Semana 02 - DBSQL

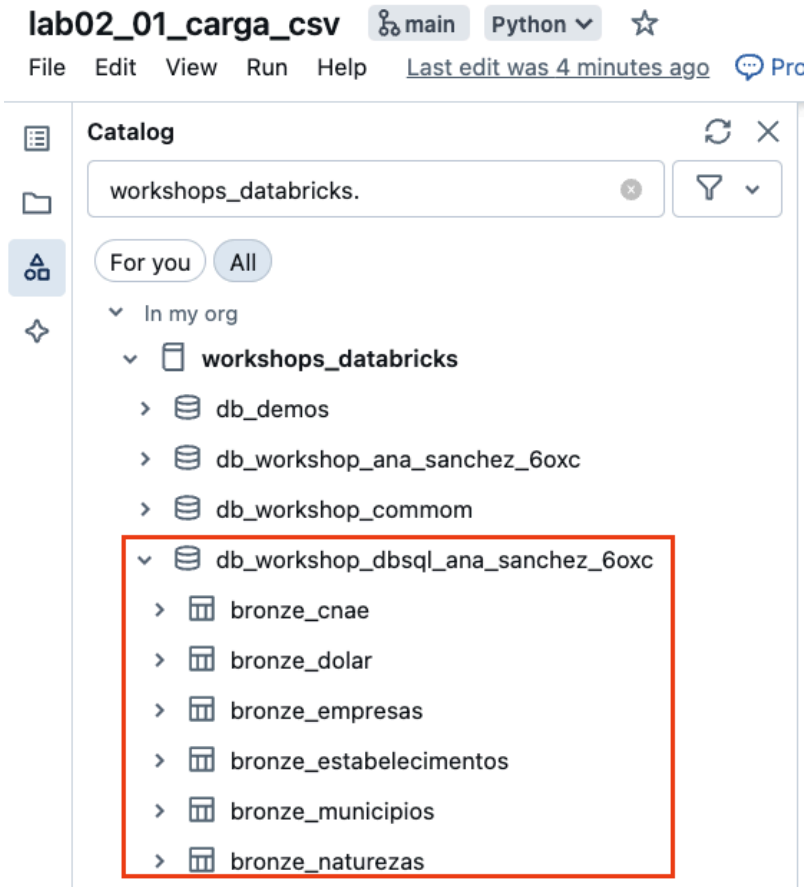
Hands-On Lab 00 - Carregar algumas tabelas prévias com Python 3
Hands-On Lab 01 - Criação de tabelas com DBSQL 4
Objetivo do exercício 4
Sessão 01: Estrutura TABELAS, DATABASE e CATÁLOGOS 4
Exercício 01.01 - Criação da tabela 5
Exercício 01.02 - Inserindo dados na Tabela através de SQL INSERT 6
Exercício 01.03 - Verificando o conteúdo da TABELA 7
Exercício 01.04 - Alterando o conteúdo da TABELA 7
Exercício 01.05 - Visualizando o Histórico de Atualizações da tabela 9
Exercício 01.06 - Visalizando o conteúdo da tabela na versão anterior (Time Travel) 10
Exercício 01.07 - Restaurando o conteúdo da tabela na versão anterior 10
Exercício 01.08 - Visualizando as propriedades da Tabela 10
Exercício 01.09 - Visualizando as informações detalhadas da Tabela 10
Hands- On 02 - Constraints 11
Exercício 02.01 - Campos NOT NULL 11
Exercício 02.02 - Check Constraint - Garantir a Qualidade dos Dados 12
Exercício 02.03 - PK Constraint - Chave Primaria 14
Exercício 02.01 - FK Constraint - Chave estrangeira 16
Hands-On Lab 03 - Criação da camada Silver 21
Objetivo do exercício 21
Exercício 03.01 - Criação da tabela Silver 21
Exercício 03.02 - Adicionando comentários 22
Exercício 03.03 - Carregando Dados JSON 23
Exercício 03.04 - Criando Streaming Tables 23
Exercício 03.06 - Criando Materialized Views 24
Hands-On Lab 04 - Filtrando e Mascarando linhas e colunas 25
Exercício 04.01 - Filtrando os registros 25
Exercício 04.02 - Mascarando os registros 28
Hands-On Lab 05 - Criação de Alertas 30
Objetivo do exercício 30
Exercício 05.01 - Criação da Query 30
Exercício 05.02 - Criando o Alerta 31
Hands-On Lab 06 - DBSQL API 39
Objetivo do exercício 39
Exercício 06.01 - Testes com o DBSQL API 39
Hands-On Lab 07 - Criando Dashboards 40
Objetivo do exercício 40
Exercício 07.01 - Criando o dashboard 40

Exercício 07.02 - Criando o dataset 40
Exercício 07.03 - Criando visualizações no Canvas 42
Exercício 07.04 - Criando Filtros 45
Exercício 07.05 - Adicionando Cabeçalhos 47
Hands-On Lab 08 - Conversando com os dados com o AI/BI Genie 50
Objetivo do exercício 50
Exercício 08.01 - Criando e configurando seu espaço 50

Hands-On Lab 00 - Carregar algumas tabelas prévias com Python

1. Dentro das pastinhas do repositório que você importou, execute o arquivo **lab00_carga_csv** dentro do diretório **00_Load Data.**

2. Verifique se as tabelas foram criadas com sucesso.

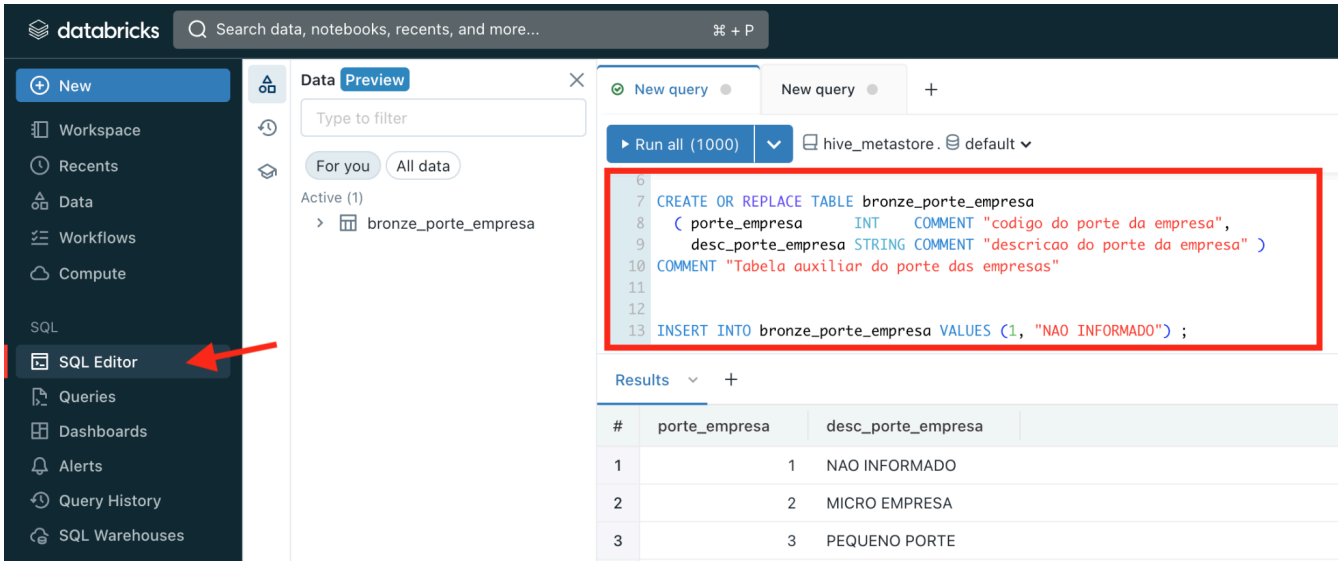


Hands-On Lab 01 - Criação de tabelas com DBSQL

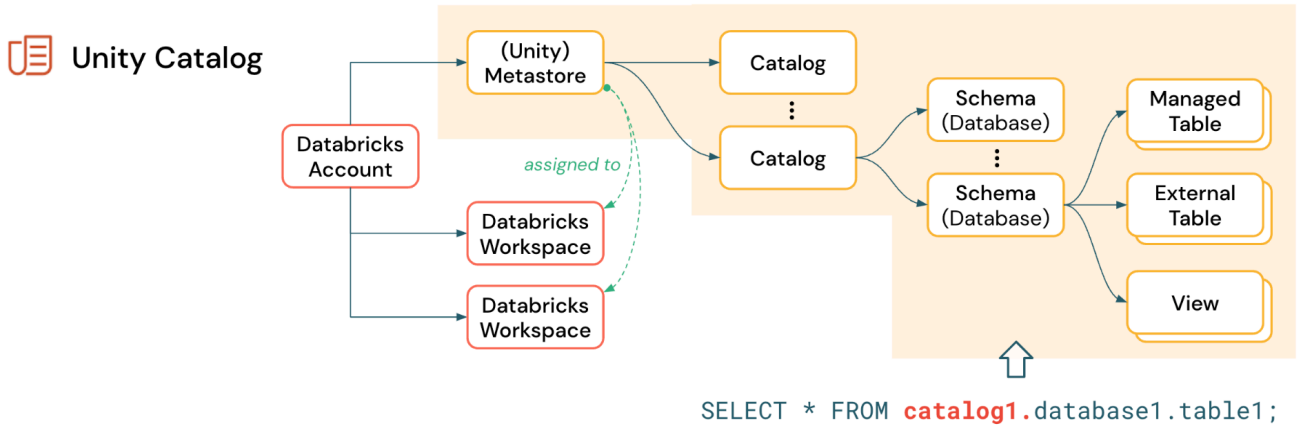
Objetivo do exercício

O objetivo desse laboratório é conhecer as funcionalidades de consulta (Query) da plataforma Databricks, utilizando a linguagem SQL (e as interfaces visuais), explorando os potenciais Analíticos.

Os exercícios deverão ser executados na opção do Menu lateral "SQL Editor".



Sessão 01: Estrutura TABELAS, DATABASE e CATÁLOGOS



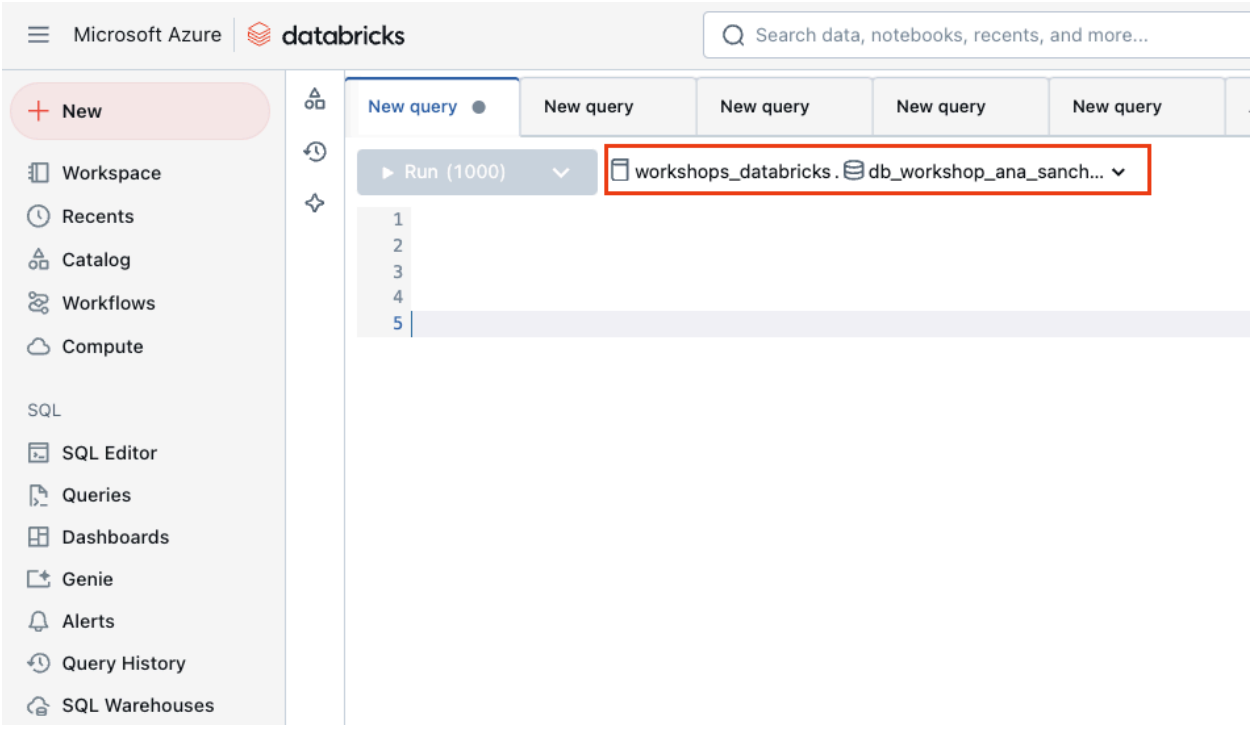
Tópico	Comando
Catálogo	CREATE CATALOG <nome_catalogo>
Schema	CREATE DATABASE IF NOT EXISTS <nome_catalogo>.<nome_database>
Tabela	CREATE OR REPLACE TABLE <nome_catalogo>.<nome_database>.<nome_tabela>
View	CREATE OR REPLACE VIEW <nome_catalogo>.<nome_database>.<nome_view>

Referência:

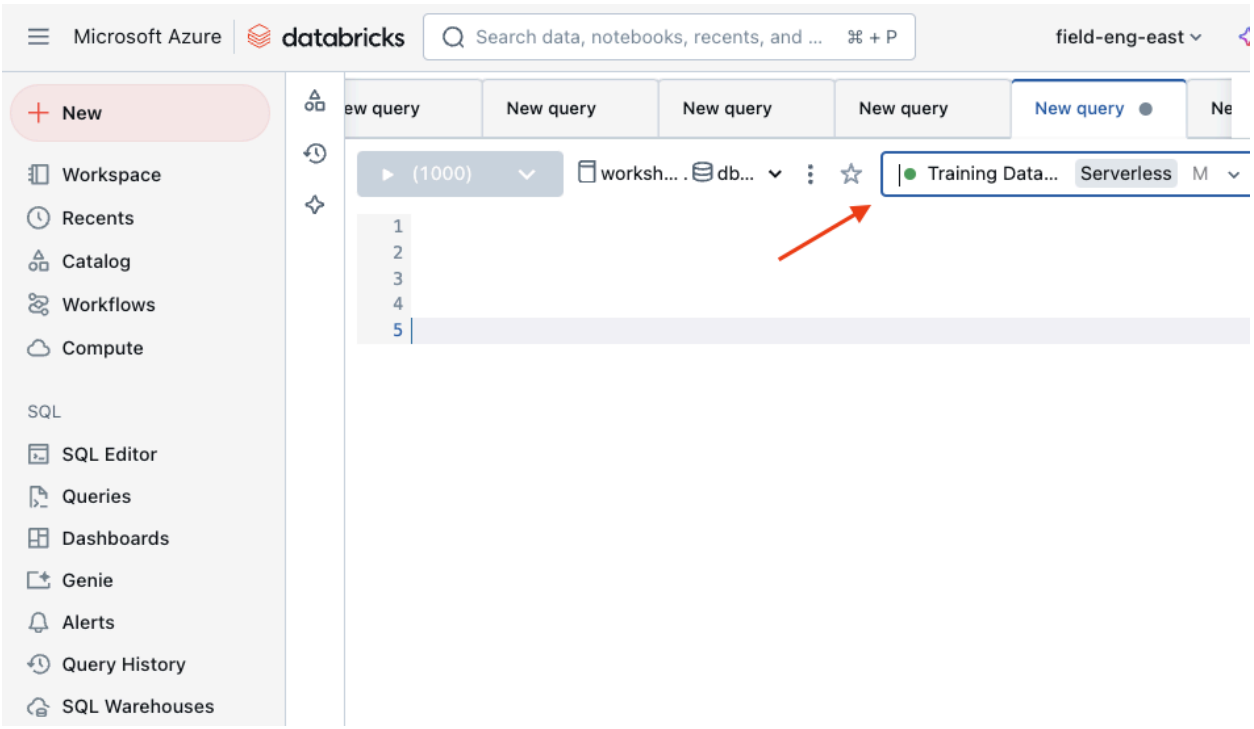
- Databricks Help - DDL Syntax

Exercício 01.01 - Criação da tabela

Selecione seu catálogo (**workshops_databricks**) e seu schema (**db_workshop_SEUNOME_OU_SUACHAVE**) no topo do Query Editor:

