**Dataside**

Documentação Técnica - Automação de Notebooks  
  
Documento de Implementação Técnica

# Objetivo do documento

Este Documento de Implementação Técnica (DIT) tem como objetivo registrar, de forma detalhada, padronizada eprofissional, as soluções técnicas desenvolvidas no escopo do projeto, com ênfase na automação e execução de scriptscontidos em notebooks.O documento visa garantir rastreabilidade, compreensão técnica, reprodutibilidade das implementações e alinhamento com asmelhores práticas de desenvolvimento e documentação adotadas pela Dataside. Além disso, busca facilitar a comunicaçãoentre equipes técnicas e não técnicas, assegurando que o conhecimento gerado esteja devidamente estruturado e acessívelpara auditorias, manutenções futuras e expansão do projeto.

# Sumário

1. 1. Empresa Ficticia Anonimizado
2. 2. Empresa Ficticia Anonimizado Completo

# Introdução

A introdução do Documento de Implementação Técnica (DIT) tem como objetivo apresentar de forma sucinta o escopo do projeto em questão, bem como o propósito deste documento. Os notebooks listados, "dim\_empresa\_ficticia\_anonimizado.md" e "dim\_empresa\_ficticia\_anonimizado\_completo.md", desempenham um papel fundamental no processo de ingestão, tratamento e análise de dados relacionados à empresa fictícia em questão.  
  
O notebook "dim\_empresa\_ficticia\_anonimizado.md" é responsável por realizar a anonimização dos dados da empresa, garantindo a privacidade e segurança das informações. Já o notebook "dim\_empresa\_ficticia\_anonimizado\_completo.md" complementa o processo, realizando a criação da dimensão completa da empresa, com dados tratados e prontos para análise.  
  
Este documento oferece à equipe técnica uma visão clara e estruturada do processo de implementação técnica, facilitando a compreensão e execução das etapas necessárias. Além disso, contribui para o sucesso do projeto, fornecendo um guia detalhado e padronizado para a equipe, resultando em maior eficiência e qualidade na entrega final.

# 1. Dim\_Empresa\_Ficticia\_Anonimizado

Tabela da dimensão de empresas do indicador farol.

### 1.1 Resumo do Código

O trecho de código apresentado no notebook "Dim\_Empresa\_Ficticia\_Anonimizado" realiza a execução da função de ingestão de dados, a qual é responsável por carregar e processar informações relevantes para a empresa fictícia de forma anônima. Essa função desencadeia o processo de coleta e preparação dos dados necessários para alimentar a dimensão relacionada à empresa em questão, contribuindo assim para a construção de um ambiente de análise e tomada de decisões mais robusto e eficiente.

#### Código Fonte

%run ../00\_config/ingestion\_function

### 1.2 Resumo do Código

O trecho de código Python apresentado tem como objetivo definir variáveis e gerar informações relacionadas a uma tabela de dimensão de empresas fictícias. São atribuídos valores às variáveis container\_target, directory, table\_name e delta\_table\_name, que compõem o caminho e o nome da tabela no ambiente Delta Lake. Além disso, é gerada a URL do arquivo Delta correspondente, utilizando informações fictícias de diretório e nome do datalake. Por fim, são exibidos na saída padrão os valores das variáveis delta\_table\_name e delta\_file. Este código é útil para configurar e visualizar informações essenciais para o processamento de dados relacionados à dimensão de empresas fictícias.

#### Código Fonte

debug = False  
  
container\_target = 'camada\_destino'  
directory = 'diretorio\_ficticio'  
table\_name = 'dim\_empresa\_ficticia'  
delta\_table\_name = f'{environment}.{container\_target}.{table\_name}'  
delta\_file = f"abfss://{container\_target}@{nome\_datalake\_ficticio}.dfs.core.windows.net/{directory}/{table\_name}/"  
comment\_delta\_table = 'Tabela de dimensão de empresas fictícias.'  
  
print(f'delta\_table\_name = {table\_name}')  
print(f'delta\_file = {delta\_file}')

### 1.3 Resumo do Código

O trecho de código apresentado no notebook "Dim\_Empresa\_Ficticia\_Anonimizado" consiste na criação de um widget interativo denominado `reprocessar`, que permite ao usuário controlar a carga da dimensão. Este widget exibe um dropdown com as opções "True" e "False", sendo que a variável `reprocessar` será atribuída o valor `True` somente se o usuário selecionar essa opção no notebook. O objetivo principal é possibilitar que o usuário escolha de forma interativa se a carga da dimensão será completa (`overwrite`) ou incremental (`merge`). Essa abordagem visa oferecer flexibilidade e controle ao usuário durante o processo de carga de dados na dimensão.

