**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 geral do projeto, o papel dos notebooks listados e o valor que este documento oferece à equipe técnica ou ao negócio.  
  
O projeto em questão envolve a manipulação e análise de dados da empresa fictícia anonimizada. Os notebooks "dim\_empresa\_ficticia\_anonimizado.md" e "dim\_empresa\_ficticia\_anonimizado\_completo.md" desempenham um papel fundamental na ingestão, tratamento e análise dos dados referentes à empresa fictícia.   
  
O primeiro notebook, "dim\_empresa\_ficticia\_anonimizado.md", é responsável pela anonimização dos dados da empresa, garantindo a privacidade e segurança das informações. Já o segundo notebook, "dim\_empresa\_ficticia\_anonimizado\_completo.md", realiza o tratamento completo dos dados, preparando-os para análises mais aprofundadas.  
  
Este documento de Implementação Técnica (DIT) tem como objetivo fornecer um guia claro e detalhado sobre a execução dos notebooks mencionados, contribuindo para a eficiência e qualidade do trabalho da equipe técnica. Além disso, o documento visa facilitar a compreensão do processo de manipulação de dados e seu impacto nos resultados do negócio, promovendo uma melhor tomada de decisão e otimização de processos.

# 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 coletar e processar informações relevantes para a empresa fictícia de forma anônima. Essa função desempenha um papel fundamental no processo de obtenção e preparação dos dados necessários para análises posteriores, contribuindo assim para a tomada de decisões estratégicas com base em informações confiáveis e atualizadas.

#### Código Fonte

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

### 1.2 Resumo do Código

O trecho de código Python 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 definidas variáveis para o nome da tabela, o caminho do arquivo delta, o nome da tabela delta e um comentário explicativo sobre a tabela. Ao final, são exibidos os valores das variáveis delta\_table\_name e delta\_file. Este código é essencial para a preparação e configuração do ambiente de armazenamento de dados, facilitando a manipulação e consulta das informações relacionadas às 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" cria um widget interativo denominado `reprocessar` para controlar a carga da dimensão. Esse widget exibe um dropdown com as opções "True" e "False", permitindo ao usuário escolher de forma interativa se a carga será completa (`overwrite`) ou incremental (`merge`). A variável `reprocessar` será atribuída o valor `True` somente se o usuário selecionar a opção correspondente no notebook, garantindo assim a flexibilidade na escolha do tipo de carga a ser realizada.

#### 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 refere-se à criação de uma tabela no ambiente Spark, utilizando o formato Delta. A tabela é definida com uma série de colunas que representam informações de uma empresa fictícia anonimizada, como código, CNPJ, razão social, endereço, dados de funcionamento, entre outros. A utilização da cláusula IF NOT EXISTS garante que a tabela só será criada se ainda não existir, evitando possíveis erros de duplicidade. Além disso, a definição de colunas com tipos de dados específicos e a inclusão de comentários facilitam a compreensão e manutenção do código.

#### 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 de empresas, selecionando diversas informações como código da empresa, CNPJ, nome fantasia, razão social, endereço, inscrições estadual e municipal, complemento, CEP, 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 tem como objetivo principal criar uma visualização temporária com os dados das empresas 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, área, porte, metragem da loja, data de abertura, dias e horário de funcionamento, categoria, 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 dentro do 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 da 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 visa fornecer uma visão consolidada das datas de abertura das filiais em relação a cada empresa, facilitando análises e tomadas de decisão no contexto empresarial.

#### 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 finalidade a criação de uma visualização de dados a partir da união de informações de empresas, filiais e faturamento. Através de consultas SQL, são selecionados diversos atributos, como código da empresa, CNPJ, razão social, endereço, dados de funcionamento da filial, entre outros. Esses dados são então organizados em uma tabela temporária chamada 'source\_df', que servirá como base para análises posteriores. A consulta é estruturada de forma a garantir a integridade e consistência dos dados, ordenando-os pelo nome fantasia da empresa.

#### 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 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 inseridos na tabela Delta sobrescrevendo os dados existentes. Caso contrário, é realizado um merge entre os dados de origem e os dados da tabela Delta, atualizando as informações correspondentes e inserindo novos registros, se necessário. O código demonstra um processo de integração de dados eficiente e flexível, garantindo a consistência e atualização das informações na tabela Delta.

#### 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 arquivo de configuração "ingestion\_function". Essa função é responsável por realizar a coleta e processamento dos dados da empresa fictícia de forma anonimizada e completa. A execução desse código é fundamental para garantir a integridade e qualidade dos dados que serão utilizados nas análises posteriores.

#### Código Fonte

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

### 2.2 Resumo do Código

O trecho de código Python 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 definidas variáveis para o nome da tabela, o caminho do arquivo delta, o nome da tabela delta e um comentário sobre a tabela. Ao final, são exibidos na saída os valores das variáveis delta\_table\_name e delta\_file. Este código é essencial para a configuração e identificação da tabela de dimensão no ambiente de armazenamento de dados.

#### 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 forma interativa, se a carga da dimensão será completa (`overwrite`) ou incremental (`merge`). Essa funcionalidade proporciona flexibilidade ao usuário no processo de carga de dados, adaptando-se às necessidades específicas de cada situação.

#### 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 Delta Lake, utilizando a linguagem SQL no Spark. A tabela é definida com uma série de colunas que representam informações de uma empresa fictícia, como código, nome, endereço, dados de contato, segmento de atuação, entre outros. Além disso, são especificados os tipos de dados de cada coluna e a geração automática de uma chave primária para identificação única das empresas. A tabela é armazenada em um local específico no Delta Lake e possui um comentário para documentação adicional. Este código é essencial para a estruturação e organização dos dados da empresa 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\_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 apresentado realiza uma consulta SQL utilizando o Apache Spark para selecionar diversas informações de empresas da tabela "tb\_tab\_empresa\_farol". Os campos selecionados incluem o código da empresa, CNPJ, nome fantasia, razão social, endereço completo, inscrição estadual e municipal, complemento de endereço, código postal, bairro e código da cidade. Esses dados são então temporariamente armazenados na visualização "tab\_empresa" para futuras análises e manipulações dentro do ambiente especificado.

#### 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 a partir de uma fonte de dados específica e os resultados são temporariamente armazenados em uma visualização de dados chamada 'dados\_filiais'. Essa operação é essencial para a análise e manipulação dos dados das filiais da empresa fictícia, permitindo a extração de informações relevantes para tomadas de decisão e estratégias de negócio.

#### 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" do ambiente especificado. 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, facilitando análises posteriores relacionadas ao faturamento.

#### 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 união 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', permitindo análises posteriores. O código também inclui cláusulas de junção e ordenação para garantir a integridade e relevância dos dados obtidos.

#### 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 realizada a operação de merge, atualizando os registros existentes com base em uma chave de correspondência ou inserindo novos registros caso não haja correspondência. O código demonstra um processo de integração de dados eficiente e flexível, garantindo a consistência e atualização da tabela Delta de forma automatizada.

#### 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.