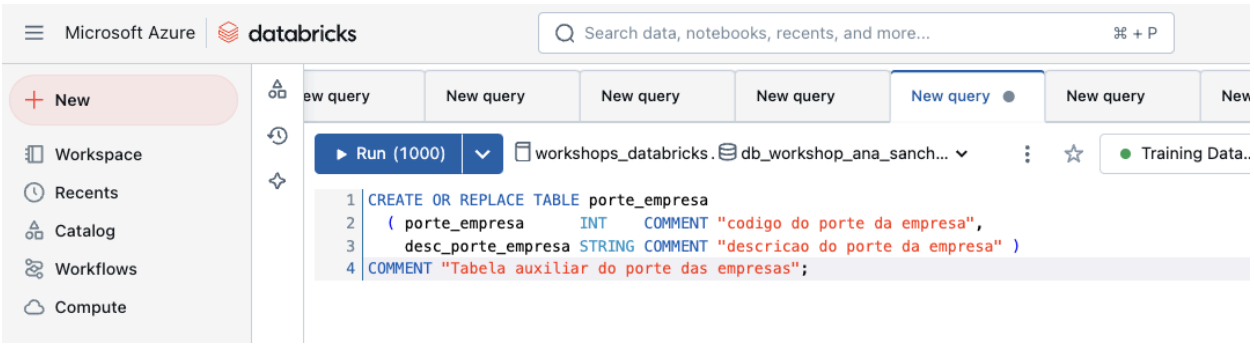


Selecione o warehouse Databricks Training:



Digite a query abaixo e clique em **Run**:

```
CREATE OR REPLACE TABLE porte_empresa
  ( porte_empresa      LONG    COMMENT "codigo do porte da empresa",
    desc_porte_empresa STRING COMMENT "descricao do porte da empresa" )
COMMENT "Tabela auxiliar do porte das empresas";
```



Exercício 01.02 - Inserindo dados na Tabela através de SQL INSERT

```
INSERT INTO porte_empresa VALUES (1, "NAO INFORMADO") ;
```

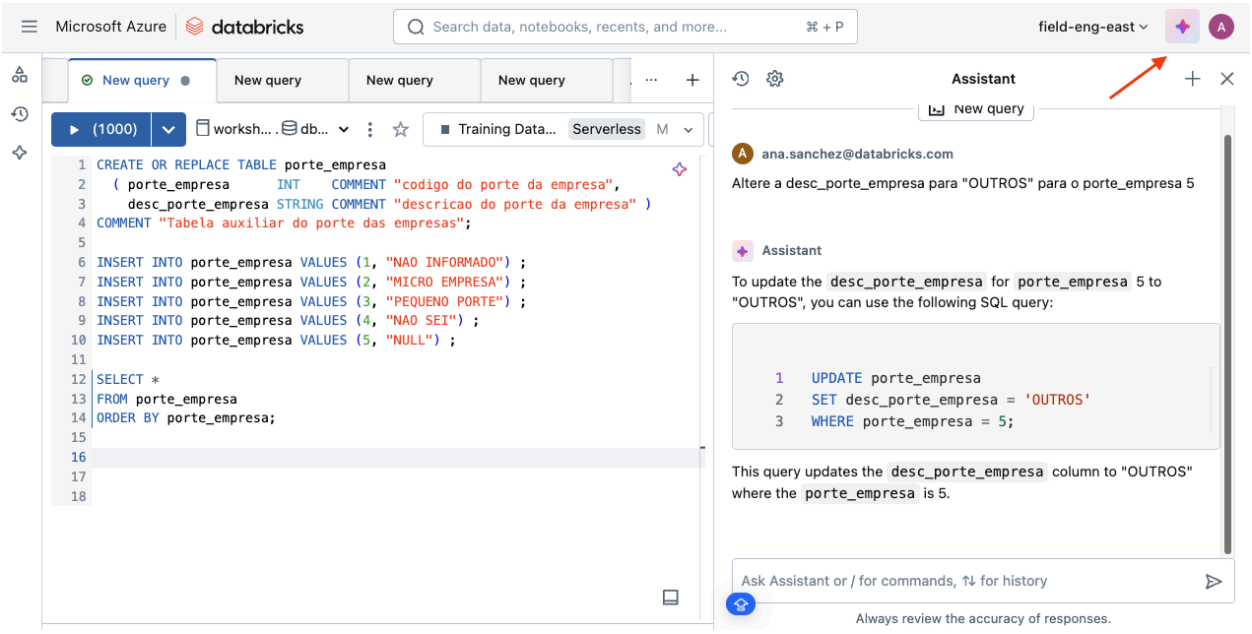
```
INSERT INTO porte_empresa VALUES (2, "MICRO EMPRESA") ;
INSERT INTO porte_empresa VALUES (3, "PEQUENO PORTE") ;
INSERT INTO porte_empresa VALUES (4, "NAO SEI") ;
INSERT INTO porte_empresa VALUES (5, "NULL") ;
```

Exercício 01.03 - Verificando o conteúdo da TABELA

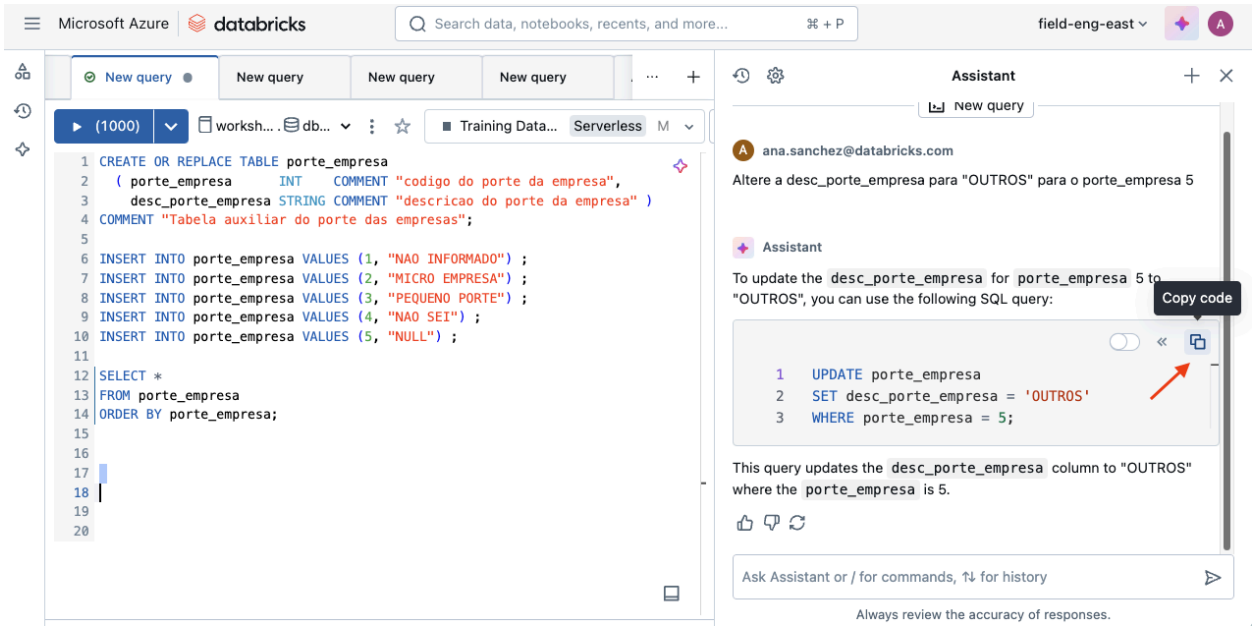
```
SELECT *
FROM porte_empresa
ORDER BY porte_empresa;
```

Exercício 01.04 - Alterando o conteúdo da TABELA

Clique no assistante e peça para ele: `Altere a desc_porte_empresa para "OUTROS" para o porte_empresa 5`



Arraste o mouse até o quadro cinza onde contém a resposta e clique no botão apresentado abaixo para copiar o código gerado:




```
UPDATE porte_empresa
SET desc_porte_empresa = "OUTROS"
WHERE porte_empresa = 5;
```

Agora repita o mesmo processo para fazer o seguinte pedido: `Delete o registro da tabela porte_empresa para o porte_empresa 4`


```
DELETE
FROM porte_empresa
WHERE porte_empresa = 4;
```

Exercício 01.05 - Visualizando o Histórico de Atualizações da tabela

```
DESCRIBE HISTORY porte_empresa;
```

Exercício 01.06 - Visalizando o conteúdo da tabela na versão anterior (Time Travel)

```
SELECT * FROM porte_empresa VERSION AS OF 5;
```

Exercício 01.07 - Restaurando o conteúdo da tabela na versão anterior

```
RESTORE TABLE porte_empresa TO VERSION AS OF 5;
```

Exercício 01.08 - Visualizando as propriedades da Tabela

```
DESCRIBE DETAIL porte_empresa;
```

Exercício 01.09 - Visualizando as informações detalhadas da Tabela

```
DESCRIBE TABLE EXTENDED porte_empresa;
```

Hands- On 02 - Constraints

Documentação: [Constraints on Databricks | Databricks on AWS]

Exercício 02.01 - Campos NOT NULL

Para adicionarmos uma coluna como chave primária precisamos garantir que uma coluna seja NOT NULL, entã para isso, vamos adicionar essa informação nas colunas que iremos transformar em chaves primárias.

```
ALTER TABLE bronze_cnae
CHANGE COLUMN cod_cnae cod_cnae BIGINT NOT NULL;

ALTER TABLE bronze_empresas
CHANGE COLUMN cnpj_basico cnpj_basico BIGINT NOT NULL;

ALTER TABLE bronze_municipios
CHANGE COLUMN cod_municipio cod_municipio BIGINT NOT NULL;

ALTER TABLE bronze_naturezas
CHANGE COLUMN codigo codigo BIGINT NOT NULL;

ALTER TABLE porte_empresa
CHANGE COLUMN porte_empresa porte_empresa LONG NOT NULL;
```

Teste uma inserção de um dado nulo em uma coluna not null.

```
INSERT INTO porte_empresa ( desc_porte_empresa)
```

```
VALUES ( "TESTE");
```

Note que o seguinte erro irá aparecer, isso porque adicionando NOT NULL na coluna, impedimos que seja inserido registros que não contenham o id do porte_empresa (coluna porte_empresa):

```
53 INSERT INTO porte_empresa ( desc_porte_empresa)
54 VALUES ( "TESTE");
55
```

[DELTA_INSERT_COLUMN_MISMATCH] Column porte_empresa is not specified in INSERT

Diagnose error

Agora faça o seguinte teste, insira um registro somente com o ID, e verifique o que vai acontecer:

```
INSERT INTO porte_empresa ( porte_empresa)
VALUES ( 6);
```

O que ocorreu? E porque?

Raw results +

	num_affected_rows	num_inserted_rows
1	1	1

Agora, apague o registro:

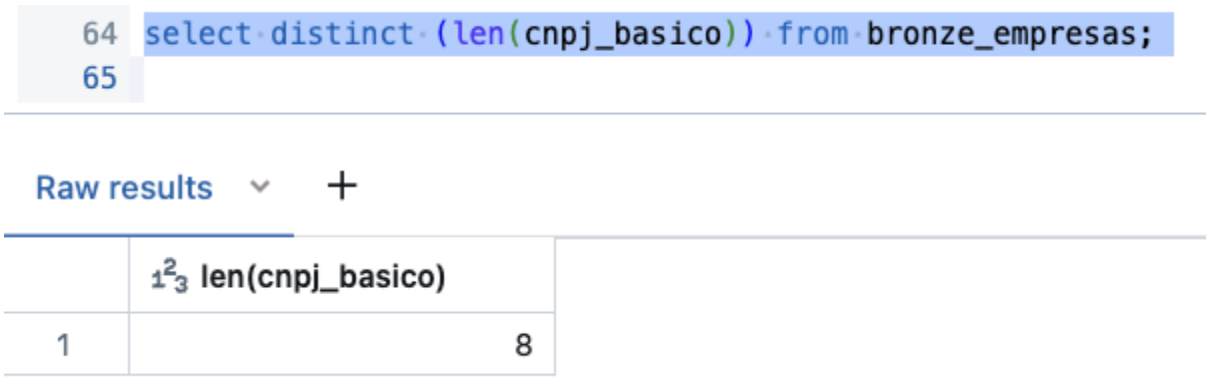
```
DELETE FROM porte_empresa
WHERE porte_empresa = 6;
```

Exercício 02.02 - Check Constraint - Garantir a Qualidade dos Dados

Agora criaremos uma constraint de checagem (**CHECK)** , ou seja, iremos bloquear os registros caso não atendam os requisitos de checagem.

Para isso, primeiro vamos verificar se a tabela atual atende os requisitos que iremos criar, vamos verificar se há um padrão na quantidade de caracteres do campo **cnpj_basico** na tabela **bronze_empresas**. (cnpj básico não tem o 0001 e os dois últimos dígitos)

```
select distinct (len(cnpj_basico)) from bronze_empresas;
```



Agora vamos adicionar a constraint:

```
ALTER TABLE bronze_empresas ADD CONSTRAINT cnpjQntCaracteres CHECK
(len(cnpj_basico) = 8);
```

E agora, vamos tentar inserir um valor não válido para essa constraint:

```
INSERT INTO bronze_empresas (cnpj_basico)
VALUES (1243);
```

```
68 INSERT INTO bronze_empresas ( cnpj_basico)
69 VALUES (1243);
```

```
[DELTA_VIOLATE_CONSTRAINT_WITH_VALUES] CHECK constraint
cnpjqntcaracteres (len(cnpj_basico) = 8) violated by row with values:
- cnpj_basico : 1243
```

Raw results +

Faça um teste sem violar a constraint para verificar se funciona:

```
INSERT INTO bronze_empresas ( cnpj_basico)
VALUES (12345678);
```

Raw results +

	num_affected_rows	num_inserted_rows
1	1	1

Agora apague esse registro:

```
DELETE FROM bronze_empresas
WHERE cnpj_basico = 12345678;
```

Exercício 02.03 - PK Constraint - Chave Primaria

Adicione as seguintes chaves primarias:

```
ALTER TABLE bronze_cnae
```

```
ADD CONSTRAINT code_cnae_pk PRIMARY KEY (cod_cnae);

ALTER TABLE bronze_empresas
ADD CONSTRAINT cnpj_basico_pk PRIMARY KEY (cnpj_basico);

ALTER TABLE bronze_municipios
ADD CONSTRAINT cod_municipio_pk PRIMARY KEY (cod_municipio);

ALTER TABLE bronze_naturezas
ADD CONSTRAINT codigo_pk PRIMARY KEY (codigo);

ALTER TABLE porte_empresa
ADD CONSTRAINT porte_empresa_pk PRIMARY KEY (porte_empresa);
```

Verifique se elas foram adicionadas no **Catalog Explorer**.