#### Código Fonte

"""  
Cria um widget interativo chamado `reprocessar` para controle da carga da dimensão.  
  
- Exibe um dropdown com as opções "True" e "False".  
- A variável `reprocessar` será `True` apenas se o usuário selecionar essa opção no notebook.  
  
Objetivo: permitir que o usuário escolha, de forma interativa, se a carga será completa (`overwrite`) ou incremental (`merge`).  
"""  
  
dbutils.widgets.dropdown("reprocessar", "False", ["True", "False"], "Reprocessar dimensão?")  
reprocessar = dbutils.widgets.get("reprocessar") == "True"

### 1.4 Resumo do Código

O trecho de código Python apresentado é responsável por criar uma tabela no ambiente Spark, caso ela ainda não exista, com uma série de colunas que representam informações de uma empresa fictícia anonimizada. Essas colunas incluem dados como código da empresa, CNPJ, razão social, nome da empresa, endereço, dados de contato, informações sobre funcionamento, entre outros. A tabela é criada utilizando o formato Delta e é especificada a localização onde os dados serão armazenados, juntamente com um comentário que descreve a finalidade da tabela. Além disso, são definidos campos de data de inserção e atualização para controle de versões dos dados. Este código é essencial para a estruturação e organização dos dados da empresa fictícia no ambiente de processamento distribuído do Spark.

#### Código Fonte

spark.sql(f"""  
CREATE TABLE IF NOT EXISTS {delta\_table\_name} (  
 sk\_empresas BIGINT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),  
 codigo\_empresa INT,  
 cnpj\_empresa STRING,  
 razao\_social STRING,  
 nome\_empresa STRING,  
 des\_logradouro STRING,  
 num\_insc\_estadual STRING,  
 num\_insc\_municipal STRING,  
 des\_complemento STRING,  
 num\_cep STRING,  
 nom\_bairro STRING,  
 cod\_cidade INT,  
 area STRING,  
 gerente\_area STRING,  
 porte STRING,  
 data\_abertura\_cupom DATE,  
 data\_abertura\_empresa STRING,  
 dias\_funcionamento STRING,  
 horario\_funcionamento STRING,  
 categoria STRING,  
 possui\_gnv STRING,  
 possui\_etanol STRING,  
 possui\_supervisor STRING,  
 quadro\_aprovado STRING,  
 pista\_diesel STRING,  
 troca\_oleo STRING,  
 sss STRING,  
 bandeira STRING,  
 e\_mail STRING,  
 e\_mail\_ga STRING,  
 m2\_loja STRING,  
 status STRING,  
 data\_inativacao STRING,  
 insert\_date TIMESTAMP,  
 update\_date TIMESTAMP  
)   
USING DELTA  
LOCATION '{delta\_file}'  
COMMENT '{comment\_delta\_table}';  
""")

### 1.5 Resumo do Código

O trecho de código Python apresentado realiza uma consulta em uma tabela específica, selecionando diversas colunas relacionadas a informações de empresas fictícias anonimizadas. Os dados extraídos incluem o código da empresa, CNPJ, nome fantasia, razão social, endereço completo, inscrições estadual e municipal, complemento, CEP, bairro e código da cidade. Essa consulta é executada no ambiente especificado e os resultados são temporariamente armazenados em uma visualização chamada 'tab\_empresa' para posterior análise e manipulação dentro do ambiente de processamento de dados.

#### Código Fonte

spark.sql(f"""  
  
select  
 cod\_empresa  
 ,num\_cnpj  
 ,nom\_fantasia  
 ,nom\_razao\_social  
 ,des\_logradouro  
 ,num\_insc\_estadual  
 ,num\_insc\_municipal  
 ,des\_complemento  
 ,num\_cep  
 ,nom\_bairro  
 ,cod\_cidade  
from {environment}.trusted.tb\_tab\_empresa\_farol  
where 1=1  
""").createOrReplaceTempView('tab\_empresa')

### 1.6 Resumo do Código

O trecho de código apresentado realiza uma consulta em uma tabela do ambiente de produção, selecionando diversas colunas relacionadas a dados de filiais de uma empresa fictícia. Essas colunas incluem informações como código da filial, área, porte, metragem da loja, data de abertura, dias e horário de funcionamento, entre outros. A consulta é feita a partir da tabela "sharepoint\_dados\_filiais" e os resultados são temporariamente armazenados em uma visualização chamada "dados\_filiais". Este processo é essencial para a análise e manipulação posterior desses dados no ambiente de processamento distribuído do Spark.

