComprafanha'-858.91, 'PUVendaFanha'-858.62, 'PUBaseFanha'-858.69, 'Data_Vencimento'-1167699600000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651568
[Comprafanha]-16.79, 'VendaFanha'-16.83, 'PUComprafanha'-898.66, 'PUVendaFanha'-898.43, 'PUBaseFanha'-898.83, 'Data_Vencimento'-1159668800000, 'Data_Base'-1135990800000, 'Tipo'-'PRE-FIXADOS', 'dt_update'-1752528651569
[Comprafanha]-16.79, 'VendaFanha'-16.83, 'PUComprafanha'-898.66, 'PUVendaFanha'-898.43, 'PUBaseFanha'-898.83, 'Data_Vencimento'-1
```

Fonte: Própria autoria

Leitura dos tópicos Kafka `postgres-dadosesouroipca_gold` e `postgres-dadosesouropre_gold`, já com dados transformados. Os campos calculados incluem médias, diferenças percentuais e tipos agregados por categoria (IPCA e PRÉ).

Imagem 9 – Leitura dos dados processados em camadas GOLD

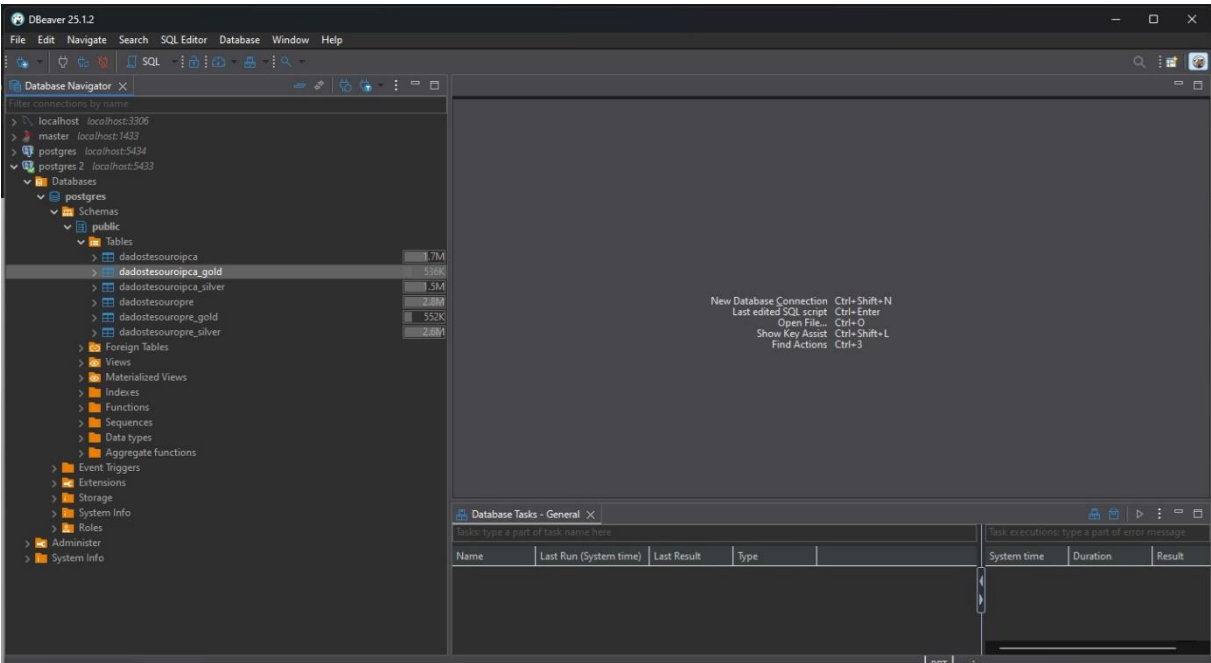
```
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# docker exec -it broker kafka-console-consumer --bootstrap-server broker:9092 --topic postgres-dadosetouroipca_gold --from-beginning --max-messages 5
{"Data_Base":12982,"Tipo":"IPCA","qtde_registros":4,"compra_manha_media":8.775,"venda_manha_media":8.845,"pu_compra_manha_media":491.69000000000005,"pu_venda_manha_media":487.735,"pu_base_manha_media":487.51,"percentual_diferenca_compra_venda":0.797278939835898}
{"Data_Base":12983,"Tipo":"IPCA","qtde_registros":4,"compra_manha_media":8.775,"venda_manha_media":8.845,"pu_compra_manha_media":491.91999999999996,"pu_venda_manha_media":487.96,"pu_base_manha_media":487.735,"percentual_diferenca_compra_venda":0.797278939835898}
{"Data_Base":12984,"Tipo":"IPCA","qtde_registros":4,"compra_manha_media":8.775,"venda_manha_media":8.845,"pu_compra_manha_media":492.84999999999999,"pu_venda_manha_media":488.225,"pu_base_manha_media":487.995,"percentual_diferenca_compra_venda":0.7971573509128669}
{"Data_Base":12985,"Tipo":"IPCA","qtde_registros":4,"compra_manha_media":8.775,"venda_manha_media":8.845,"pu_compra_manha_media":492.40999999999997,"pu_venda_manha_media":488.45500000000004,"pu_base_manha_media":488.225,"percentual_diferenca_compra_venda":0.7971573509128669}
{"Data_Base":12986,"Tipo":"IPCA","qtde_registros":4,"compra_manha_media":8.775,"venda_manha_media":8.845,"pu_compra_manha_media":492.76,"pu_venda_manha_media":488.895,"pu_base_manha_media":488.45500000000004,"percentual_diferenca_compra_venda":0.7971573509128669}
Processed a total of 5 messages
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source#
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source#