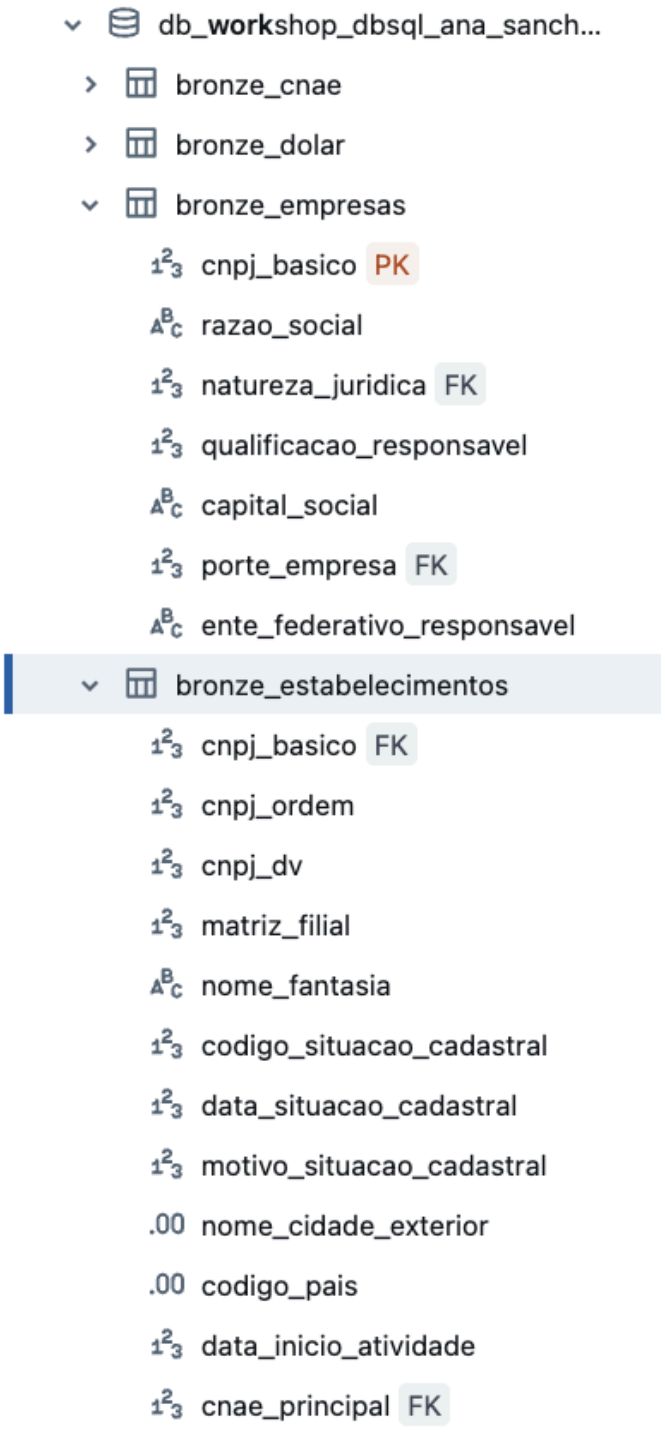
Exercício 02.01 - FK Constraint - Chave estrangeira

Adicione as seguintes chaves estrangeiras:

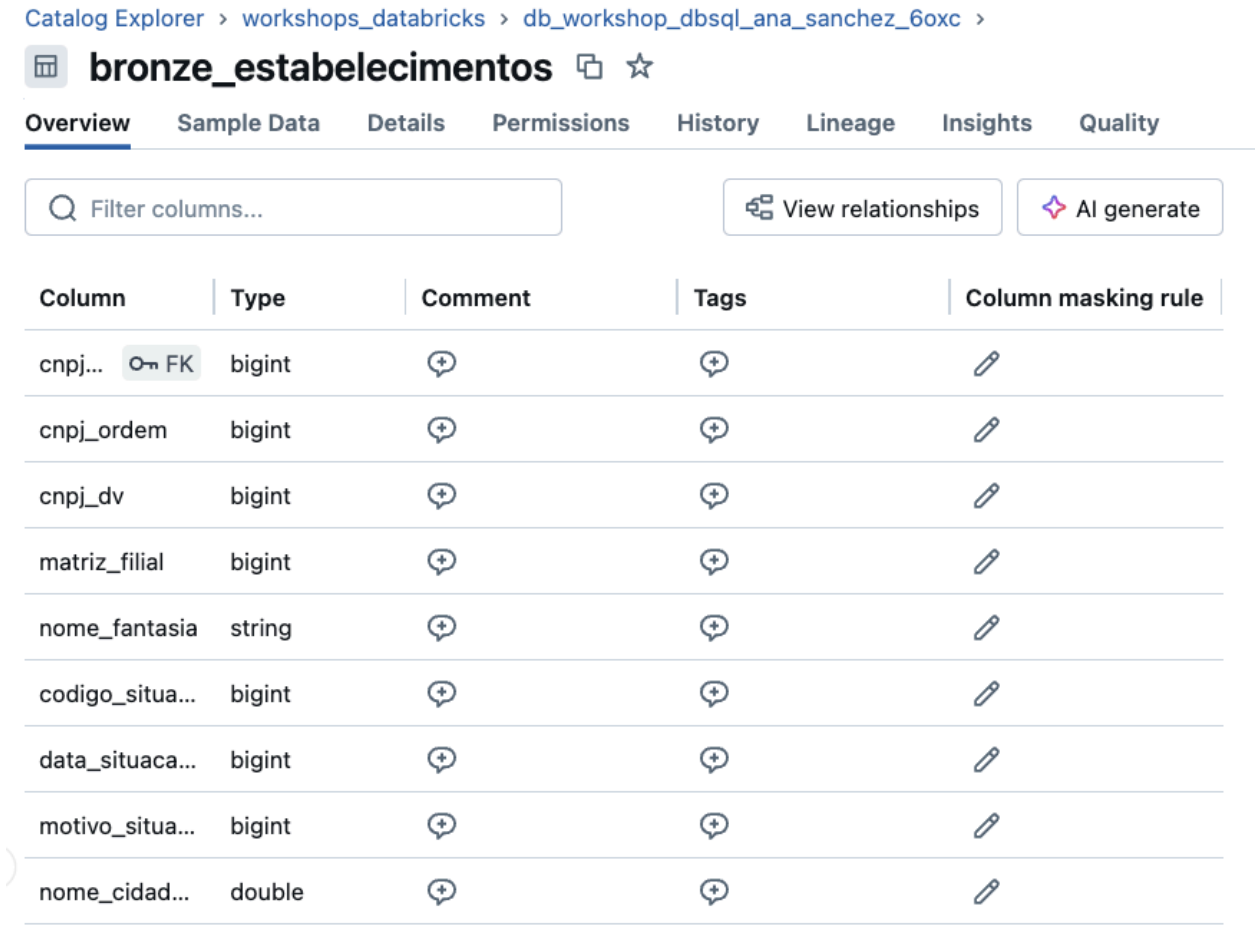
```
ALTER TABLE bronze_estabelecimentos
ADD CONSTRAINT fk_cnpj_basico
```

```
FOREIGN KEY (cnpj_basico)
REFERENCES bronze_empresas(cnpj_basico);

ALTER TABLE bronze_estabelecimentos
ADD CONSTRAINT fk_cnae_principal
FOREIGN KEY (cnae_principal)
REFERENCES bronze_cnae(cod_cnae);

ALTER TABLE bronze_empresas
ADD CONSTRAINT fk_porte_empresa
FOREIGN KEY (porte_empresa)
REFERENCES porte_empresa(porte_empresa);

ALTER TABLE bronze_empresas
ADD CONSTRAINT fk_natureza_juridica
FOREIGN KEY (natureza_juridica)
REFERENCES bronze_naturezas(codigo);
```

Verifique se elas foram adicionadas no **Catalog Explorer**.

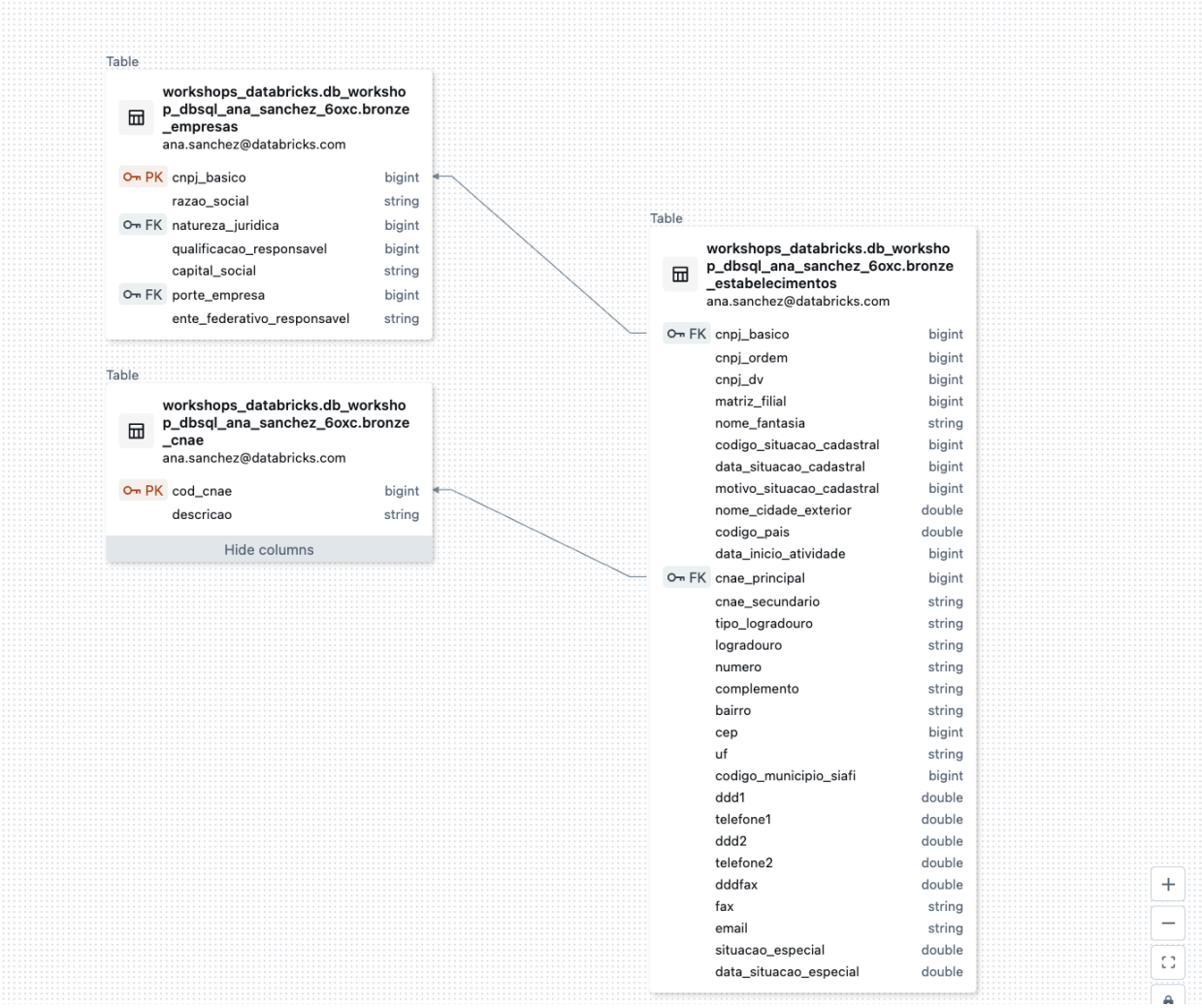

db_workshop_dbsql_ana_sanch...
bronze_cnae
bronze_dolar
bronze_empresas
cnpj_basico PK
razao_social
natureza_juridica FK
qualificacao_responsavel
capital_social
porte_empresa FK
ente_federativo_responsavel
bronze_estabelecimentos
cnpj_basico FK
cnpj_ordem
cnpj_dv
matriz_filial
nome_fantasia
codigo_situacao_cadastral
data_situacao_cadastral
motivo_situacao_cadastral
nome_cidade_exterior
codigo_pais
data_inicio_atividade
cnae_principal FK

Acesse a tabela através do **Catalog Explorer** e clique em **View relationships.**



E vizualize o diagrame de entidade relacional gerado:

Entity Relationship Diagram for workshops_databricks.db_workshop_dbsql_ana_sanchez_6oxc.bronze_es...



Hands-On Lab 03 - Criação da camada Silver

Objetivo do exercício

O objetivo deste exercício é demonstrar como criar uma tabela na camada 'Silver' a partir das outras tabelas já ingeridas no laboratório anterior.

Exercício 03.01 - Criação da tabela Silver

```
CREATE OR REPLACE TABLE silver_empresas
AS SELECT
  est.cnpj_basico AS cnpj_basico,
  matriz_filial AS nome_matriz,
  nome_fantasia AS nome_fantasia_empresa,
  razao_social AS nome_razao_social,
  codigo_situacao_cadastral AS cod_situacao_cadastral,
  data_situacao_cadastral AS data_situacao_cadastral,
  motivo_situacao_cadastral AS motivo_situacao_cadastral,
  data_inicio_atividade AS data_inicio_atividade,
  cnae_principal AS cnae_principal,
  cnae.descricao AS cnae_descricao,
  tipo_logradouro AS endereco_tipo_logradouro,
  logradouro AS endereco_nome_logradouro,
  numero AS endereco_numero_logradouro,
  bairro AS endereco_bairro_logradouro,
  cep AS endereco_numero_cep,
  uf AS endereco_unidade_federativa,
  codigo_municipio_siafi AS codigo_municipio_siafi,
  natureza_juridica AS cod_natureza_juridica,
  nat.descricao AS desc_natureza_juridica,
  qualificacao_responsavel AS qualificacao_responsavel,
  capital_social AS val_capital_social,
  emp.porte_empresa AS cod_porte_empresa,
  porte.desc_porte_empresa AS desc_porte_empresa,
  ente_federativo_responsavel AS ente_federativo_responsavel
from bronze_estabelecimentos est
join bronze_empresas emp
on est.cnpj_basico = emp.cnpj_basico
left join bronze_cnae cnae
on est.cnae_principal = cnae.cod_cnae
left join porte_empresa porte
```

```
on emp.porte_empresa = porte.porte_empresa
left join bronze_naturezas nat
on emp.natureza_juridica = nat.codigo;
```

Verifique no **Catalog Explorer** se a tabela realmente foi criada.

Exercício 03.02 - Adicionando comentários

```
COMMENT ON TABLE silver_empresas IS 'Tabela contendo dados das empresas';

ALTER TABLE silver_empresas ALTER COLUMN cnpj_basico COMMENT 'número do CNPJ raiz
de 8 posições';
ALTER TABLE silver_empresas ALTER COLUMN nome_matriz COMMENT 'Nome da Matriz';
ALTER TABLE silver_empresas ALTER COLUMN nome_fantasia_empresa COMMENT 'Nome
fantasia da empresa';
ALTER TABLE silver_empresas ALTER COLUMN nome_razao_social COMMENT 'Nome da Razão
Social';
ALTER TABLE silver_empresas ALTER COLUMN cod_situacao_cadastral COMMENT 'Código da
Situação Cadastral';
ALTER TABLE silver_empresas ALTER COLUMN data_situacao_cadastral COMMENT 'Data da
Situação Cadastral';
ALTER TABLE silver_empresas ALTER COLUMN motivo_situacao_cadastral COMMENT 'Motivo
da Situação Cadastral';
ALTER TABLE silver_empresas ALTER COLUMN data_inicio_atividade COMMENT 'Data de
início da atividade';
ALTER TABLE silver_empresas ALTER COLUMN cnae_principal COMMENT 'CNAE Código da
Natureza Economica';
ALTER TABLE silver_empresas ALTER COLUMN cnae_descricao COMMENT 'Descrição do
CNAE';
ALTER TABLE silver_empresas ALTER COLUMN endereco_tipo_logradouro COMMENT 'Endereço
- Tipo de Logradouro';
ALTER TABLE silver_empresas ALTER COLUMN endereco_nome_logradouro COMMENT 'Endereço
- Nome do Logradouro';
ALTER TABLE silver_empresas ALTER COLUMN endereco_numero_logradouro COMMENT
'Endereço - Número do Logradouro';
ALTER TABLE silver_empresas ALTER COLUMN endereco_bairro_logradouro COMMENT
'Endereço - Bairro do Logradouro';
ALTER TABLE silver_empresas ALTER COLUMN endereco_numero_cep COMMENT 'Endereço -
número do CEP';
ALTER TABLE silver_empresas ALTER COLUMN endereco_unidade_federativa COMMENT
'Endereço - Unidade Federativa';
ALTER TABLE silver_empresas ALTER COLUMN codigo_municipio_siafi COMMENT 'Código do
Município SIAFI';
ALTER TABLE silver_empresas ALTER COLUMN cod_natureza_juridica COMMENT 'Código da
Natureza Jurídica';
```

```
ALTER TABLE silver_empresas ALTER COLUMN desc_natureza_juridica COMMENT 'Descrição da Natureza Jurídica';
ALTER TABLE silver_empresas ALTER COLUMN qualificacao_responsavel COMMENT 'Qualificação do Responsável';
ALTER TABLE silver_empresas ALTER COLUMN val_capital_social COMMENT 'Valor do Capital Social';
ALTER TABLE silver_empresas ALTER COLUMN cod_porte_empresa COMMENT 'Código do Porte da Empresa';
ALTER TABLE silver_empresas ALTER COLUMN desc_porte_empresa COMMENT 'Descrição do Porte da Empresa';
ALTER TABLE silver_empresas ALTER COLUMN ente_federativo_responsavel COMMENT 'Ente Federativo Responsável';
```

Verifique no **Catalog Explorer** se a descrição da tabela e os comentários das colunas realmente foram adicionados.