#### Código Fonte

spark.sql(f"""  
  
select  
 codigo  
 ,area  
 ,porte  
 ,m2\_loja  
 ,area  
 ,ga   
 ,porte   
 ,data\_abertura  
 ,dias\_de\_func   
 ,horario\_de\_func   
 ,categoria  
 ,possui\_gnv  
 ,possui\_etanol  
 ,possui\_supervisor  
 ,quadro\_aprovado  
 ,pista\_diesel  
 ,troca\_oleo  
 ,sss  
 ,bandeira  
 ,e\_mail  
 ,e\_mail\_ga  
 ,m2\_loja  
 ,status  
 ,data\_inativacao  
from {environment}.raw.sharepoint\_dados\_filiais  
where 1=1;  
  
""").createOrReplaceTempView('dados\_filiais')

### 1.7 Resumo do Código

O trecho de código Python apresentado tem como finalidade extrair informações relacionadas ao código da empresa e à data de abertura de filiais a partir da tabela "tb\_farol\_faturamento". Utilizando a função SQL do Spark, o código realiza a seleção do código da empresa e a data mínima de abertura da filial, agrupando os resultados por código da empresa. Posteriormente, os resultados são armazenados temporariamente na visualização "tab\_faturamento" para consultas posteriores. Este processo é essencial para a análise e acompanhamento da evolução das filiais da empresa fictícia em questão.

#### Código Fonte

spark.sql(f"""  
  
select  
 cod\_empresa  
 ,to\_date(min(data), 'yyyy-MM-dd') as data\_abertura\_filial  
from {environment}.refined.tb\_farol\_faturamento  
where 1=1  
group by  
 cod\_empresa;  
  
""").createOrReplaceTempView('tab\_faturamento')

### 1.8 Resumo do Código

O trecho de código Python apresentado tem como objetivo realizar uma consulta em um ambiente de processamento distribuído utilizando o Spark SQL. A consulta seleciona diversas colunas de diferentes tabelas, como código da empresa, CNPJ, razão social, entre outros, e realiza um join entre as tabelas "tab\_empresa", "dados\_filiais" e "tab\_faturamento". Após a execução da consulta, é criada uma visualização temporária chamada 'source\_df'. Este processo visa integrar e estruturar informações relevantes sobre empresas e filiais para análises posteriores no ambiente de Big Data.

#### Código Fonte

source\_df = spark.sql("""  
   
select   
 emp.cod\_empresa as codigo\_empresa  
 ,emp.num\_cnpj as cnpj\_empresa  
 ,emp.nom\_razao\_social as razao\_social  
 ,emp.nom\_fantasia as nome\_empresa  
 ,emp.des\_logradouro  
 ,emp.num\_insc\_estadual  
 ,emp.num\_insc\_municipal  
 ,emp.des\_complemento  
 ,emp.num\_cep  
 ,emp.nom\_bairro  
 ,emp.cod\_cidade  
 ,fil.area  
 ,fil.ga as gerente\_area  
 ,fil.porte as porte  
 ,fat.data\_abertura\_filial as data\_abertura\_cupom  
 ,fil.data\_abertura as data\_abertura\_empresa  
 ,fil.dias\_de\_func as dias\_funcionamento  
 ,fil.horario\_de\_func as horario\_funcionamento  
 ,fil.categoria  
 ,fil.possui\_gnv  
 ,fil.possui\_etanol  
 ,fil.possui\_supervisor  
 ,fil.quadro\_aprovado  
 ,fil.pista\_diesel  
 ,fil.troca\_oleo  
 ,fil.sss  
 ,fil.bandeira  
 ,fil.e\_mail  
 ,fil.e\_mail\_ga  
 ,fil.m2\_loja  
 ,fil.status  
 ,fil.data\_inativacao  
from tab\_empresa emp  
left join dados\_filiais fil  
 on emp.cod\_empresa = fil.codigo  
left join tab\_faturamento fat  
 on emp.cod\_empresa = fat.cod\_empresa  
where 1=1  
order by emp.nom\_fantasia  
""")  
  
source\_df.createOrReplaceTempView('source\_df')