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source#
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source# docker exec -it broker kafka-console-consumer --bootstrap-server broker:9092 --topic postgres-dadosetouroipca_gold --from-beginning --max-messages 5
{"Data_Base":12783,"Tipo":"PRE-FIXADOS","qtde_registros":12,"compra_manha_media":18.153333333333336,"venda_manha_media":18.213333333333335,"pu_compra_manha_media":896.61500000000001,"pu_venda_manha_media":898.28000000000002,"pu_base_manha_media":897.6833333333334,"percentual_diferenca_compra_venda":0.3309412768097963}
{"Data_Base":12786,"Tipo":"PRE-FIXADOS","qtde_registros":12,"compra_manha_media":18.153333333333332,"venda_manha_media":18.213333333333335,"pu_compra_manha_media":899.21,"pu_venda_manha_media":898.8783333333332,"pu_base_manha_media":898.28000000000002,"percentual_diferenca_compra_venda":0.3309412768097963}
{"Data_Base":12787,"Tipo":"PRE-FIXADOS","qtde_registros":10,"compra_manha_media":18.3,"venda_manha_media":18.36,"pu_compra_manha_media":879.238,"pu_venda_manha_media":878.85600000000001,"pu_base_manha_media":878.268,"percentual_diferenca_compra_venda":0.3292459795576596}
{"Data_Base":12788,"Tipo":"PRE-FIXADOS","qtde_registros":10,"compra_manha_media":18.294,"venda_manha_media":18.359999999999996,"pu_compra_manha_media":879.8520000000001,"pu_venda_manha_media":879.47,"pu_base_manha_media":878.884,"percentual_diferenca_compra_venda":0.3292993862131134}
{"Data_Base":12789,"Tipo":"PRE-FIXADOS","qtde_registros":10,"compra_manha_media":18.344,"venda_manha_media":18.4,"pu_compra_manha_media":880.09999999999998,"pu_venda_manha_media":879.74799999999999,"pu_base_manha_media":879.16199999999999,"percentual_diferenca_compra_venda":0.30639787640791394}
Processed a total of 5 messages
root@DESKTOP-4U74L9U: /mnt/c/Users/Francieli Muniz/Desktop/projeto_final_pos_engenharia_de_dados/connectors/source#
```

Fonte: Própria autoria

PostgreSQL (DBeaver)

As imagens 10, 11 e 12 mostram a estrutura de tabelas no banco PostgreSQL conectada via DBeaver, organizadas conforme a arquitetura de dados em camadas: Bronze (dadostesouroipca, dadostesouropre), Silver (*_silver) e Gold (*_gold). Essa separação visa representar a evolução do dado bruto até informações refinadas e preparadas para análise.

Imagem 10 – Organização das camadas no banco de dados PostgreSQL



Fonte: Própria autoria

Imagem 11 – Leitura dos dados processados em camadas Bronze (IPCA)

	CompraManha	VendaManha	PUCompraManha	PUVendaManha	PUBaseManha	Data_Vencimento	Data_Base	A2_Tipo	dt_update