Exercício 03.03 - Carregando Dados JSON

Dentro das pastinhas do repositório que você importou no início desse doc, execute o arquivo **lab00_carga_csv** dentro do diretório **lab03.03 - ingerir socios json.**

Exercício 03.04 - Criando Streaming Tables

Crie a tabela de streaming que irá carregar as informações dos sócios da empresa:

```
CREATE OR REFRESH STREAMING TABLE bronze_socios(
 cnpj_basico BIGINT COMMENT 'Número do CNPJ raiz de 8 posições',
 identificador_socio BIGINT COMMENT 'Código de identificação do sócio',
 nome_socio STRING COMMENT 'Nome do sócio',
 cpf_cnpj_socio STRING COMMENT 'CPF ou CNPJ do sócio',
 qualificacao_socio BIGINT COMMENT 'Código de qualificação do sócio',
 data_entrada_sociedade STRING COMMENT 'Data de entrada do sócio na sociedade',
 pais_socio BIGINT COMMENT 'Código do país do sócio, se estrangeiro',
 representante_legal STRING COMMENT 'Nome do representante legal',
 faixa_etaria BIGINT COMMENT 'Faixa etária do sócio',
 _rescued_data STRING
)
SCHEDULE CRON '0/1 0/1 * ? * * *'
AS SELECT * FROM STREAM
read_files('/Volumes/workshops_databricks/db_workshop_SEUNOME_OU_CHAVE/landing/socios/socios*', format => 'json')
```

Lembrem-se de alterar o caminho do volume para o volume informado pelo instrutor.

Dentro das pastinhas do repositório que você importou no início desse doc, execute o arquivo **lab00_carga_csv** dentro do diretório **lab03.04 - ingerir novos socios json.**

Verifique se a tabela de streaming recebeu os novos dados.

Exercício 03.06 - Criando Materialized Views

Nesta etapa vamos criar a view materializada que conterá algumas agregações que utilizaremos futuramente nos exercícios posteriores. Copiem o código abaixo e executem-no no SQL Editor.

```
CREATE OR REPLACE MATERIALIZED VIEW gold_analise_empresarial
SCHEDULE CRON '0 0 0 * * ? *'
AS
SELECT
 c.descricao AS setor_economico,
 e.endereco_unidade_federativa AS uf,
 pe.desc_porte_empresa AS porte_empresa,
 COUNT(DISTINCT e.cnpj_basico) AS qtd_empresas,
 AVG(s.num_socios) AS media_socios_por_empresa,
 SUM(CASE WHEN s.pais_socio != 1058 THEN 1 ELSE 0 END) / COUNT(DISTINCT
e.cnpj_basico) * 100 AS percentual_socios_estrangeiros,
 AVG(CAST(e.val_capital_social AS DOUBLE)) AS capital_social_medio,
 COLLECT_SET(n.descricao) AS principais_naturezas_juridicas
FROM
 silver_empresas e
JOIN
 bronze_cnae c ON e.cnae_principal = c.cod_cnae
JOIN
 bronze_porte_empresa pe ON e.desc_porte_empresa = pe.desc_porte_empresa
LEFT JOIN
 (SELECT cnpj_basico, COUNT(*) AS num_socios, MAX(pais_socio) AS pais_socio
  FROM bronze_socios
  GROUP BY cnpj_basico) s ON e.cnpj_basico = s.cnpj_basico
JOIN
 bronze_naturezas n ON e.cod_natureza_juridica = n.codigo
GROUP BY ALL
```

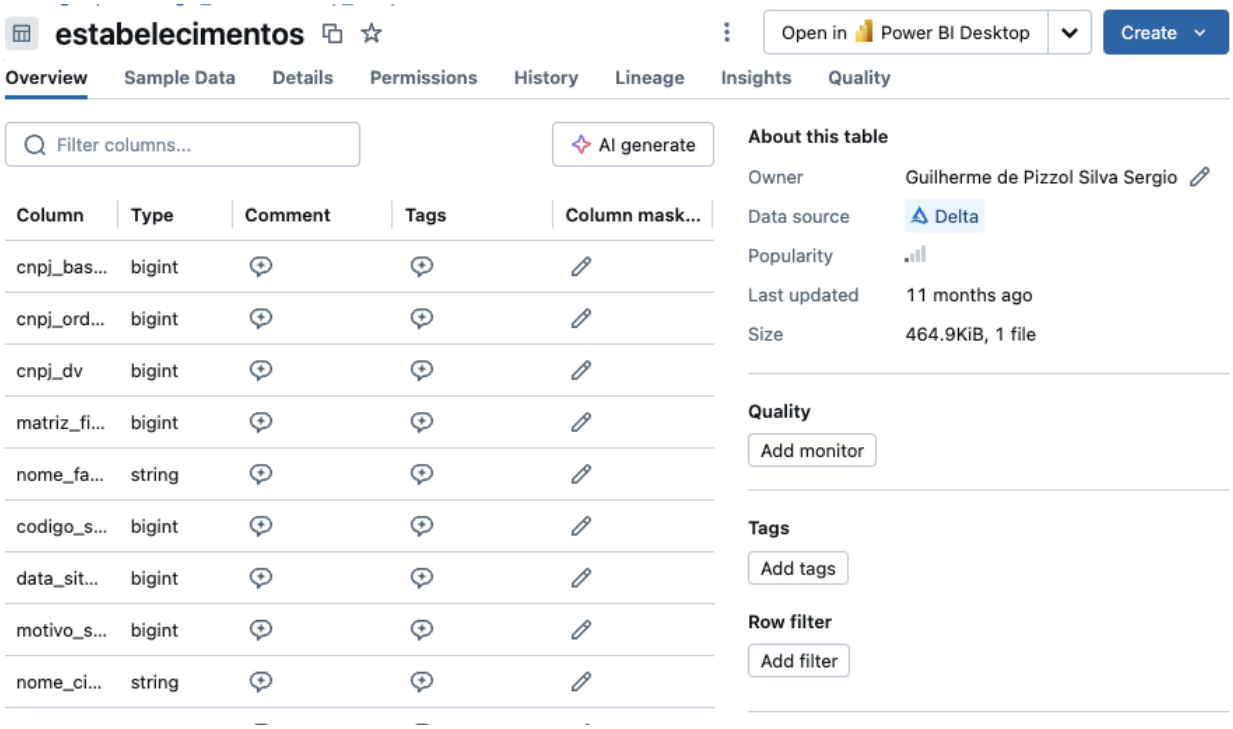
Hands-On Lab 04 - Filtrando e Mascarando linhas e colunas

Exercício 04.01 - Filtrando os registros

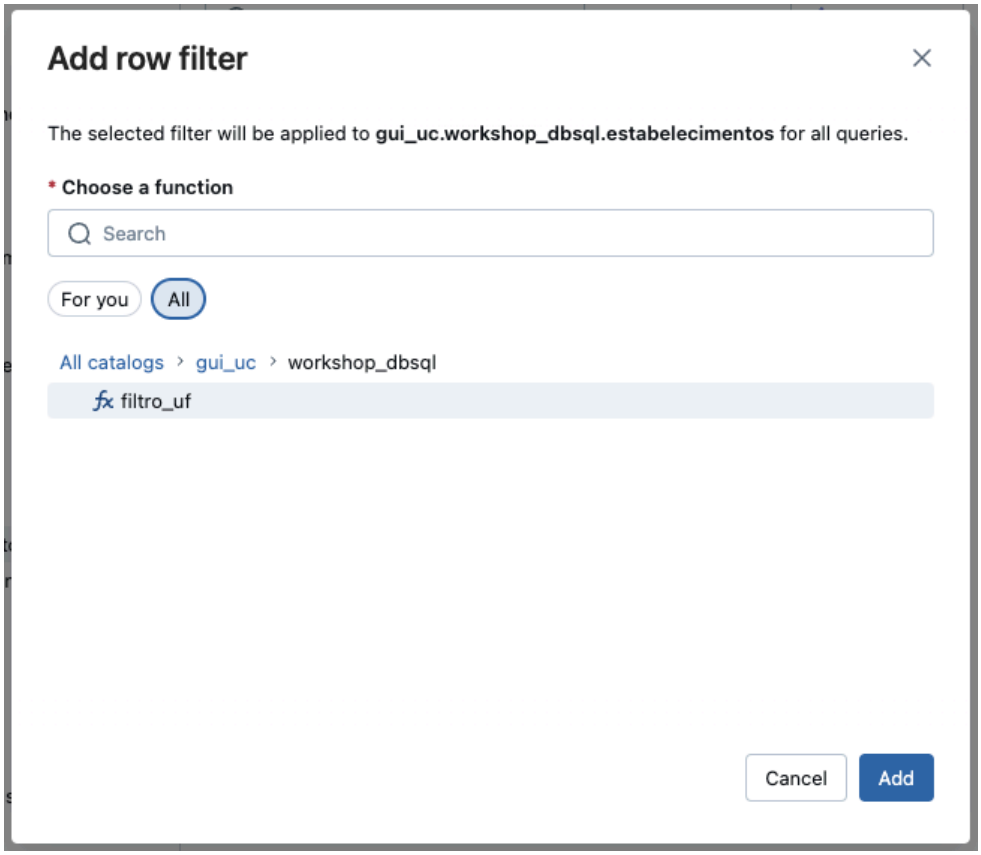
Para criarmos um filtro de linhas dentro do Databricks, é necessário criarmos uma função que irá dizer qual é a lógica por trás da filtragem das linhas, para tal, copie o código abaixo e execute-o em uma nova aba do SQL Editor:

```
CREATE OR REPLACE FUNCTION filtro_uf(uf STRING)
RETURN
is_account_group_member('admin') or
   exists (
     SELECT 1
     FROM silver_empresas
     WHERE is_account_group_member('grupo_1') AND endereco_unidade_federativa = uf
   )
```

Após a criação da função, existem duas formas de adicionar a função à uma tabela. Através da interface, como mostrado nas imagens abaixo:



Navegue até a tabela "bronze_estabelecimentos" e clique no botão 'Add filter' no canto inferior direito.



Selecione a função que você criou no passo anterior.

Add row filter

The selected filter will be applied to **gui_uc.workshop_dbsql.silver_empresas** for all queries.

* **Choose a function**

Search

For you | All

All catalogs › gui_uc › workshop_dbsql

✓ *fx* filtro_uf

Function definition

```
is_account_group_member('admin') or
    exists (
      SELECT 1
      FROM silver_empresas
      WHERE is_account_group_member('grupo_1') AND endereco_unidade_federativa
= uf
    )
```

uf

Select a column

Cancel | Add

Selecione a coluna que a função se baseia para filtrar os registros, neste caso, vamos utilizar a coluna 'uf'.

Outra maneira de adicionar o filtro é através do SQL, caso prefira, tente adicionar o filtro através do código:

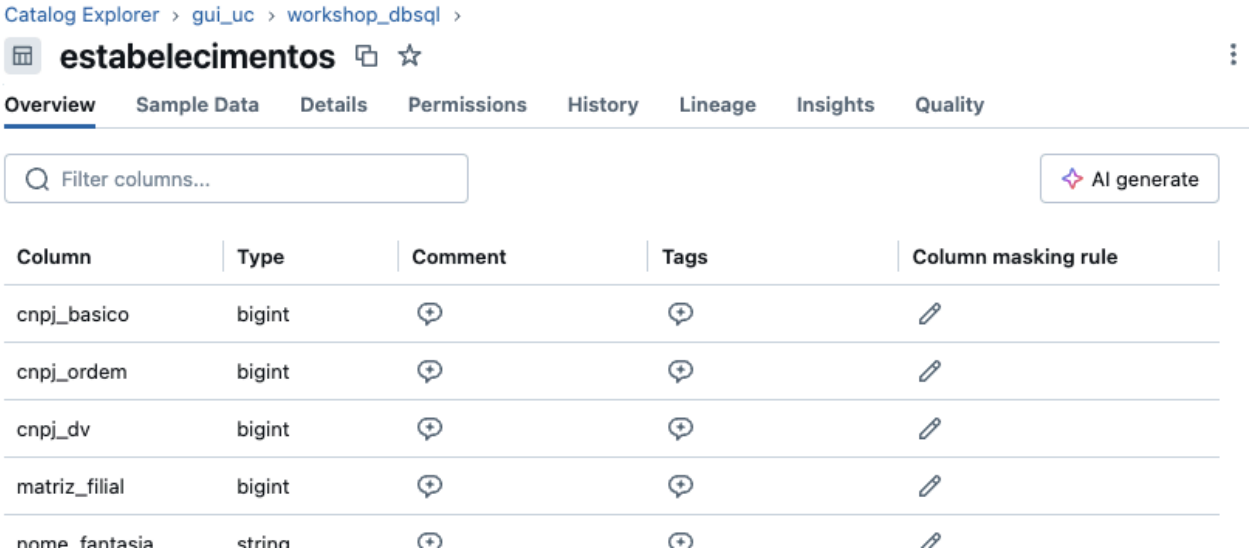
```
ALTER TABLE bronze_estabelecimentos SET ROW FILTER filtro_uf ON (uf);
```