### 1.9 Resumo do Código

O trecho de código Python apresentado no notebook "Dim\_Empresa\_Ficticia\_Anonimizado" tem como objetivo realizar a carga de dados em uma tabela Delta, utilizando a lógica de reprocessamento ou merge. Caso a variável "reprocessar" seja verdadeira, os dados são inseridos na tabela Delta sobrescrevendo os dados existentes. Caso contrário, é realizada uma operação de merge, atualizando os registros existentes com base na correspondência de chaves e inserindo novos registros caso não haja correspondência. O código realiza a atualização de diversas colunas da tabela alvo com os valores correspondentes da tabela de origem, garantindo a integridade e consistência dos dados. Ao final, é exibida uma mensagem indicando o sucesso da carga ou informando que está sendo realizada a carga em modo merge.

#### Código Fonte

if reprocessar:  
 source\_df = source\_df.withColumn("insert\_date", lit(current\_timestamp() ) )  
 source\_df.write.format("delta")\  
 .mode("overwrite")\  
 .saveAsTable(f"{delta\_table\_name}", path=f"{delta\_file}")  
 print(f"Carga overwrite realizada com sucesso! {delta\_table\_name}")  
else:  
 print("Realizando carga em Merge..")  
 spark.sql(f"""  
  
MERGE INTO {delta\_table\_name} AS target  
USING source\_df AS source  
 ON target.codigo\_empresa = source.codigo\_empresa  
WHEN MATCHED THEN  
 UPDATE SET  
 target.cnpj\_empresa = source.cnpj\_empresa,  
 target.razao\_social = source.razao\_social,  
 target.nome\_empresa = source.nome\_empresa,  
 target.des\_logradouro = source.des\_logradouro,  
 target.num\_insc\_estadual = source.num\_insc\_estadual,  
 target.num\_insc\_municipal = source.num\_insc\_municipal,  
 target.des\_complemento = source.des\_complemento,  
 target.num\_cep = source.num\_cep,  
 target.nom\_bairro = source.nom\_bairro,  
 target.cod\_cidade = source.cod\_cidade,  
 target.area = source.area,  
 target.gerente\_area = source.gerente\_area,  
 target.porte = source.porte,  
 target.data\_abertura\_cupom = source.data\_abertura\_cupom,  
 target.data\_abertura\_empresa = source.data\_abertura\_empresa,  
 target.dias\_funcionamento = source.dias\_funcionamento,  
 target.horario\_funcionamento = source.horario\_funcionamento,  
 target.categoria = source.categoria,  
 target.possui\_gnv = source.possui\_gnv,  
 target.possui\_etanol = source.possui\_etanol,  
 target.possui\_supervisor = source.possui\_supervisor,  
 target.quadro\_aprovado = source.quadro\_aprovado,  
 target.pista\_diesel = source.pista\_diesel,  
 target.troca\_oleo = source.troca\_oleo,  
 target.sss = source.sss,  
 target.bandeira = source.bandeira,  
 target.e\_mail = source.e\_mail,  
 target.e\_mail\_ga = source.e\_mail\_ga,  
 target.m2\_loja = source.m2\_loja,  
 target.status = source.status,  
 target.data\_inativacao = source.data\_inativacao,  
 target.update\_date = current\_timestamp()  
WHEN NOT MATCHED THEN  
 INSERT (  
 codigo\_empresa,  
 cnpj\_empresa,  
 razao\_social,  
 nome\_empresa,  
 des\_logradouro,  
 num\_insc\_estadual,  
 num\_insc\_municipal,  
 des\_complemento,  
 num\_cep,  
 nom\_bairro,  
 cod\_cidade,  
 area,  
 gerente\_area,  
 porte,  
 data\_abertura\_cupom,  
 data\_abertura\_empresa,  
 dias\_funcionamento,  
 horario\_funcionamento,  
 categoria,  
 possui\_gnv,  
 possui\_etanol,  
 possui\_supervisor,  
 quadro\_aprovado,  
 pista\_diesel,  
 troca\_oleo,  
 sss,  
 bandeira,  
 e\_mail,  
 e\_mail\_ga,  
 m2\_loja,  
 status,  
 data\_inativacao,  
 insert\_date  
 )  
 VALUES (  
 source.codigo\_empresa,  
 source.cnpj\_empresa,  
 source.razao\_social,  
 source.nome\_empresa,  
 source.des\_logradouro,  
 source.num\_insc\_estadual,  
 source.num\_insc\_municipal,  
 source.des\_complemento,  
 source.num\_cep,  
 source.nom\_bairro,  
 source.cod\_cidade,  
 source.area,  
 source.gerente\_area,  
 source.porte,  
 source.data\_abertura\_cupom,  
 source.data\_abertura\_empresa,  
 source.dias\_funcionamento,  
 source.horario\_funcionamento,  
 source.categoria,  
 source.possui\_gnv,  
 source.possui\_etanol,  
 source.possui\_supervisor,  
 source.quadro\_aprovado,  
 source.pista\_diesel,  
 source.troca\_oleo,  
 source.sss,  
 source.bandeira,  
 source.e\_mail,  
 source.e\_mail\_ga,  
 source.m2\_loja,  
 source.status,  
 source.data\_inativacao,  
 current\_timestamp()  
 )  
   