1	8.84	8.92	325.74	321.33	320.98	2024-08-15 00:00:00.000	2005-12-29 00:00:00.000	IPCA	2025-07-19 09:40:40.145
2	8.65	8.71	722.67	718.96	718.18	2015-05-15 00:00:00.000	2005-12-29 00:00:00.000	IPCA	2025-07-19 09:40:40.144
3	8.65	8.71	722.67	718.96	718.5	2015-05-15 00:00:00.000	2005-12-30 00:00:00.000	IPCA	2025-07-19 09:40:40.144
4	8.84	8.92	325.74	321.33	321.12	2024-08-15 00:00:00.000	2005-12-30 00:00:00.000	IPCA	2025-07-19 09:40:40.143
5	8.66	8.72	722.36	718.65	718.34	2015-05-15 00:00:00.000	2006-01-02 00:00:00.000	IPCA	2025-07-19 09:40:40.142

Fonte: Própria autoria

Imagem 12 – Leitura dos dados processados em camadas Bronze (Pre)

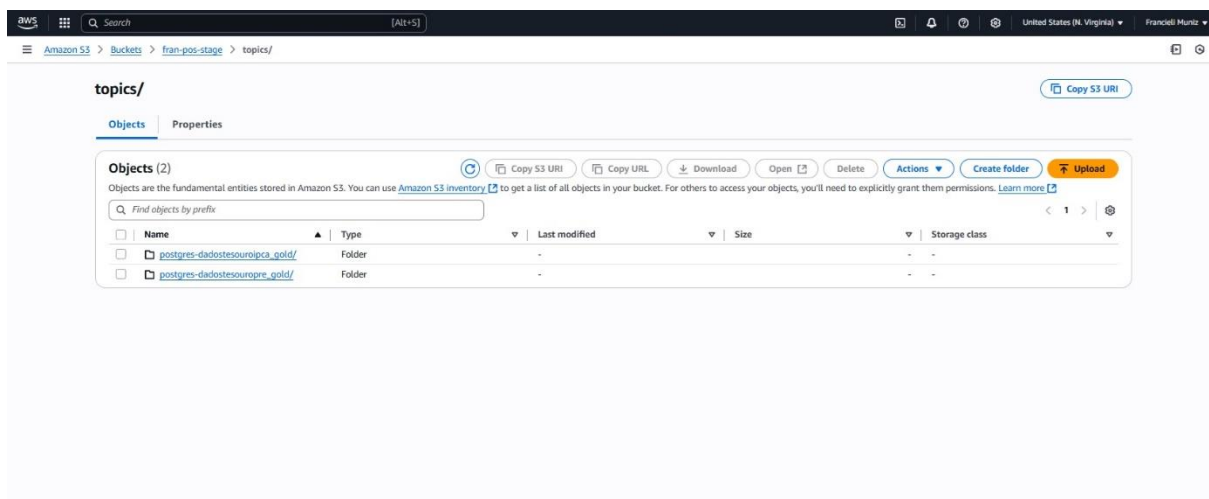
	ComprManha	VendaManha	PUCompraManha	PUVendaManha	PUBaseManha	Data_Vencimento	Data_Base	A2 Tipo	dt_update
1	17.56	17.59	960.36	960.3	959.06	2006-04-01 00:00:00.000	2005-12-29 00:00:00.000	PRE-FIXADOS	2025-07-19 09:40:29.172
2	16.63	16.67	858.98	858.69	857.64	2007-01-01 00:00:00.000	2005-12-29 00:00:00.000	PRE-FIXADOS	2025-07-19 09:40:29.171
3	16.82	16.86	890.49	890.26	889.16	2006-10-01 00:00:00.000	2005-12-29 00:00:00.000	PRE-FIXADOS	2025-07-19 09:40:29.171
4	17.09	17.13	925.3	925.14	923.98	2006-07-01 00:00:00.000	2005-12-29 00:00:00.000	PRE-FIXADOS	2025-07-19 09:40:29.170
5	16.48	16.52	828.39	828.04	827.03	2007-04-01 00:00:00.000	2005-12-29 00:00:00.000	PRE-FIXADOS	2025-07-19 09:40:29.169

Fonte: Própria autoria

DataLake Amazon S3

Esta imagem exhibe a organização de objetos no bucket Gold do Amazon S3, destacando metadados como nome, tipo, data de modificação e classe de armazenamento.

Imagem 13 – Estrutura de Objetos na Camada Gold (S3)



Fonte: Própria autoria

Nas imagens 14 e 15, observa-se os conjuntos de arquivos no formato JSON, particionados no S3 para otimização de consultas.

Imagem 14 – Dados do IPCA Armazenados no S3 com Particionamento

Name	Type	Last modified	Size	Storage class
postgres-dadosesouroipca_gold+0+0000000000.json	json	July 19, 2025, 18:01:36 (UTC-03:00)	526.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000002.json	json	July 19, 2025, 18:01:37 (UTC-03:00)	565.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000004.json	json	July 19, 2025, 18:01:38 (UTC-03:00)	575.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000006.json	json	July 19, 2025, 18:01:38 (UTC-03:00)	587.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000008.json	json	July 19, 2025, 18:01:39 (UTC-03:00)	563.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000010.json	json	July 19, 2025, 18:01:40 (UTC-03:00)	552.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000012.json	json	July 19, 2025, 18:01:41 (UTC-03:00)	563.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000014.json	json	July 19, 2025, 18:01:41 (UTC-03:00)	551.0 B	Standard