Exercício 04.02 - Mascarando os registros

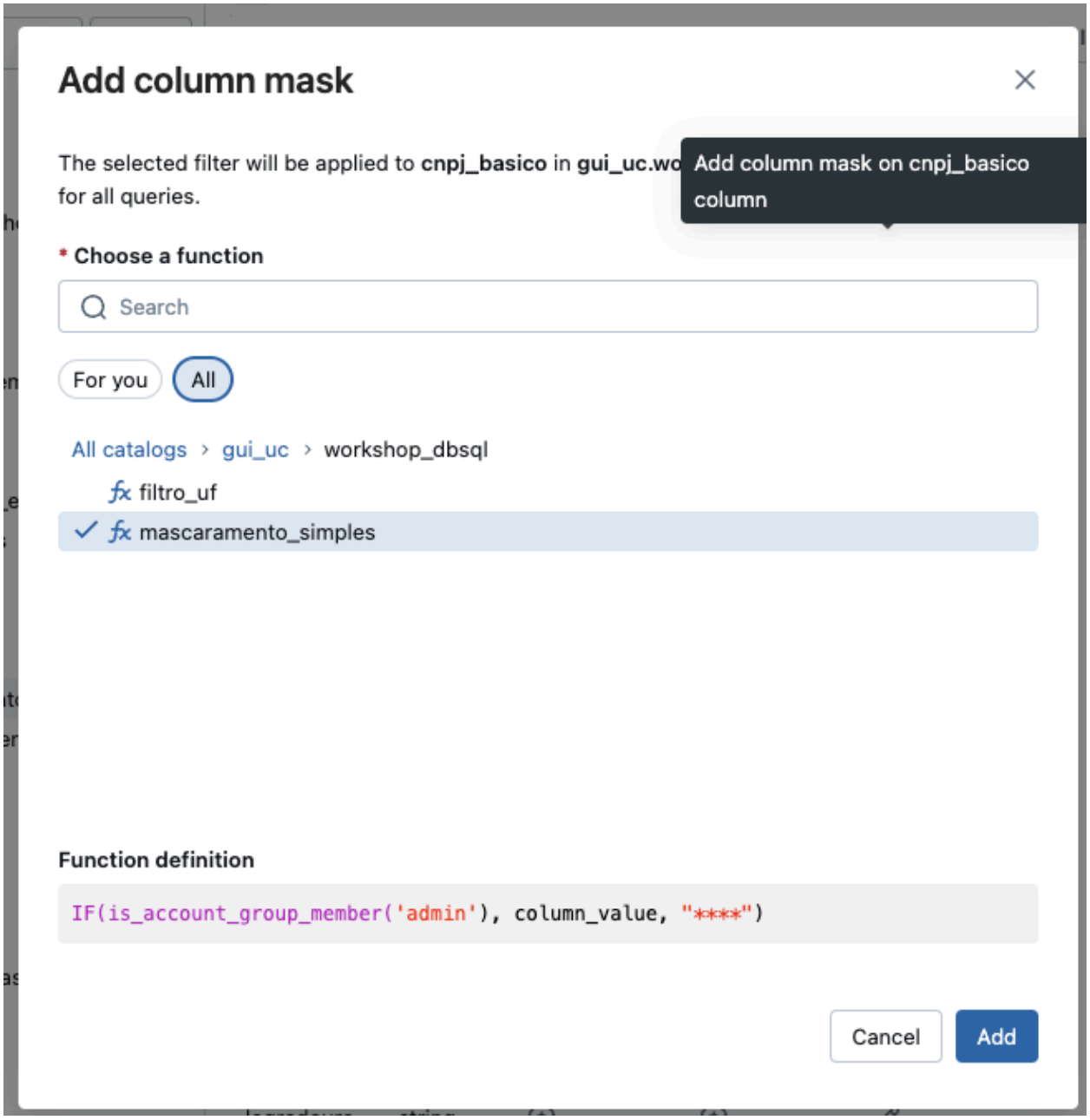
De forma bem semelhante, o mascaramento necessita de uma função por trás para definir quais colunas serão mascaradas. Crie a função de mascaramento com base no código abaixo:

```
CREATE OR REPLACE FUNCTION mascaramento_simples(column_value STRING)
RETURN
 IF(is_account_group_member('admin'), column_value, "****");
```

Também é possível adicionar a função de mascaramento, através da interface:



A última coluna na aba de overview de cada tabela contém a informação das funções de mascaramento que são aplicadas à coluna. Vamos adicionar uma nova regra na coluna cnpj_basico, para tal, clique no 'lápis' e selecione a função criada no passo anterior:



Teste o resultado final com uma query como "select * from bronze_estabelecimentos".

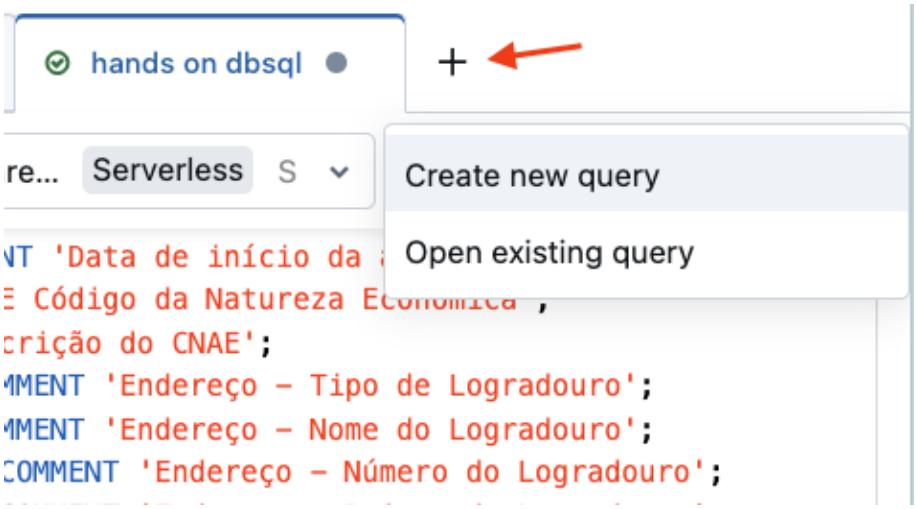
Hands-On Lab 05 - Criação de Alertas

Objetivo do exercício

O objetivo desse laboratório é explorar as funcionalidade de criação de um ALERTA

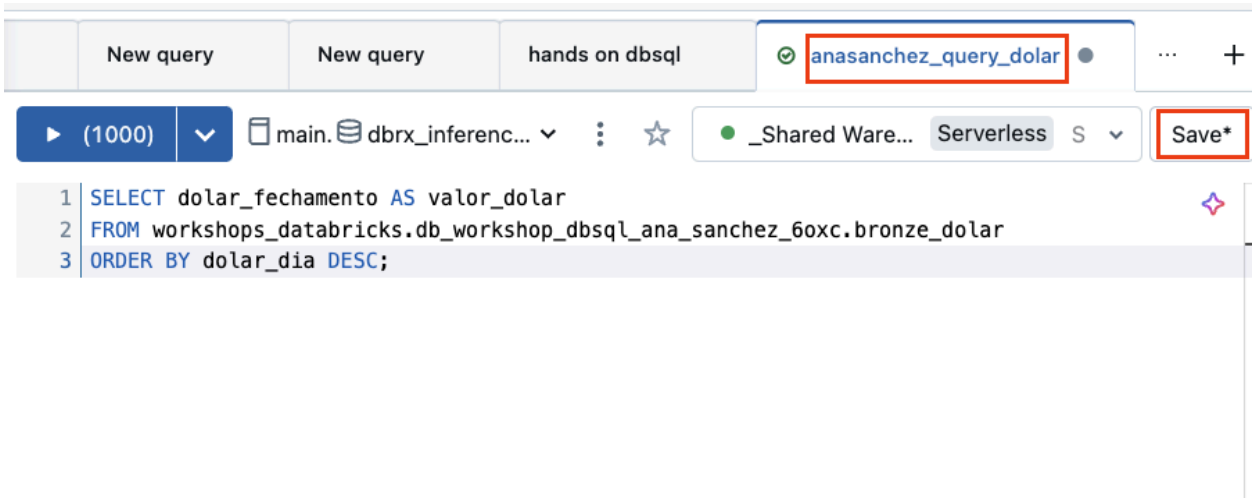
Exercício 05.01 - Criação da Query

Abra uma nova aba de Query:



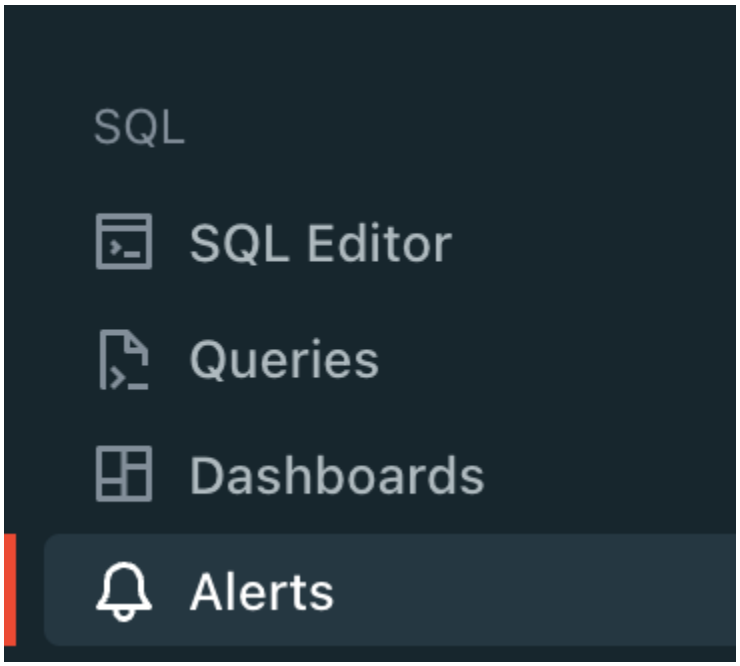
Adicione a seguinte query e salve ela como : SEUNOME_query_dolar

```
SELECT dolar_fechamento AS valor_dolar
FROM
workshops_databricks.db_workshop_dbsql_SEU_NOME_ou_SUA_CHAVE.bronze_dolar
ORDER BY dolar_dia DESC;
```



Exercício 05.02 - Criando o Alerta

Vamos utilizar a opção do menu "ALERT".



Clique no botão CREATE e configure o ALERTA conforme abaixo, com o seguinte nome: SEUNOME_alerta_dolar e após preencher todos os campos clique em Preview alert

Workspace › Users › ana.sanchez@databricks.com

Edit alert

Alert name

anasanchez_alerta_dolar

Query

anasanchez_query_dolar

Trigger condition

Value column

valor_dolar | First row

Operator

>

Threshold value

5

When query result has no rows, set state to OK

OK This alert would not trigger with the current data

valor_dolar	Threshold
4.786	5

Notifications

When alert is triggered

Send notification just once until back to normal.

When alert returns back to normal

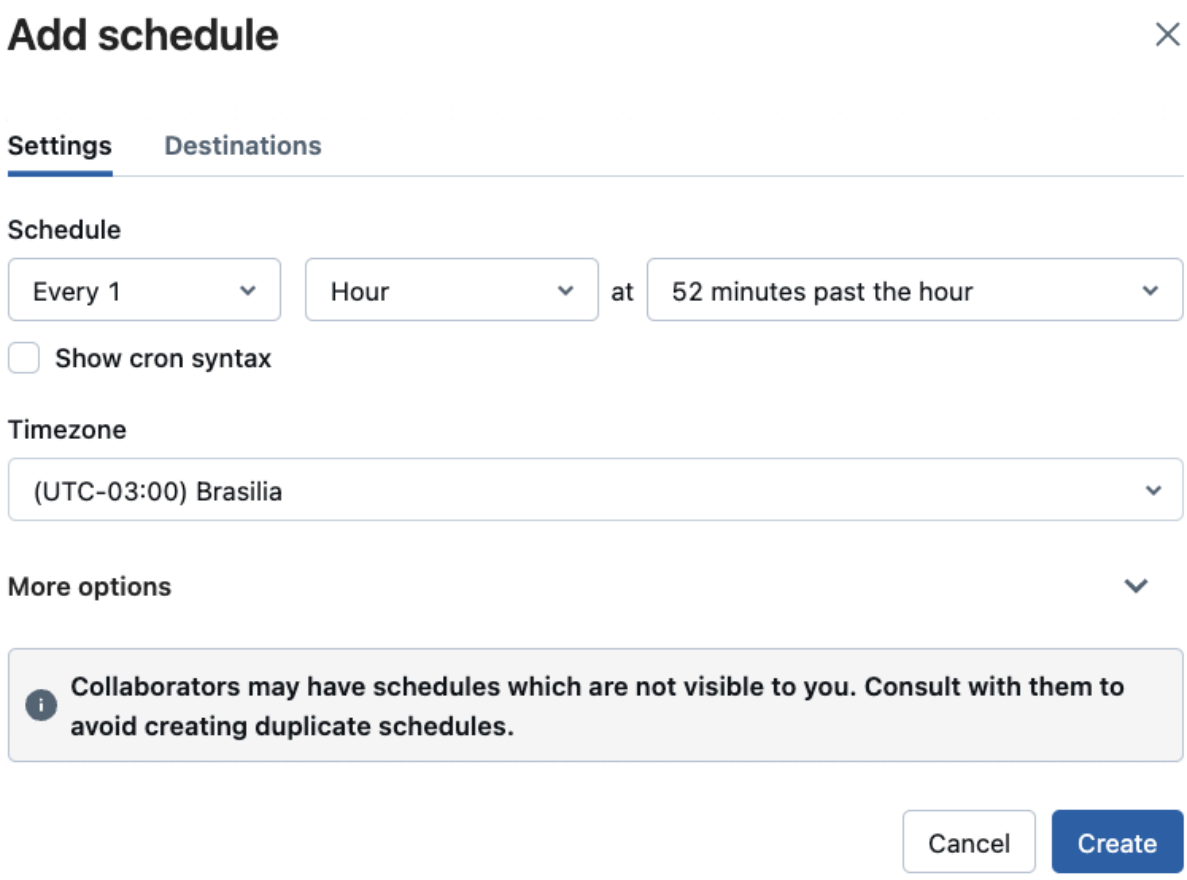
☑ Send notification

Template

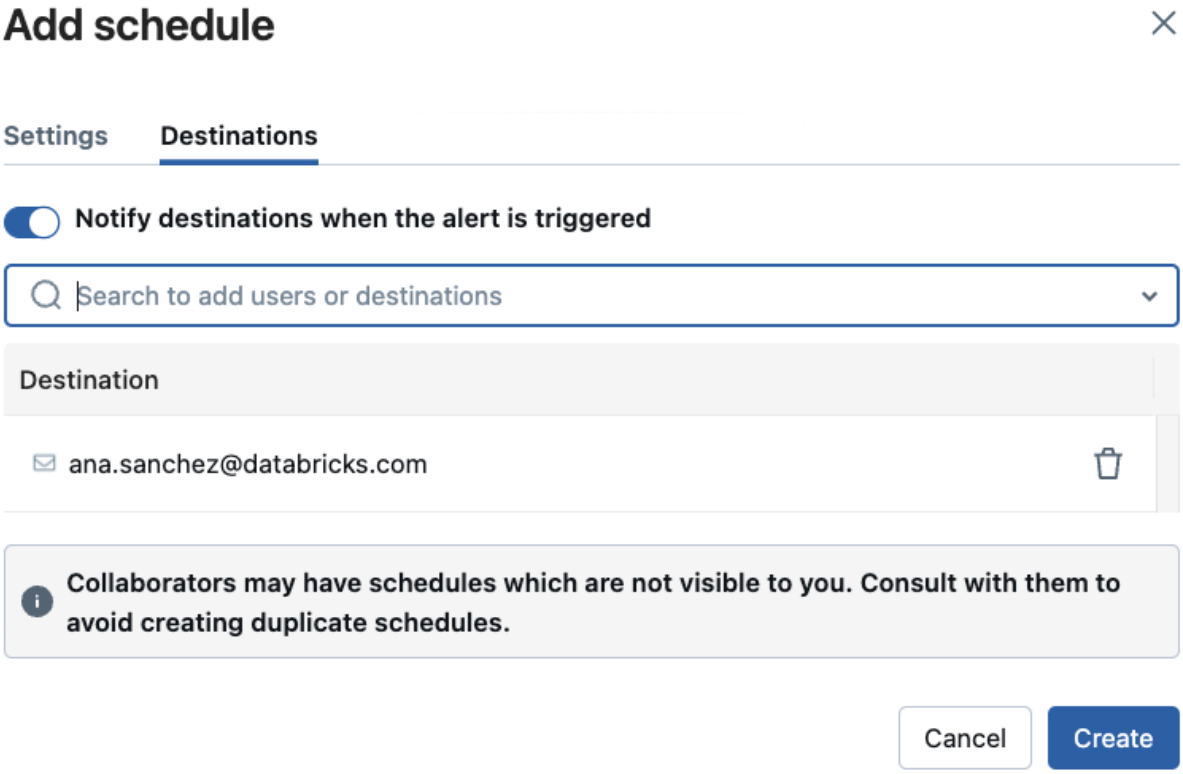
Use default template

Adicionar um e-mail para quando o alerta dispalar, para isso clique em **Add schedule.**

Deixe os valores default pois depois da execuçaõ manual iremos desligar.