 """)

# 2. Dim\_Empresa\_Ficticia\_Anonimizado\_Completo

Tabela da dimensão de empresas do indicador farol.

### 2.1 Resumo do Código

O trecho de código apresentado no notebook "Dim\_Empresa\_Ficticia\_Anonimizado\_Completo" realiza a execução da função de ingestão de dados, que está localizada no diretório de configuração "00\_config". Essa função é responsável por realizar a carga de dados de uma fonte externa para o ambiente de trabalho, possibilitando a utilização dessas informações em análises posteriores. A execução desse código é fundamental para garantir a disponibilidade e integridade dos dados necessários para as atividades de análise e processamento dentro do contexto da empresa fictícia em questão.

#### Código Fonte

%run ../00\_config/ingestion\_function

### 2.2 Resumo do Código

O trecho de código apresentado tem como objetivo definir variáveis e configurar parâmetros para a criação de uma tabela de dimensão de empresas fictícias em um ambiente de data lake. São atribuídos valores para o nome da tabela, o caminho do arquivo delta e o comentário associado à tabela. Além disso, é possível habilitar ou desabilitar o modo de depuração por meio da variável "debug". Ao final, são exibidos na saída os valores da variável "delta\_table\_name" e do caminho do arquivo delta.

#### Código Fonte

debug = False  
  
container\_target = 'camada\_destino'  
directory = 'diretorio\_ficticio'  
table\_name = 'dim\_empresa\_ficticia'  
delta\_table\_name = f'{environment}.{container\_target}.{table\_name}'  
delta\_file = f"abfss://{container\_target}@{nome\_datalake\_ficticio}.dfs.core.windows.net/{directory}/{table\_name}/"  
comment\_delta\_table = 'Tabela de dimensão de empresas fictícias.'  
  
print(f'delta\_table\_name = {table\_name}')  
print(f'delta\_file = {delta\_file}')

### 2.3 Resumo do Código

O trecho de código apresentado cria um widget interativo denominado `reprocessar` para controlar a carga da dimensão. Este widget exibe um menu suspenso com as opções "True" e "False", sendo que a variável `reprocessar` será definida como `True` somente se o usuário selecionar essa opção no notebook. O objetivo dessa implementação é permitir que o usuário escolha, de maneira interativa, se a carga da dimensão será completa (`overwrite`) ou incremental (`merge`). Essa funcionalidade proporciona flexibilidade ao usuário ao determinar o tipo de carga a ser realizada, de acordo com as necessidades do processo de ETL.

#### Código Fonte

"""  
Cria um widget interativo chamado `reprocessar` para controle da carga da dimensão.  
  
- Exibe um dropdown com as opções "True" e "False".  
- A variável `reprocessar` será `True` apenas se o usuário selecionar essa opção no notebook.  
  
Objetivo: permitir que o usuário escolha, de forma interativa, se a carga será completa (`overwrite`) ou incremental (`merge`).  
"""  
  
dbutils.widgets.dropdown("reprocessar", "False", ["True", "False"], "Reprocessar dimensão?")  
reprocessar = dbutils.widgets.get("reprocessar") == "True"

### 2.4 Resumo do Código

O trecho de código apresentado refere-se à criação de uma tabela no ambiente Spark utilizando o formato Delta. Nesta tabela, são definidos diversos campos relacionados a informações de empresas fictícias anonimizadas, como código, nome, endereço, dados de contato, segmento de atuação, entre outros. Além disso, são especificados campos para controle de datas de abertura, encerramento, inserção e atualização dos registros. A tabela é criada no local especificado, com um comentário adicional para documentação. Este código é essencial para a estruturação e organização dos dados relacionados às empresas fictícias no ambiente de processamento Spark.