postgres-dadosesouroipca_gold+0+0000000016.json	json	July 19, 2025, 18:01:42 (UTC-03:00)	564.0 B	Standard

Fonte: Própria autoria

Imagem 15 – Dados do PRE Armazenados no S3 com Particionamento

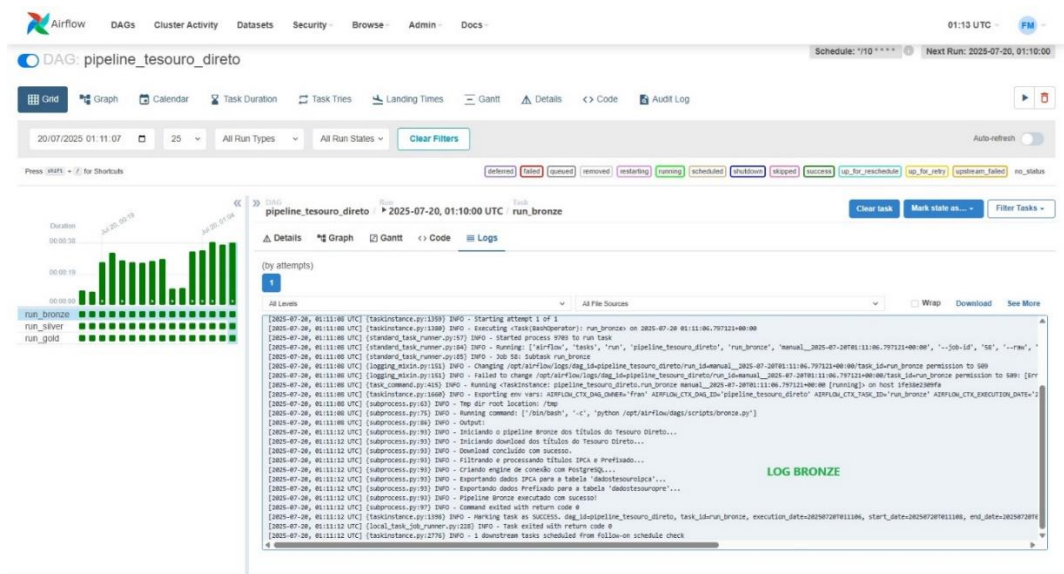
Name	Type	Last modified	Size	Storage class
postgres-dadosesouropre_gold+0+0000000000.json	json	July 19, 2025, 18:01:40 (UTC-03:00)	625.0 B	Standard
postgres-dadosesouropre_gold+0+0000000002.json	json	July 19, 2025, 18:01:41 (UTC-03:00)	557.0 B	Standard
postgres-dadosesouropre_gold+0+0000000004.json	json	July 19, 2025, 18:01:42 (UTC-03:00)	578.0 B	Standard
postgres-dadosesouropre_gold+0+0000000006.json	json	July 19, 2025, 18:01:43 (UTC-03:00)	557.0 B	Standard
postgres-dadosesouropre_gold+0+0000000008.json	json	July 19, 2025, 18:01:43 (UTC-03:00)	601.0 B	Standard
postgres-dadosesouropre_gold+0+0000000010.json	json	July 19, 2025, 18:01:44 (UTC-03:00)	581.0 B	Standard
postgres-dadosesouropre_gold+0+0000000012.json	json	July 19, 2025, 18:01:45 (UTC-03:00)	568.0 B	Standard
postgres-dadosesouropre_gold+0+0000000014.json	json	July 19, 2025, 18:01:46 (UTC-03:00)	579.0 B	Standard
postgres-dadosesouropre_gold+0+0000000016.json	json	July 19, 2025, 18:01:46 (UTC-03:00)	595.0 B	Standard

Fonte: Própria autoria

Airflow

Este log mostra a execução bem-sucedida da task run_bronze. Ela é responsável pela ingestão de dados brutos do Tesouro Direto via URL, tratamento inicial (Pandas) e exportação para o formato Parquet na camada Bronze, garantindo a primeira etapa da pipeline.

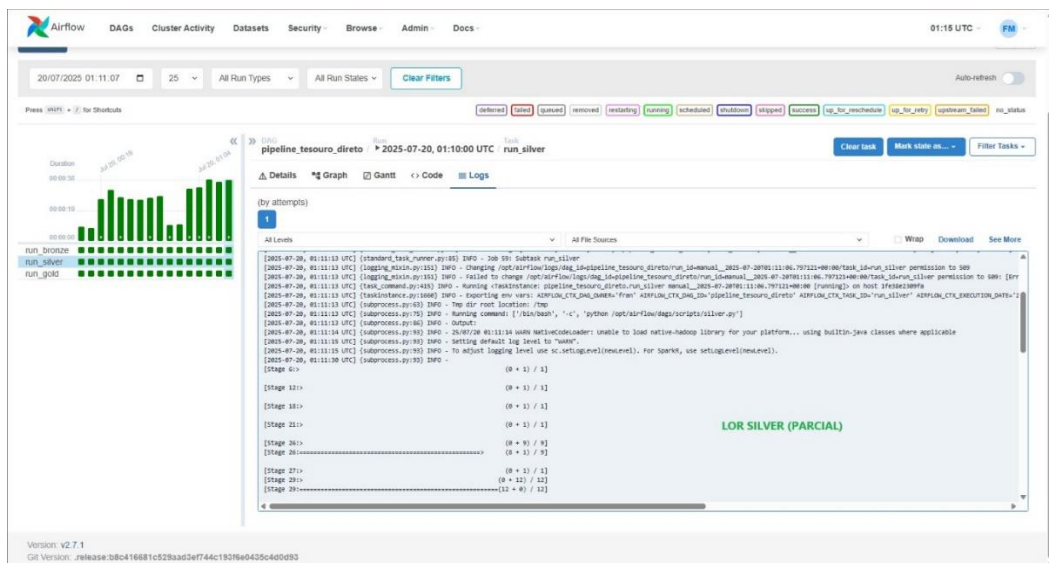
Imagem 16 – Airflow (Log Bronze)



Fonte: Própria autoria

Este log indica o início da execução do script silver.py. Esta camada foca na limpeza, padronização e transformações intermediárias dos dados, preparando-os para as agregações futuras na camada Gold.

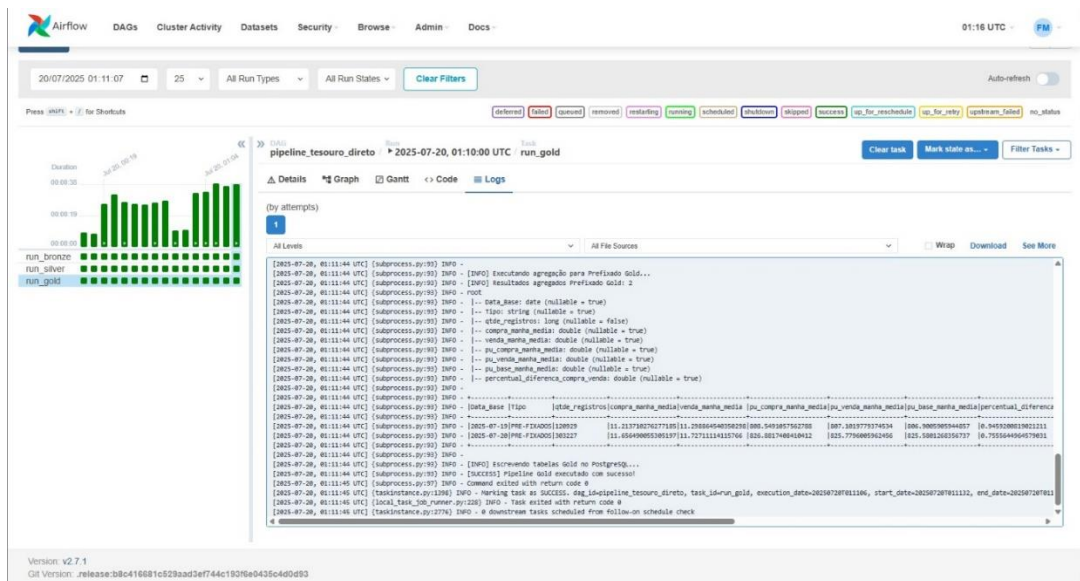
Imagem 17 – Airflow (Log Silver)



Fonte: Própria autoria

Detalha o log da task run_gold, que foi executada com sucesso. Esta camada realiza agregações complexas e cálculos de indicadores sobre os dados da camada Silver. O log mostra o esquema final das colunas e a exportação dos dados prontos para consumo para o S3.

Imagem 18 – Airflow (Log Gold)



Fonte: Própria autoria

Códigos e informações complementares

Bronze.py