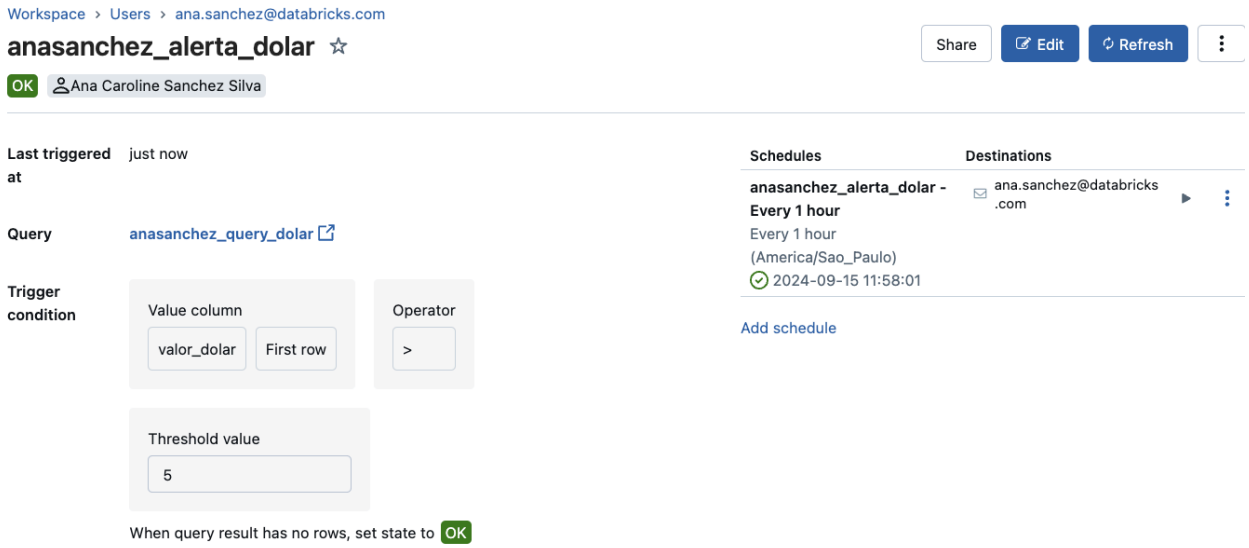


Vá para a aba **Destinations,** adicione o seu email e clique em **Create:**

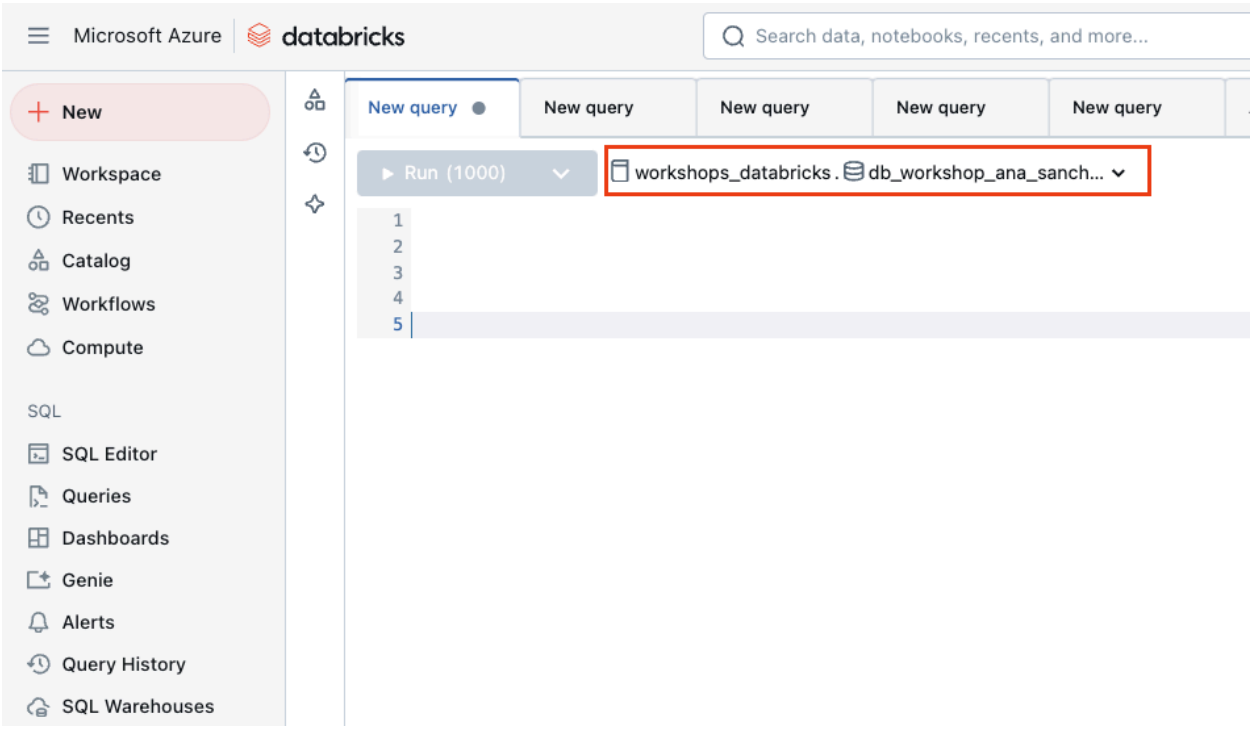


Clique em ▶ .

Qualdo terminar você vai ver o status do alerta abaixo do nome dele:



Agora vamos inserir um valor de dolar para um novo dia, com um valor maior que 5 para ver se o alerta vai disparar quando o executarmos, para isso abra uma nova aba o **SQL Editor** e selecione seu catálogo (**workshops_databricks**) e seu schema (**db_workshop_SEUNOME_OU_SUACHAVE**) no topo do Query Editor:



Crie uma query para adicionar um dia no maior dia da tabela **bronze_dolar** no campo **dolar_dia**

```
SELECT
   MAX(dolar_dia) ultimo_dia,
   to_date(DATEADD(day, 1, MAX(dolar_dia))) proximo_dia
FROM bronze_dolar;
```

```sql
SELECT
    MAX(dolar_dia) ultimo_dia,
    to_date(DATEADD(day, 1, MAX(dolar_dia))) proximo_dia
FROM bronze_dolar;
```

Raw results +

	ultimo_dia	proximo_dia
1	2023-06-30	2023-07-01

Agora insira os seguinte valores na tabela bronze:

- Dolar_dia: calcule o ultimo dia da tabela e some com 1
- Dolar_mes: pegue o mês do campo calculado anteriormente
- Dolar_ano: pegue o ano do campo calculado anteriormente
- Dolar_fechamento: 5.12
- Dolar_abertura: pegue o dolar de fechamento do dia anterior
- Dolar_max: 5.22
- Dolar_min: 4.72

```sql
INSERT INTO bronze_dolar
   (dolar_dia, dolar_mes, dolar_ano, dolar_fechamento, dolar_abertura, dolar_max, dolar_min)
SELECT
   to_date(DATEADD(day, 1, MAX(dolar_dia))),
   month(DATEADD(day, 1, MAX(dolar_dia))),
   year(DATEADD(day, 1, MAX(dolar_dia))),
   5.12,
   (SELECT dolar_fechamento FROM bronze_dolar ORDER BY dolar_dia DESC LIMIT 1),
   5.22,
   4.72
FROM bronze_dolar;
```

Verifique o resultado:

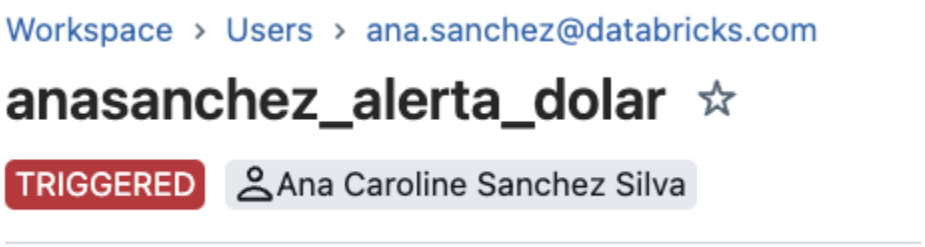
```sql
SELECT *
FROM workshops_databricks.db_workshop_dbsql_ana_sanchez_6oxc.bronze_dolar
```

```
ORDER BY dolar_dia DESC limit 1;
```

Raw results +

	dolar_ano	dolar_mes	dolar_dia	dolar_fechamento	dolar_abertura	dolar_max	dolar_min
1	2023	7	2023-07-01	5.12	4.786	5.22	4.72

Agora volte para o seu alerta e o execute ▶ :



Verifique seu email:

Databricks SQL alert: anasanchez_alerta_dolar (TRIGGERED)

External Inbox ×

Microsoft Azure <azure-noreply@microsoft.com> 12:53 PM (1 minute ago)

to me

Microsoft Azure Azure Databricks

Databricks SQL alert: anasanchez_alerta_dolar (TRIGGERED)

You're receiving this email because ana.sanchez@databricks.com defined an alert and added you as a recipient.

anasanchez_alerta_dolar

Review the alert details.

Review the alert >

You can also view the query.

Databricks SQL

Give feedback - Privacy policy

Did you find this email helpful? Yes No

Privacy Statement

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

Microsoft

Reply Forward

Hands-On Lab 06 - DBSQL API

Objetivo do exercício

O objetivo desse laboratório é realizar queries SQL no Databricks SQL através de sua API REST.

Exercício 06.01 - Testes com o DBSQL API

1. Acesse ferramentas como Postman ou Insomnia.

2. Importe a collection de consultas de APIs em sua ferramenta: dbsql-api/postman_files/Databricks SQL Execution API.postman_collection.json at main · anasanchezss9/dbsql-api (github.com)

3. Importe o arquivo de environment em sua ferramenta:dbsql-api/postman_files/Databricks Environment.postman_environment.json at main · anasanchezss9/dbsql-api (github.com)

4. Siga os passos abaixo do link a seguir para configurar o arquivo de environment e depois realizar os testar com a API: dbsql-api/README.md at main · anasanchezss9/dbsql-api (github.com)

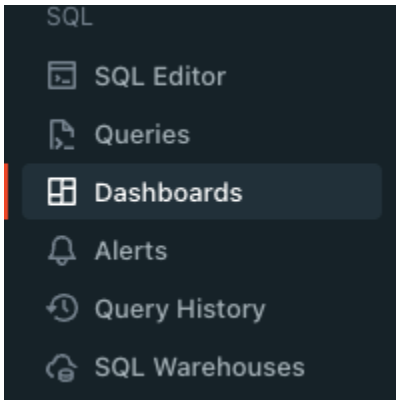
Hands-On Lab 07 - Criando Dashboards

Objetivo do exercício

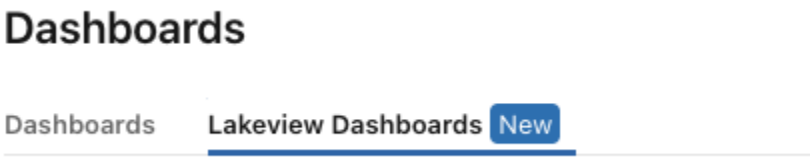
O objetivo desse laboratório é explorar e entender como funciona a nova ferramenta Lakeview Dashboard.

Exercício 07.01 - Criando o dashboard

Acesse o menu "Dashboards" na aba lateral esquerda.



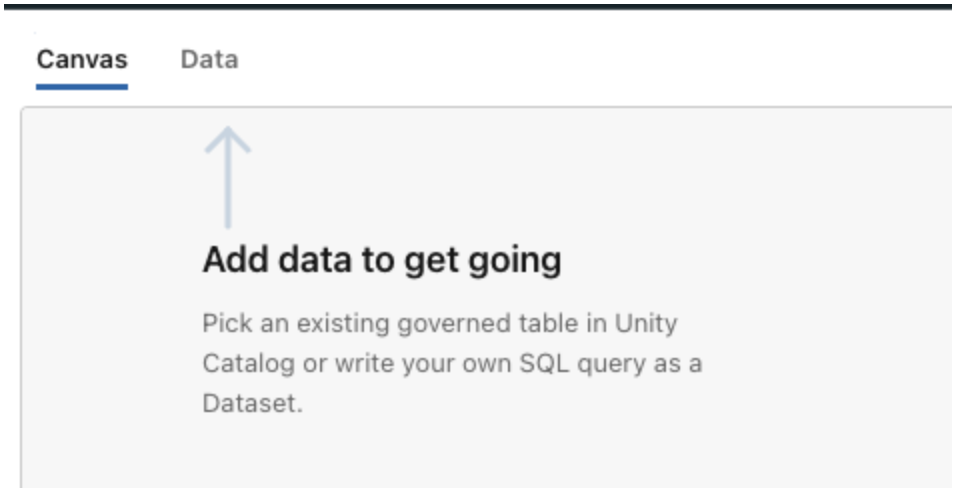
Selecione a aba "Dashboards"



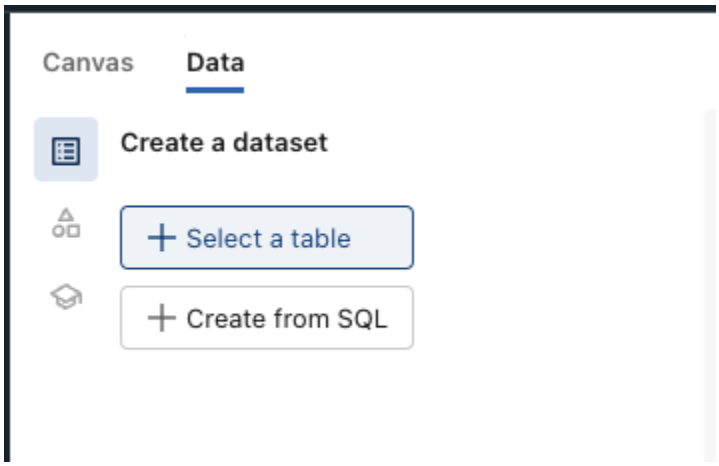
Por último clique no botão "Create Lakeview dashboard" localizado no canto superior direito

Exercício 07.02 - Criando o dataset

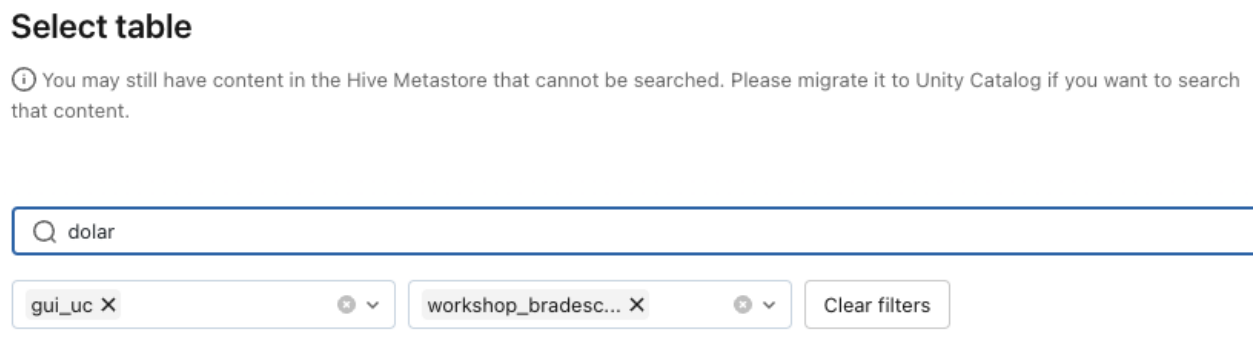
Clique na aba "Data" conforme exibido na figura



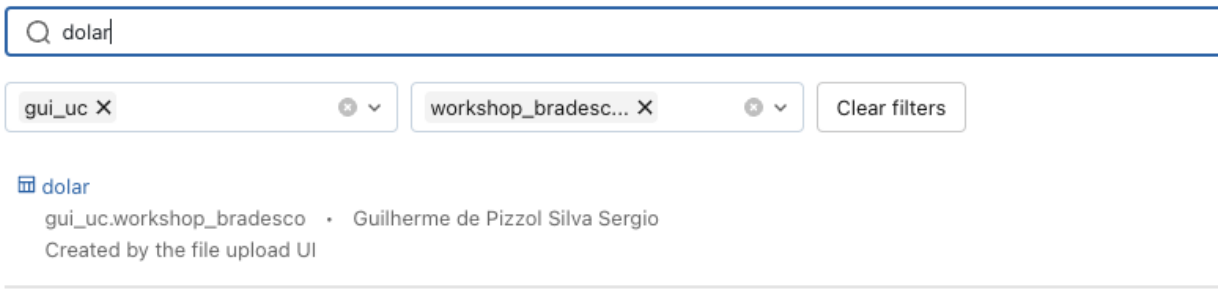
Em seguida clique no botão "+ Select a table"