#### Código Fonte

spark.sql(f"""  
CREATE TABLE IF NOT EXISTS {delta\_table\_name} (  
 sk\_empresas BIGINT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),  
 codigo\_ficticio INT,  
 doc\_empresa STRING,  
 nome\_razao\_social STRING,  
 nome\_fantasia STRING,  
 logradouro STRING,  
 inscricao\_estadual STRING,  
 inscricao\_municipal STRING,  
 complemento\_endereco STRING,  
 codigo\_postal STRING,  
 bairro STRING,  
 codigo\_cidade INT,  
 regiao STRING,  
 responsavel\_regional STRING,  
 categoria\_porte STRING,  
 data\_abertura\_atividade DATE,  
 data\_abertura\_empresa STRING,  
 dias\_atividade STRING,  
 horario\_atividade STRING,  
 segmento STRING,  
 tem\_gnv STRING,  
 tem\_etanol STRING,  
 tem\_supervisor STRING,  
 qtd\_funcionarios STRING,  
 tem\_diesel STRING,  
 tem\_troca\_oleo STRING,  
 codigo\_externo STRING,  
 marca\_parceira STRING,  
 email\_contato STRING,  
 email\_gerente STRING,  
 tamanho\_loja STRING,  
 situacao STRING,  
 data\_encerramento STRING,  
 data\_insercao TIMESTAMP,  
 data\_atualizacao TIMESTAMP  
)   
USING DELTA  
LOCATION '{delta\_file}'  
COMMENT '{comment\_delta\_table}';  
""")

### 2.5 Resumo do Código

O trecho de código Python apresentado realiza uma consulta em uma tabela de empresas, selecionando diversas informações como código da empresa, CNPJ, nome fantasia, razão social, endereço, inscrições estadual e municipal, complemento de endereço, código postal, bairro e código da cidade. Esses dados são extraídos da tabela "tb\_tab\_empresa\_farol" do ambiente especificado, e são temporariamente armazenados em uma visualização chamada "tab\_empresa". A cláusula "where 1=1" indica que não há filtros adicionais sendo aplicados na consulta. Este código é essencial para a preparação e análise de dados relacionados às empresas de forma anônima e completa.

#### Código Fonte

spark.sql(f"""  
  
select  
 cod\_empresa  
 ,num\_cnpj  
 ,nom\_fantasia  
 ,nom\_razao\_social  
 ,logradouro  
 ,inscricao\_estadual  
 ,inscricao\_municipal  
 ,complemento\_endereco  
 ,codigo\_postal  
 ,bairro  
 ,codigo\_cidade  
from {environment}.trusted.tb\_tab\_empresa\_farol  
where 1=1  
""").createOrReplaceTempView('tab\_empresa')

### 2.6 Resumo do Código

O trecho de código Python apresentado realiza uma consulta em uma tabela de dados de filiais, selecionando diversas colunas como código, região, categoria de porte, entre outras. A consulta é feita no ambiente especificado e os resultados são temporariamente armazenados em uma visualização chamada 'dados\_filiais'. Essa operação permite a análise e manipulação dos dados das filiais da empresa de forma mais eficiente e organizada, facilitando a extração de informações relevantes para tomada de decisões estratégicas.

#### Código Fonte

spark.sql(f"""  
  
select  
 codigo  
 ,regiao  
 ,categoria\_porte  
 ,tamanho\_loja  
 ,regiao  
 ,ga   
 ,categoria\_porte   
 ,data\_abertura  
 ,dias\_de\_func   
 ,horario\_de\_func   
 ,segmento  
 ,tem\_gnv  
 ,tem\_etanol  
 ,tem\_supervisor  
 ,qtd\_funcionarios  
 ,tem\_diesel  
 ,tem\_troca\_oleo  
 ,codigo\_externo  
 ,marca\_parceira  
 ,email\_contato  
 ,email\_gerente  
 ,tamanho\_loja  
 ,situacao  
 ,data\_encerramento  
from {environment}.raw.sharepoint\_dados\_filiais  
where 1=1;  
  
""").createOrReplaceTempView('dados\_filiais')

### 2.7 Resumo do Código

O trecho de código Python apresentado realiza uma consulta utilizando a linguagem SQL no ambiente Spark. A consulta seleciona o código da empresa e a data de abertura da filial mais antiga, a partir da tabela "tb\_farol\_faturamento". Os dados são agrupados pelo código da empresa e, em seguida, uma visualização temporária chamada 'tab\_faturamento' é criada para armazenar o resultado da consulta. Este código tem como objetivo extrair informações relevantes sobre a data de abertura das filiais das empresas presentes na tabela, facilitando análises posteriores.