```
import pandas as pd
from sqlalchemy import create_engine
from datetime import datetime, timedelta
```

```
def busca_titulos_tesouro_direto():
    """
```

Faz o download dos dados públicos de preços e taxas dos títulos do Tesouro Direto diretamente do site Tesouro Transparente, processa e retorna um DataFrame com índice multinível.

Returns:

pd.DataFrame: DataFrame contendo os dados dos títulos do Tesouro Direto, com índice multinível (Tipo de Título, Data de Vencimento, Data Base).

```

"""
print("Iniciando download dos títulos do Tesouro Direto...")
url = 'https://www.tesourotransparente.gov.br/ckan/dataset/df56aa42-484a-4a59-8184-7676580c81e3/resource/796d2059-14e9-44e3-80c9-2d9e30b405c1/download/PrecoTaxaTesouroDireto.csv'
df = pd.read_csv(url, sep=';', decimal=',')
df['Data Vencimento'] = pd.to_datetime(df['Data Vencimento'],
dayfirst=True)
df['Data Base'] = pd.to_datetime(df['Data Base'], dayfirst=True)
multi_indice = pd.MultiIndex.from_frame(df.iloc[:, :3])
df = df.set_index(multi_indice).iloc[:, 3:]
print("Download concluído com sucesso.")
return df

def gerar_timestamps_simulados(df, start_time=None, interval_ms=100):
    """
    Adiciona uma coluna de timestamps simulados ao DataFrame, com intervalo
    definido entre as linhas.

    Args:
        df (pd.DataFrame): DataFrame de entrada ao qual serão adicionados os
        timestamps.
        start_time (datetime, optional): Horário inicial para os timestamps.
        Se None, usa uma hora antes do tempo
        atual.
        interval_ms (int, optional): Intervalo entre os timestamps em
        milissegundos. Padrão é 100ms.