Filtre pela tabela "dolar" adicionando o filtro de catálogo e schema caso necessário (pressione Enter para exibir o resultado da busca)

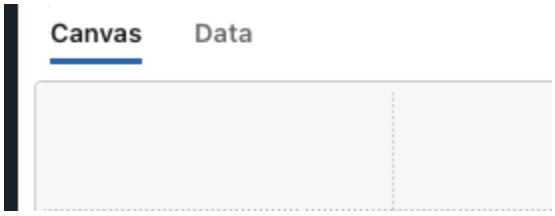


Selecione a tabela "dolar" que criamos no laboratório anterior, conforme mostrado na imagem abaixo:

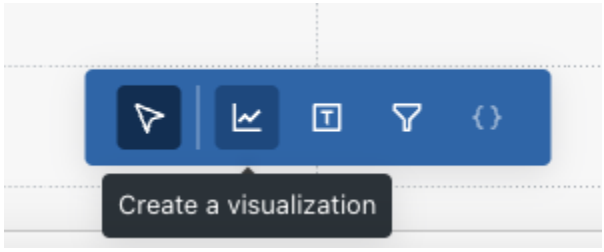


Exercício 07.03 - Criando visualizações no Canvas

Volte para a aba "Canvas"

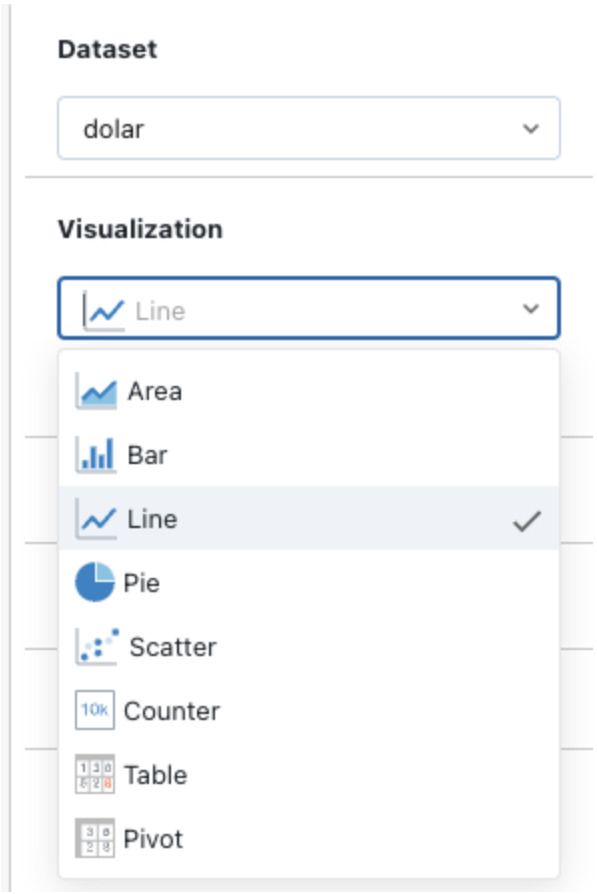


Clique no botão "Create a visualization" conforme mostrado na imagem abaixo:

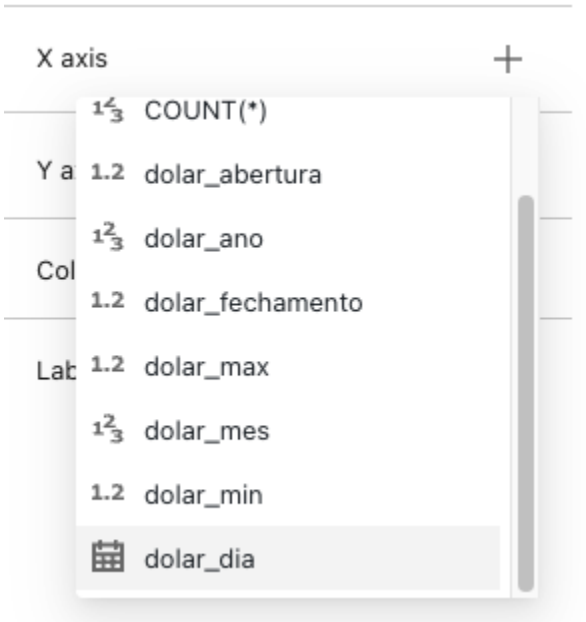


Em seguida selecione a área que deseja criar a visualização

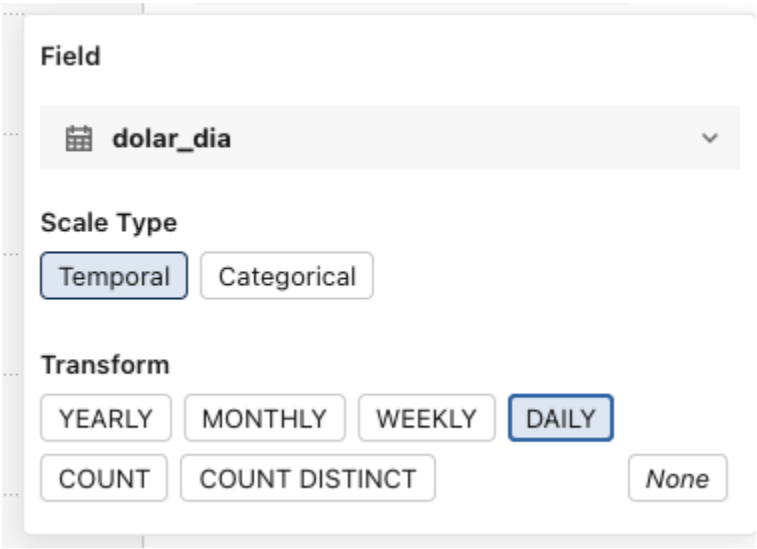
Na aba lateral direita, verifique se o dataset "dolar" já está selecionado, e se assim estiver, selecione a opção "Line" nos tipos de visualizações



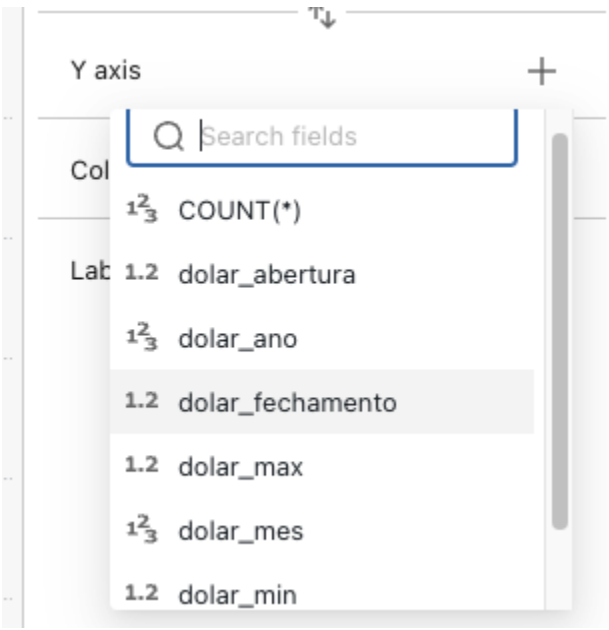
Para o eixo X selecione a coluna "dolar_dia" conforme a imagem abaixo



Clique na agregação MONTHLY(dolar_dia) e mude para DAILY

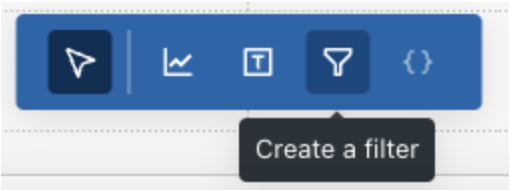


Para o eixo Y selecione a coluna "dolar_fechamento"

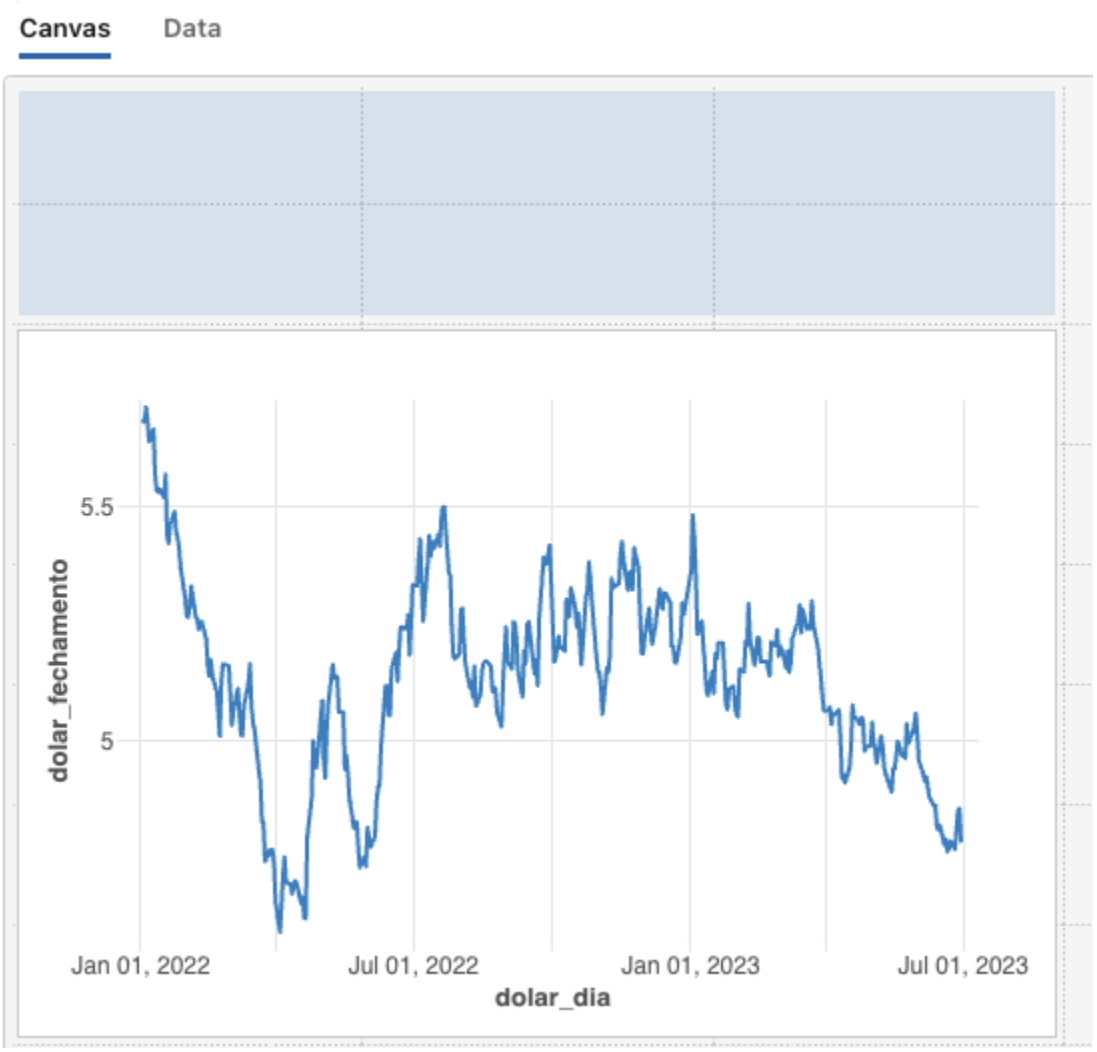


Exercício 07.04 - Criando Filtros

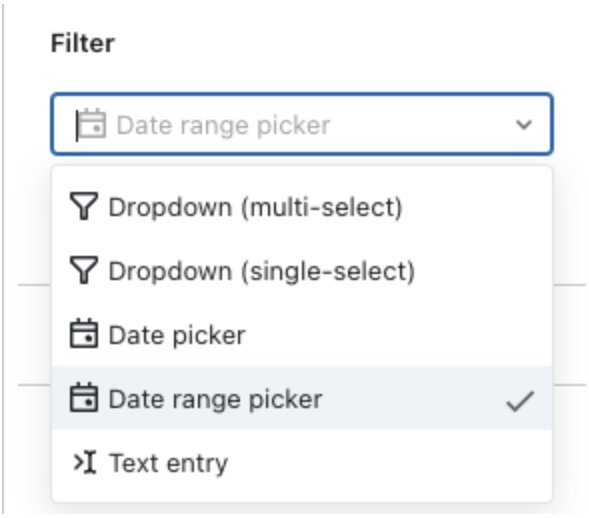
No menu central clique no botão "Create a filter"



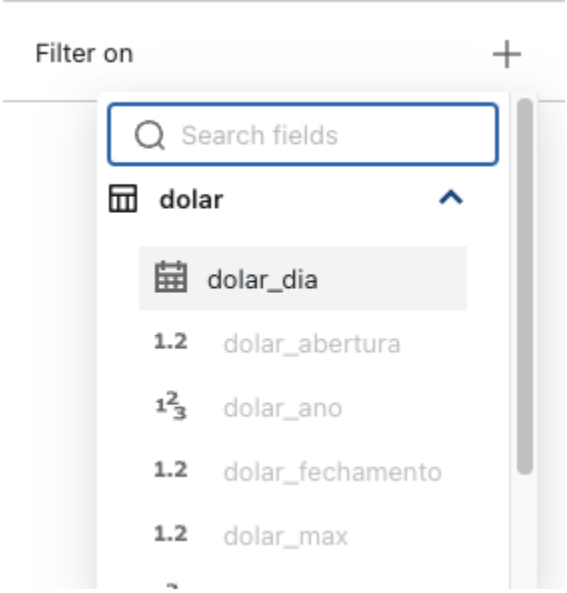
Posicione o filtro como desejar



No menu lateral direito selecione como tipo de filtro "Date range picker"

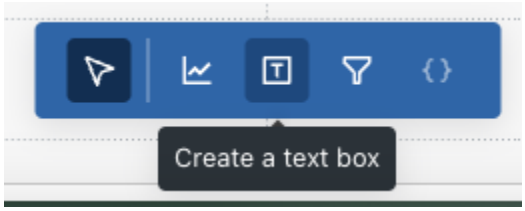


Clique no "+" da opção Filter on e selecione a coluna "dolar_dia"