#### Código Fonte

spark.sql(f"""  
  
select  
 cod\_empresa  
 ,to\_date(min(data), 'yyyy-MM-dd') as data\_abertura\_filial  
from {environment}.camada\_destino.tb\_farol\_faturamento  
where 1=1  
group by  
 cod\_empresa;  
  
""").createOrReplaceTempView('tab\_faturamento')

### 2.8 Resumo do Código

O trecho de código Python apresentado tem como finalidade a criação de uma visualização de dados a partir da junção de informações de diferentes tabelas relacionadas a empresas fictícias. Através de consultas SQL, são selecionados e renomeados diversos campos, como código da empresa, CNPJ, razão social, endereço, dados de filiais, faturamento, entre outros. Esses dados são então organizados em uma visualização temporária chamada 'source\_df', que servirá como base para análises posteriores. O código segue uma estrutura clara e eficiente, utilizando funções do Spark para manipulação e processamento dos dados de forma otimizada.

#### Código Fonte

source\_df = spark.sql("""  
   
select   
 emp.cod\_empresa as codigo\_ficticio  
 ,emp.num\_cnpj as doc\_empresa  
 ,emp.nom\_razao\_social as nome\_razao\_social  
 ,emp.nom\_fantasia as nome\_fantasia  
 ,emp.logradouro  
 ,emp.inscricao\_estadual  
 ,emp.inscricao\_municipal  
 ,emp.complemento\_endereco  
 ,emp.codigo\_postal  
 ,emp.bairro  
 ,emp.codigo\_cidade  
 ,fil.regiao  
 ,fil.ga as responsavel\_regional  
 ,fil.categoria\_porte as categoria\_porte  
 ,fat.data\_abertura\_filial as data\_abertura\_atividade  
 ,fil.data\_abertura as data\_abertura\_empresa  
 ,fil.dias\_de\_func as dias\_atividade  
 ,fil.horario\_de\_func as horario\_atividade  
 ,fil.segmento  
 ,fil.tem\_gnv  
 ,fil.tem\_etanol  
 ,fil.tem\_supervisor  
 ,fil.qtd\_funcionarios  
 ,fil.tem\_diesel  
 ,fil.tem\_troca\_oleo  
 ,fil.codigo\_externo  
 ,fil.marca\_parceira  
 ,fil.email\_contato  
 ,fil.email\_gerente  
 ,fil.tamanho\_loja  
 ,fil.situacao  
 ,fil.data\_encerramento  
from tab\_empresa emp  
left join dados\_filiais fil  
 on emp.cod\_empresa = fil.codigo  
left join tab\_faturamento fat  
 on emp.cod\_empresa = fat.cod\_empresa  
where 1=1  
order by emp.nom\_fantasia  
""")  
  
source\_df.createOrReplaceTempView('source\_df')

### 2.9 Resumo do Código

O trecho de código Python apresentado é responsável por realizar a carga de dados em uma tabela Delta, utilizando a lógica de reprocessamento ou merge. Caso a variável "reprocessar" seja verdadeira, os dados são sobrescritos na tabela Delta com a inclusão de um timestamp de inserção. Caso contrário, é realizado um merge entre os dados da fonte e os dados existentes na tabela Delta, atualizando as informações correspondentes e inserindo novos registros, também com a inclusão de um timestamp de inserção. Essa estrutura permite a atualização e inserção de dados de forma eficiente e controlada, garantindo a integridade e consistência das informações na tabela Delta.