    Returns:
        pd.DataFrame: DataFrame com uma nova coluna 'dt_update' contendo os
        timestamps simulados.
    """
    if start_time is None:
        start_time = datetime.now() - timedelta(hours=1)
    timestamps = [start_time + timedelta(milliseconds=i * interval_ms) for i
in range(len(df))]
    df = df.copy()
    df['dt_update'] = timestamps
    return df

def run():
    """
    Executa o pipeline de ingestão de dados (camada Bronze) dos títulos do
    Tesouro Direto.

    Realiza:
        - Download dos dados do Tesouro.
        - Filtragem dos títulos do tipo IPCA+ e Prefixado.
        - Adição de timestamps simulados.
        - Renomeação de colunas.
        - Exportação para tabelas PostgreSQL específicas por tipo de título.
    """

```



```

print("Iniciando o pipeline Bronze dos títulos do Tesouro Direto...")

titulos = busca_titulos_tesouro_direto()

# Cria coluna "Tipo"
titulos.loc[titulos.index.get_level_values(0) == 'Tesouro Prefixado',
'Tipo'] = "PRE-FIXADOS"
titulos.loc[titulos.index.get_level_values(0) == 'Tesouro IPCA+', 'Tipo']
= "IPCA"

print("Filtrando e processando títulos IPCA e Prefixado...")
ipca = titulos.loc[('Tesouro IPCA+').copy()]
prefixado = titulos.loc[('Tesouro Prefixado')].copy()

# Gera timestamps simulados com intervalo de 100ms
ipca = gerar_timestamps_simulados(ipca, interval_ms=100)
prefixado = gerar_timestamps_simulados(prefixado, interval_ms=100)

rename_cols = {
    "Taxa Compra Manhã": "CompraManha",
    "Taxa Venda Manhã": "VendaManha",
    "PU Compra Manhã": "PUCompraManha",
    "PU Venda Manhã": "PUVendaManha",
    "PU Base Manhã": "PUBaseManha",
    "Data Vencimento": "DataVencimento",
    "Data Base": "Data_Base"
}

ipca = ipca.rename(columns=rename_cols)
prefixado = prefixado.rename(columns=rename_cols)

print("Criando engine de conexão com PostgreSQL...")
connection_string =
"postgresql://postgres:postgres@postgres:5432/postgres"
engine = create_engine(connection_string)

print("Exportando dados IPCA para a tabela 'dadostesouroipca'...")
ipca.to_sql("dadostesouroipca", con=engine, if_exists="append",
index=False)

print("Exportando dados Prefixado para a tabela 'dadostesouropre'...")
prefixado.to_sql("dadostesouropre", con=engine, if_exists="append",
index=False)

print("Pipeline Bronze executado com sucesso!")

if __name__ == "__main__":

```

```
run()
```

Silver.py

```
import os
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, upper, to_date
from dotenv import load_dotenv

def main():
    """
    Executa o pipeline de processamento da camada Silver usando PySpark.

    Objetivo:
        - Limpar, padronizar e enriquecer os dados da camada Bronze
        (`dadostesouroipca` e `dadostesouopre`)
        antes de disponibilizá-los para análises e agregações na camada
        Gold.

    Etapas:
        1. Carrega variáveis de ambiente (.env) com credenciais AWS e PostgreSQL.
        2. Inicializa uma sessão Spark configurada para acessar dados S3 e
        PostgreSQL.
        3. Lê as tabelas Bronze (`dadostesouroipca` e `dadostesouopre`) do
        PostgreSQL.
        4. Remove registros duplicados e linhas com `null` nas colunas essenciais.
        5. Converte a coluna `dt_update` em uma nova coluna `Data_Base` (apenas
        data).
        6. Padroniza a coluna `Tipo` para letras maiúsculas.
        7. Exibe esquema e amostras dos dados processados.
        8. Escreve os dados tratados nas tabelas Silver:
            - `dadostesouroipca_silver`
            - `dadostesouopre_silver`

    Pré-requisitos:
        - As bibliotecas JAR necessárias (JDBC, AWS SDK e Hadoop AWS) devem
        estar no diretório `/opt/spark/jars/`.
        - As variáveis de ambiente AWS e PostgreSQL devem estar definidas no
        arquivo `.env_kafka_connect`.
        - As tabelas Bronze já devem estar populadas no PostgreSQL.
    """
    print("[INFO] Carregando variáveis de ambiente...")
    load_dotenv("/opt/airflow/.env_kafka_connect")

    aws_access_key = os.getenv("AWS_ACCESS_KEY_ID")
    aws_secret_key = os.getenv("AWS_SECRET_ACCESS_KEY")
    aws_region = "us-east-1"