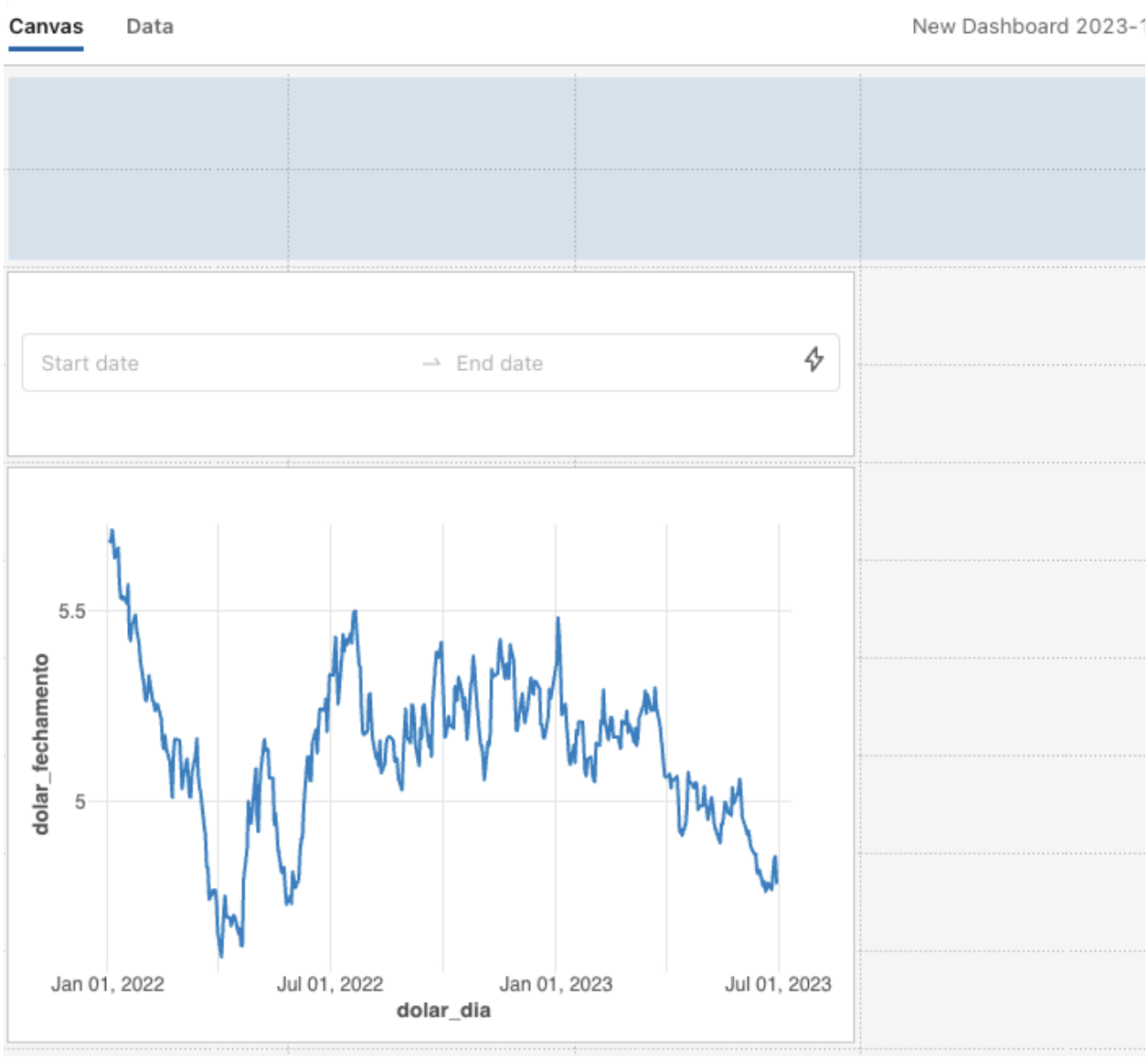


Exercício 07.05 - Adicionando Cabeçalhos

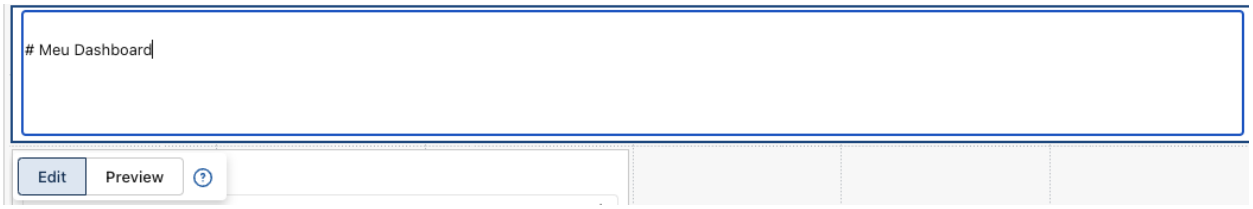
No menu central clique na opção "Create a text box"



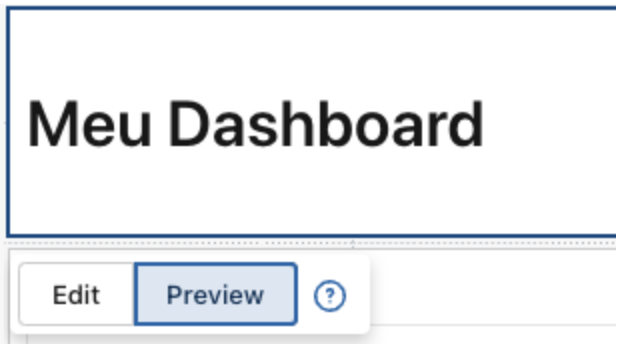
Posicione a caixa de texto como desejar



Na caixa de texto, escreva um título para seu dashboard utilizando a notação de markdown como base (lembre-se que é possível de adicionar links e imagens a este campo de texto para deixar seus dashboards ainda mais bacanas!)



Clique em "Preview" para pré-visualizar o resultado



Quando estiver satisfeito, nomeie seu dashboard e compartilhe com seus colegas.

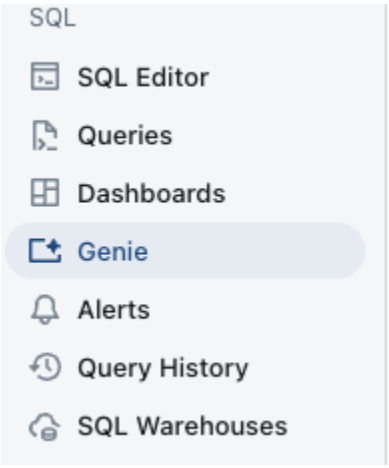
Hands-On Lab 08 - Conversando com os dados com o AI/BI Genie

Objetivo do exercício

O objetivo desse exercício é demonstrar as capacidades do AI/BI Genie para obter resultados com queries SQL utilizando linguagem natural através da sua interface.

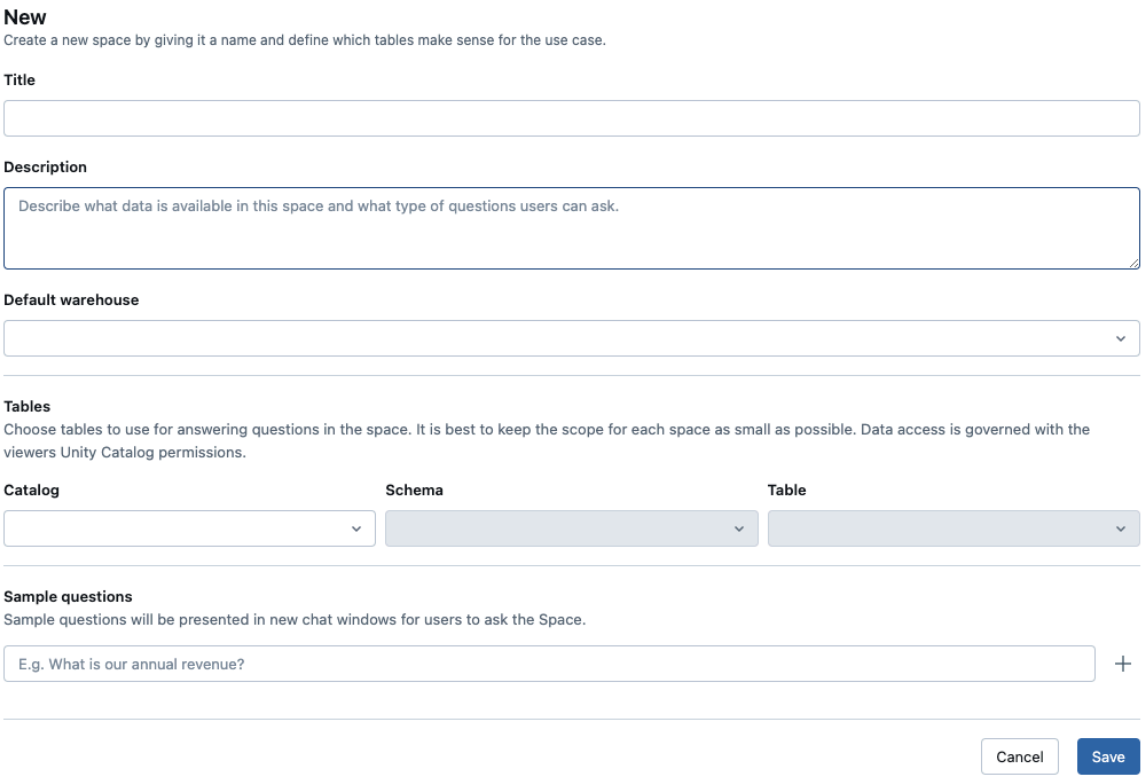
Exercício 08.01 - Criando e configurando seu espaço

O primeiro passo para criar um novo espaço é acessar o menu lateral e clicar no botão 'Genie', como na imagem abaixo:

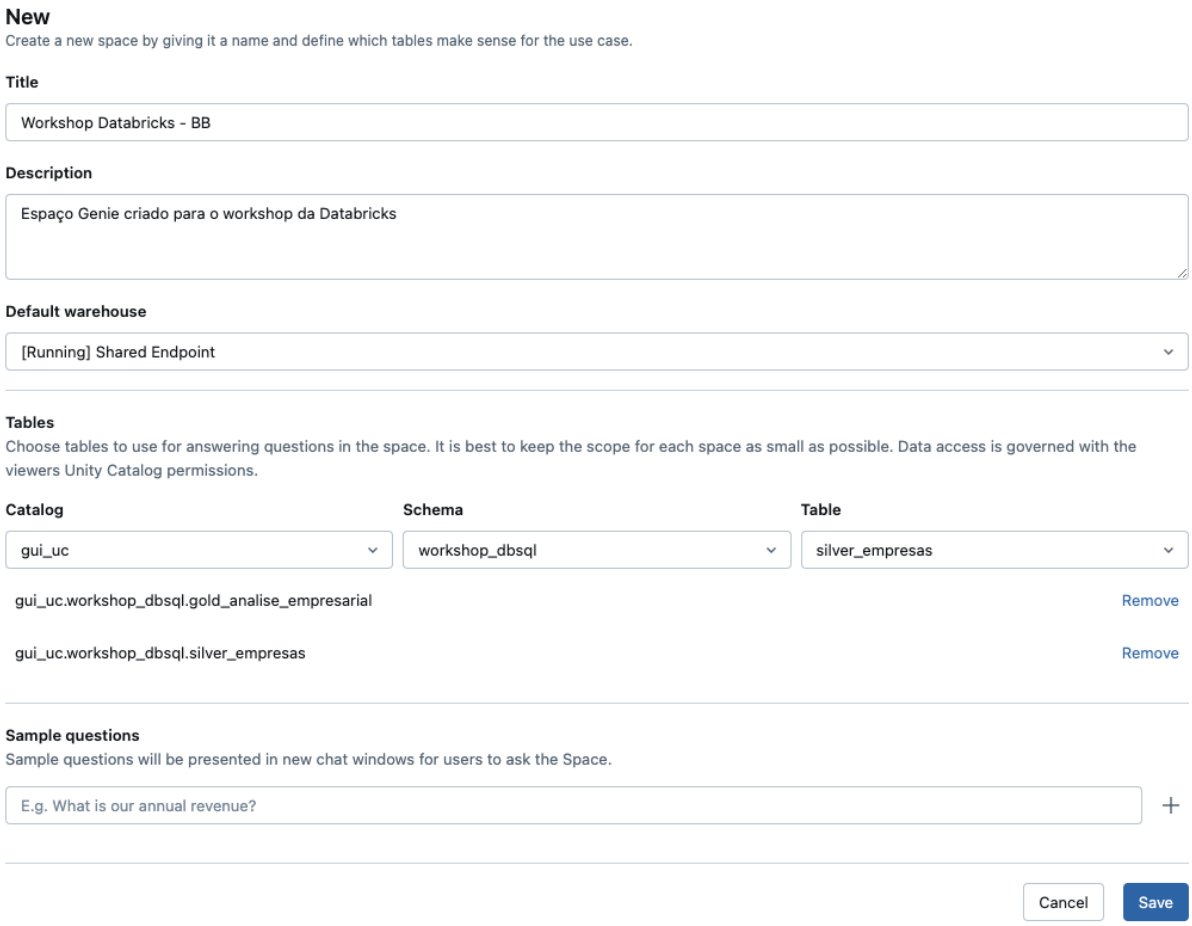


Feito isso, clique no botão 'New' no canto superior direito:

E comece a configurar seu espaço:

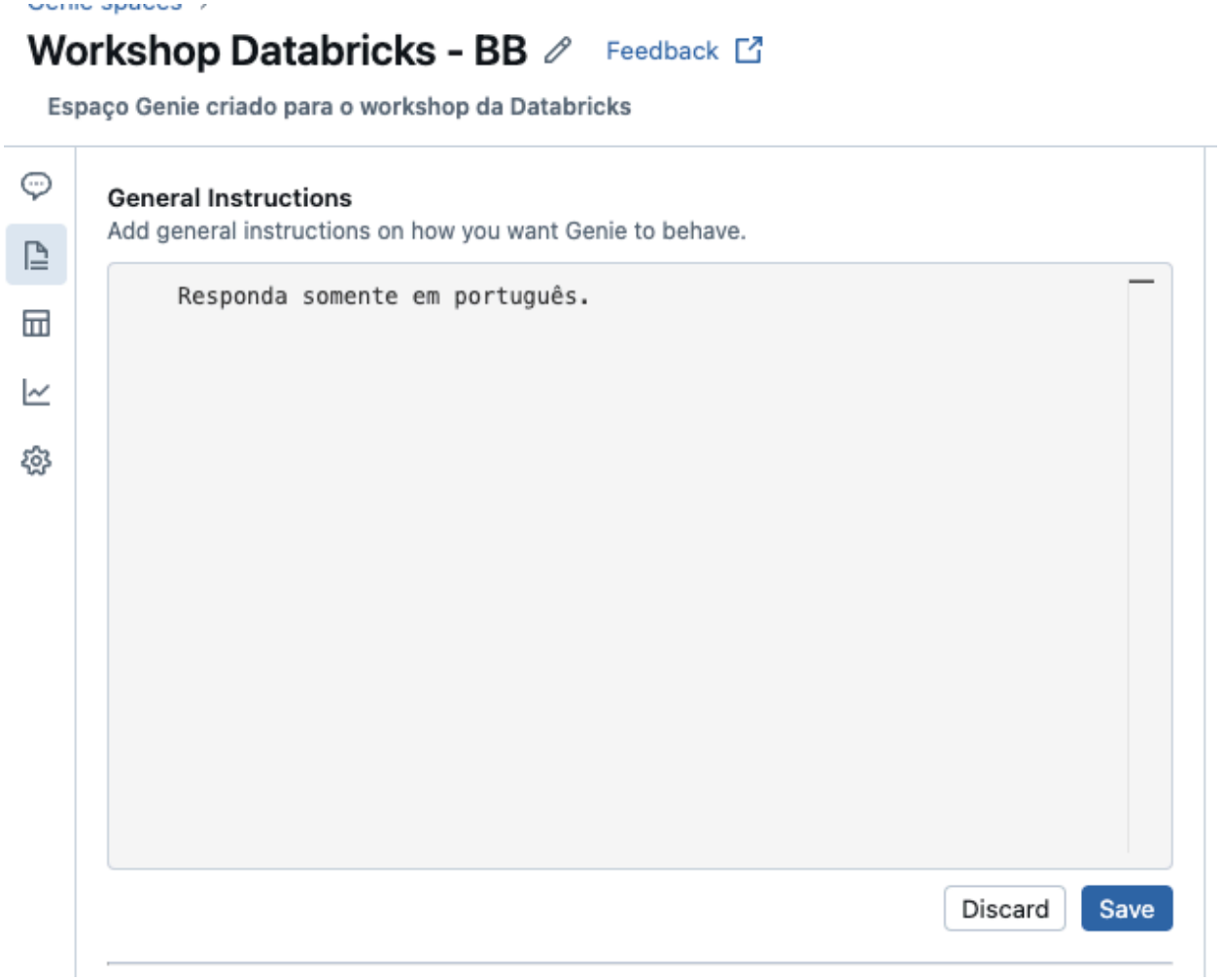


Nessa etapa vamos editar o nome do espaço, sua descrição, selecionar o Warehouse e adicionar as tabelas iniciais que farão parte da análise:



(Iremos utilizar apenas as tabelas gold_analise_empresarial e silver_empresas)

Por último, vamos adicionar uma instrução na aba de 'Instructions' como na imagem abaixo:



Fique à vontade para explorar o espaço criado fazendo perguntas na aba 'Chat':