#### Código Fonte

if reprocessar:  
 source\_df = source\_df.withColumn("data\_insercao", lit(current\_timestamp() ) )  
 source\_df.write.format("delta")\  
 .mode("overwrite")\  
 .saveAsTable(f"{delta\_table\_name}", path=f"{delta\_file}")  
 print(f"Carga overwrite realizada com sucesso! {delta\_table\_name}")  
else:  
 print("Realizando carga em Merge..")  
 spark.sql(f"""  
  
MERGE INTO {delta\_table\_name} AS target  
USING source\_df AS source  
 ON target.codigo\_ficticio = source.codigo\_ficticio  
WHEN MATCHED THEN  
 UPDATE SET  
 target.doc\_empresa = source.doc\_empresa,  
 target.nome\_razao\_social = source.nome\_razao\_social,  
 target.nome\_fantasia = source.nome\_fantasia,  
 target.logradouro = source.logradouro,  
 target.inscricao\_estadual = source.inscricao\_estadual,  
 target.inscricao\_municipal = source.inscricao\_municipal,  
 target.complemento\_endereco = source.complemento\_endereco,  
 target.codigo\_postal = source.codigo\_postal,  
 target.bairro = source.bairro,  
 target.codigo\_cidade = source.codigo\_cidade,  
 target.regiao = source.regiao,  
 target.responsavel\_regional = source.responsavel\_regional,  
 target.categoria\_porte = source.categoria\_porte,  
 target.data\_abertura\_atividade = source.data\_abertura\_atividade,  
 target.data\_abertura\_empresa = source.data\_abertura\_empresa,  
 target.dias\_atividade = source.dias\_atividade,  
 target.horario\_atividade = source.horario\_atividade,  
 target.segmento = source.segmento,  
 target.tem\_gnv = source.tem\_gnv,  
 target.tem\_etanol = source.tem\_etanol,  
 target.tem\_supervisor = source.tem\_supervisor,  
 target.qtd\_funcionarios = source.qtd\_funcionarios,  
 target.tem\_diesel = source.tem\_diesel,  
 target.tem\_troca\_oleo = source.tem\_troca\_oleo,  
 target.codigo\_externo = source.codigo\_externo,  
 target.marca\_parceira = source.marca\_parceira,  
 target.email\_contato = source.email\_contato,  
 target.email\_gerente = source.email\_gerente,  
 target.tamanho\_loja = source.tamanho\_loja,  
 target.situacao = source.situacao,  
 target.data\_encerramento = source.data\_encerramento,  
 target.data\_atualizacao = current\_timestamp()  
WHEN NOT MATCHED THEN  
 INSERT (  
 codigo\_ficticio,  
 doc\_empresa,  
 nome\_razao\_social,  
 nome\_fantasia,  
 logradouro,  
 inscricao\_estadual,  
 inscricao\_municipal,  
 complemento\_endereco,  
 codigo\_postal,  
 bairro,  
 codigo\_cidade,  
 regiao,  
 responsavel\_regional,  
 categoria\_porte,  
 data\_abertura\_atividade,  
 data\_abertura\_empresa,  
 dias\_atividade,  
 horario\_atividade,  
 segmento,  
 tem\_gnv,  
 tem\_etanol,  
 tem\_supervisor,  
 qtd\_funcionarios,  
 tem\_diesel,  
 tem\_troca\_oleo,  
 codigo\_externo,  
 marca\_parceira,  
 email\_contato,  
 email\_gerente,  
 tamanho\_loja,  
 situacao,  
 data\_encerramento,  
 data\_insercao  
 )  
 VALUES (  
 source.codigo\_ficticio,  
 source.doc\_empresa,  
 source.nome\_razao\_social,  
 source.nome\_fantasia,  
 source.logradouro,  
 source.inscricao\_estadual,  
 source.inscricao\_municipal,  
 source.complemento\_endereco,  
 source.codigo\_postal,  
 source.bairro,  
 source.codigo\_cidade,  
 source.regiao,  
 source.responsavel\_regional,  
 source.categoria\_porte,  
 source.data\_abertura\_atividade,  
 source.data\_abertura\_empresa,  
 source.dias\_atividade,  
 source.horario\_atividade,  
 source.segmento,  
 source.tem\_gnv,  
 source.tem\_etanol,  
 source.tem\_supervisor,  
 source.qtd\_funcionarios,  
 source.tem\_diesel,  
 source.tem\_troca\_oleo,  
 source.codigo\_externo,  
 source.marca\_parceira,  
 source.email\_contato,  
 source.email\_gerente,  
 source.tamanho\_loja,  
 source.situacao,  
 source.data\_encerramento,  
 current\_timestamp()  
 )  
   
 """)

Documento Interno - A divulgação sem autorização prévia viola as normas e diretrizes da organização.