    pg_host = os.getenv("POSTGRES_HOST")
```

```

pg_port = os.getenv("POSTGRES_PORT", "5432")
pg_db = os.getenv("POSTGRES_DB")
pg_user = os.getenv("POSTGRES_USER")
pg_password = os.getenv("POSTGRES_PASSWORD")

jdbc_url = f"jdbc:postgresql://{pg_host}:{pg_port}/{pg_db}"
jdbc_properties = {
    "user": pg_user,
    "password": pg_password,
    "driver": "org.postgresql.Driver"
}

hadoop_aws_jar = "/opt/spark/jars/hadoop-aws-3.3.4.jar"
aws_sdk_jar = "/opt/spark/jars/aws-java-sdk-bundle-1.12.262.jar"
postgres_jdbc_jar = "/opt/spark/jars/postgresql-42.6.0.jar"
jars_path = f"{hadoop_aws_jar},{aws_sdk_jar},{postgres_jdbc_jar}"

print("[INFO] Inicializando SparkSession...")
spark = SparkSession.builder \
    .appName("Pipeline - Silver") \
    .config("spark.jars", jars_path) \
    .config("spark.hadoop.fs.s3a.impl",
"org.apache.hadoop.fs.s3a.S3AFileSystem") \
    .config("spark.hadoop.fs.s3a.access.key", aws_access_key) \
    .config("spark.hadoop.fs.s3a.secret.key", aws_secret_key) \
    .config("spark.hadoop.fs.s3a.endpoint",
f"s3.{aws_region}.amazonaws.com") \
    .config("spark.hadoop.fs.s3a.connection.ssl.enabled", "true") \
    .config("spark.hadoop.fs.s3a.path.style.access", "true") \
    .getOrCreate()

colunas_obrigatorias = [
    "CompraManha", "VendaManha", "PUCompraManha", "PUVendaManha",
"PUBaseManha", "Tipo", "dt_update"
]

print("[INFO] Lendo tabela dadostesouroipca do PostgreSQL...")
df_ipca = spark.read.jdbc(url=jdbc_url, table="public.dadostesouroipca",
properties=jdbc_properties)
print(f"[INFO] Registros lidos IPCA: {df_ipca.count()}")

print("[INFO] Lendo tabela dadostesouropre do PostgreSQL...")
df_pre = spark.read.jdbc(url=jdbc_url, table="public.dadostesouropre",
properties=jdbc_properties)
print(f"[INFO] Registros lidos Prefixado: {df_pre.count()}")

```

```

    print("[INFO] Removendo duplicatas e linhas com valores nulos nas colunas obrigatórias...")
    df_ipca = df_ipca.dropDuplicates().dropna(subset=colunas_obrigatorias)
    df_pre = df_pre.dropDuplicates().dropna(subset=colunas_obrigatorias)

    print(f"[INFO] Registros após limpeza IPCA: {df_ipca.count()}")
    print(f"[INFO] Registros após limpeza Prefixado: {df_pre.count()}")

    print("[INFO] Criando coluna 'Data_Base' a partir de 'dt_update' (apenas data)...")
    df_ipca = df_ipca.withColumn("Data_Base", to_date(col("dt_update")))
    df_pre = df_pre.withColumn("Data_Base", to_date(col("dt_update")))

    print("[INFO] Convertendo coluna 'Tipo' para maiúsculas...")
    df_ipca = df_ipca.withColumn("Tipo", upper(col("Tipo")))
    df_pre = df_pre.withColumn("Tipo", upper(col("Tipo")))

    print("[INFO] Exibindo esquema e amostra dos dados IPCA pós-processamento:")
    df_ipca.printSchema()
    df_ipca.show(5, truncate=False)

    print("[INFO] Exibindo esquema e amostra dos dados Prefixado pós-processamento:")
    df_pre.printSchema()
    df_pre.show(5, truncate=False)

    print("[INFO] Escrevendo dados processados para tabelas Silver no PostgreSQL...")
    df_ipca.write.jdbc(url=jdbc_url, table="public.dadostesouroipca_silver", mode="overwrite", properties=jdbc_properties)
    df_pre.write.jdbc(url=jdbc_url, table="public.dadostesouropre_silver", mode="overwrite", properties=jdbc_properties)

    print("[SUCCESS] Pipeline Silver executado com sucesso!")

if __name__ == "__main__":
    main()

```

Gold.py

```

from pyspark.sql import SparkSession
from dotenv import load_dotenv
import os

def main():
    """
    Executa o pipeline de transformação de dados da camada Silver para Gold usando PySpark.
    """

```

Etapas:

1. Carrega variáveis de ambiente do arquivo ``.env``.
2. Cria conexão JDBC com banco de dados PostgreSQL.
3. Lê dados das tabelas Silver (``dadostesouroipca_silver`` e ``dadostesouopre_silver``).
4. Cria views temporárias no Spark para processamento com SQL.
5. Realiza agregações:
 - Calcula média de taxas e preços unitários.
 - Calcula percentual de diferença entre taxa de venda e de compra.
 - Agrupa por ``Data_Base`` e ``Tipo``.
6. Escreve os dados transformados em tabelas Gold (``dadostesouroipca_gold`` e ``dadostesouopre_gold``).

Requisitos:

- O driver JDBC do PostgreSQL deve estar disponível no caminho ``/opt/spark/jars/postgresql-42.6.0.jar``.
- As variáveis de ambiente com configurações do PostgreSQL devem estar no arquivo ``/opt/airflow/.env_kafka_connect``.

Banco de destino:

PostgreSQL com as tabelas:

- public.dadostesouroipca_gold
- public.dadostesouopre_gold

"""

