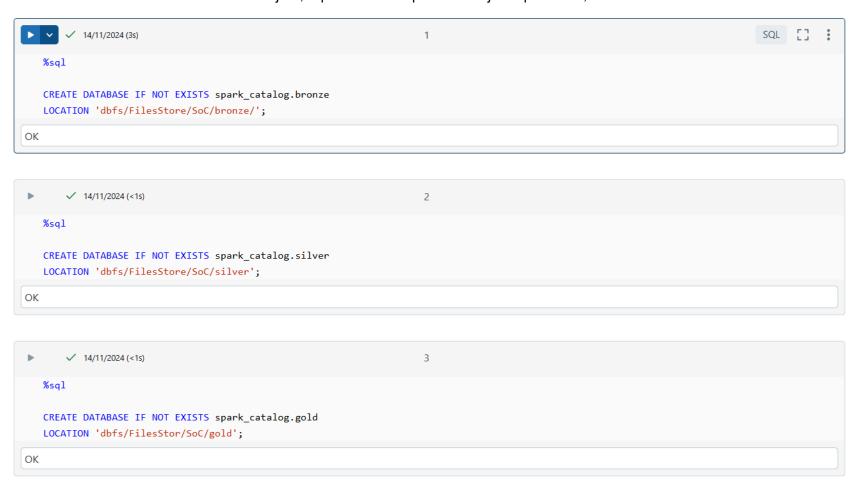
Primeiramente iniciamos com a criação de um notebook para configurarmos a criação do Database. Aqui já definimos os 3 bancos de dados necessários na arquitetura medalhão.

A arquitetura medalhão tem um papel fundamental para um engenheiro de dados. Com elas, todo o projeto fica organizado e sabe-se como os dados estão em cada camada para futuras manipulações e consultas conforme a estratégia de negócio.

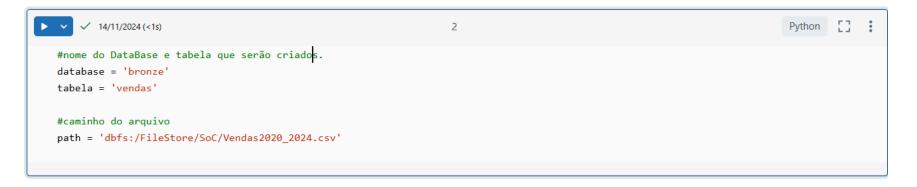
Na camada BRONZE temos os dados "sujos", aqueles dados que vem do jeito que estão, sem tratamento



Na camada BRONZE temos os dados que chamamos "crus", "originais", aqueles dados que vem do jeito que estão, sem tratamento algum, necessitando de assim fazê-los.

```
#Aqui começamos a importar as bibliotecas que serão usadas na camada bronze do projeto.

from pyspark.sql.functions \
    import \
        current_date, \
        current_timestamp, \
        expr, \
        col, \
        split, \
        hour, \
        to_date, \
        substring, \
        trim
```



Aqui já podemos iniciar a padronização dos nomes das colunas da tabela, e já temos previamente o s tipos de dados que o parâmetro ".option("inferSchema", True)" reconhece no DataFrame.

```
▶ ✓ √ 14/11/2024 (<1s)
                                                                                                                               Python []
  df = df.withColumnRenamed("Código", "ProdutoID")
  df = df.withColumnRenamed("Descrição", "Descrição")
  df = df.withColumnRenamed("Valor Unitário", "Valor_Unitario")
  df = df.withColumnRenamed("Valor Total", "Valor_Total")
  df = df.withColumnRenamed("Nota Fiscal", "Nota_Fiscal")
  df = df.withColumnRenamed("Emissão", "Emissao")
  df = df.withColumnRenamed("Unnamed: 8", "Unnamed")
▼ ■ df: pyspark.sql.dataframe.DataFrame
       ProdutoID: string
       Descricao: string
       Qtd: integer
       Valor_Unitario: double
       Valor_Total: double
       Nota Fiscal: integer
       Emissao: string
       Vendedor: integer
```

Aqui separamos alguns dados que vieram unificados numa mesma coluna, como no caso 'Data' e 'Hora'. Foi utilizado linhas de código Python para realizar essa ação. Abaixo pode ser observada cada parte do código.

```
14/11/2024 (<1s)</p>
                                                                        5
  # Separando a data e a hora da coluna "Emissao"
  df = df.withColumn("Data_Emissao", trim(split(col("Emissao"), " ")[0])) \
         .withColumn("Hora_Emissao", trim(split(col("Emissao"), " ")[1]))
  # Convertendo a coluna "Data_Emissao" para o tipo Date (com formato dd-MM-yyyy)
  df = df.withColumn("Data_Emissao", to_date(col("Data_Emissao"), 'dd/MM/yyyy'))
  # Removendo a coluna antiga "Emissao"
  df = df.drop("Emissao")
▶ ■ df: pyspark.sql.dataframe.DataFrame = [ProdutoID: string, Descricao: string ... mais 7 campos]
```

```
#Aqui é atribuído as funções que capturam a data da carga e data e hora da carga para um melhor controle de versões e quando foi feito tal processamento.

df = df.withColumn("data_carga", current_date())
df = df.withColumn("data_hora_carga", expr("current_timestamp() - INTERVAL 3 HOURS"))

df: pyspark.sql.dataframe.DataFrame = [ProdutoID: string, Descricao: string ... mais 9 campos]
```