```
print("[INFO] Carregando variáveis de ambiente...")
load_dotenv("/opt/airflow/.env_kafka_connect")
```

```
pg_host = os.getenv("POSTGRES_HOST")
pg_port = os.getenv("POSTGRES_PORT", "5432")
pg_db = os.getenv("POSTGRES_DB")
pg_user = os.getenv("POSTGRES_USER")
pg_password = os.getenv("POSTGRES_PASSWORD")
```

```
jdbc_url = f"jdbc:postgresql://{pg_host}:{pg_port}/{pg_db}"
jdbc_properties = {
    "user": pg_user,
    "password": pg_password,
    "driver": "org.postgresql.Driver"
}
```

```
print("[INFO] Inicializando SparkSession...")
spark = SparkSession.builder \
    .appName("Pipeline - Gold") \
    .config("spark.jars", "/opt/spark/jars/postgresql-42.6.0.jar") \
    .getOrCreate()
```



```

print("[INFO] Lendo tabelas Silver do PostgreSQL...")
df_ipca = spark.read.jdbc(
    url=jdbc_url,
    table="public.dadostesouroipca_silver",
    properties=jdbc_properties
)
print(f"[INFO] Registros IPCA Silver: {df_ipca.count()}")
df_ipca.printSchema()
df_ipca.show(5, truncate=False)


df_pre = spark.read.jdbc(
    url=jdbc_url,
    table="public.dadostesouropre_silver",
    properties=jdbc_properties
)
print(f"[INFO] Registros Prefixado Silver: {df_pre.count()}")
df_pre.printSchema()
df_pre.show(5, truncate=False)

print("[INFO] Registrando views temporárias...")
df_ipca.createOrReplaceTempView("ipca_silver")
df_pre.createOrReplaceTempView("pre_silver")

print("[INFO] Executando agregação para IPCA Gold...")
df_ipca_gold = spark.sql("""
    SELECT
        Data_Base,
        Tipo,
        COUNT(*) AS qtde_registros,
        AVG(CompraManha) AS compra_manha_media,
        AVG(VendaManha) AS venda_manha_media,
        AVG(PUCompraManha) AS pu_compra_manha_media,
        AVG(PUVendaManha) AS pu_venda_manha_media,
        AVG(PUBaseManha) AS pu_base_manha_media,
        AVG(100 * (VendaManha - CompraManha) / CompraManha) AS
percentual_diferenca_compra_venda
    FROM ipca_silver
    GROUP BY Data_Base, Tipo
    ORDER BY Data_Base, Tipo
""")
print(f"[INFO] Resultados agregados IPCA Gold: {df_ipca_gold.count()}")
df_ipca_gold.printSchema()
df_ipca_gold.show(5, truncate=False)

print("[INFO] Executando agregação para Prefixado Gold...")
df_pre_gold = spark.sql("""
    SELECT
        Data_Base,
        Tipo,
        COUNT(*) AS qtde_registros,
        AVG(CompraManha) AS compra_manha_media,

```



```

        AVG(VendaManha) AS venda_manha_media,
        AVG(PUCompraManha) AS pu_compra_manha_media,
        AVG(PUVendaManha) AS pu_venda_manha_media,
        AVG(PUBaseManha) AS pu_base_manha_media,
        AVG(100 * (VendaManha - CompraManha) / CompraManha) AS
percentual_diferenca_compra_venda
    FROM pre_silver
    GROUP BY Data_Base, Tipo
    ORDER BY Data_Base, Tipo
    """)
    print(f"[INFO] Resultados agregados Prefixado Gold:
{df_pre_gold.count()}")
    df_pre_gold.printSchema()
    df_pre_gold.show(5, truncate=False)

    print("[INFO] Escrevendo tabelas Gold no PostgreSQL...")
    df_ipca_gold.write.jdbc(
        url=jdbc_url,
        table="public.dadostesouroipca_gold",
        mode="overwrite",
        properties=jdbc_properties
    )
    df_pre_gold.write.jdbc(
        url=jdbc_url,
        table="public.dadostesouropre_gold",
        mode="overwrite",
        properties=jdbc_properties
    )

    print("[SUCCESS] Pipeline Gold executado com sucesso!")

if __name__ == "__main__":
    main()

```

Github

https://github.com/FranMuniz/bootcamp_eng_dados_projeto_final

