

VOCÊ JÁ SE PERGUNTOU COMO
SIMPLIFICAR A CONSULTA A
DIVERSOS BANCOS DE DADOS E
FORMATOS DE ARQUIVOS SEM
COMPLICAÇÕES?

O TRINO É A RESPOSTA!



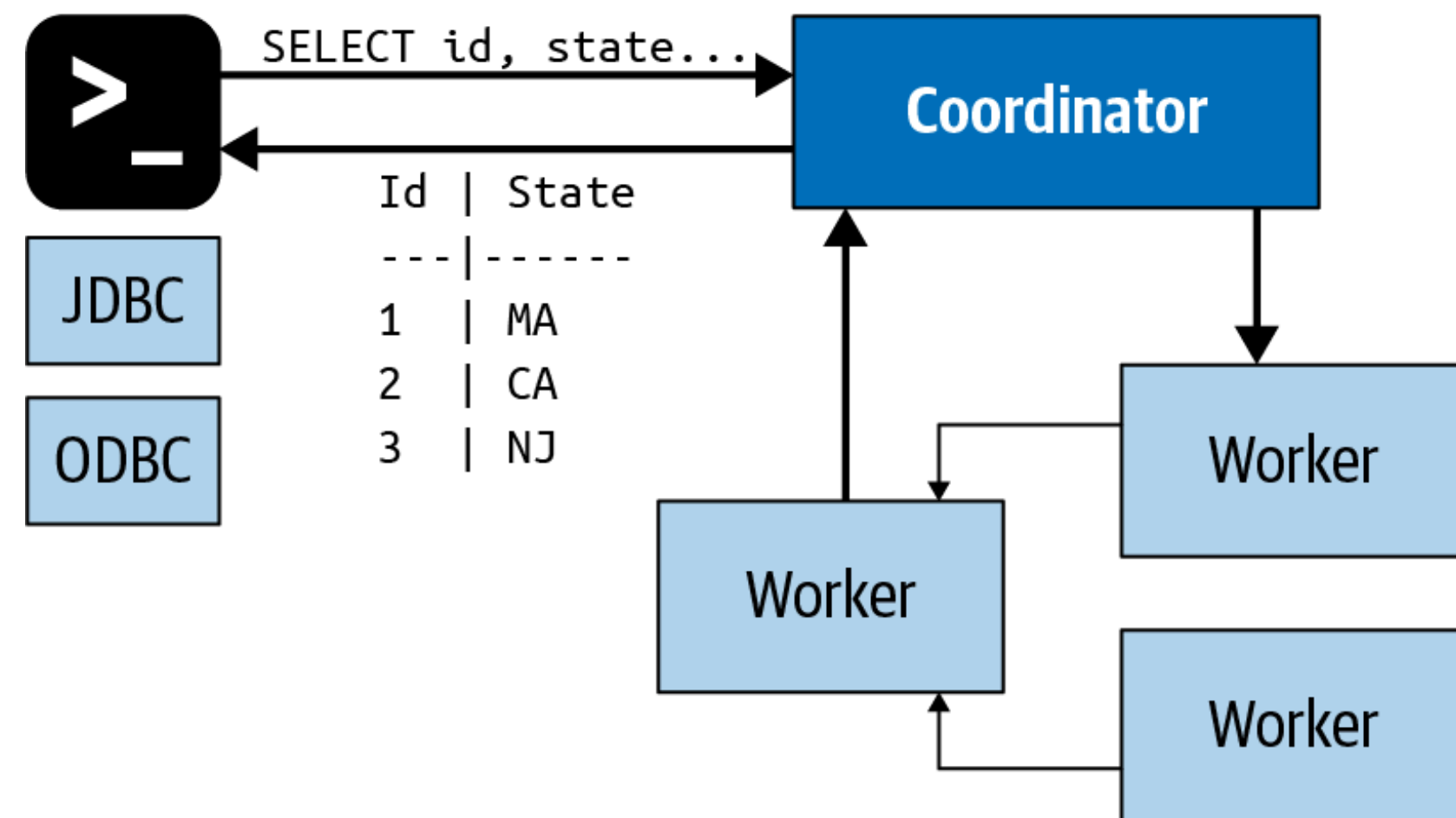
trino





Trino é um mecanismo de consulta SQL distribuído de código aberto, projetado para análise interativa em várias fontes de dados heterogêneas

- 1. Velocidade:** O Trino é altamente paralelo e distribuído, otimizado para consultas de baixa latência e eficientes.
- 2. Escala:** Organizações de grande porte usam o Trino para consultar datalakes com exabytes de dados e enormes datawarehouses.
- 3. Simplicidade:** O Trino é compatível com SQL ANSI e funciona com ferramentas de BI como R, Tableau, Power BI e Superset.
- 4. Versatilidade:** Suporta diversos casos de uso.
- 5. Análise no local:** Você pode consultar dados diretamente em sistemas como Hadoop, S3, Cassandra, MySQL e outros, sem a necessidade de processos complexos e lentos de cópia de dados.
- 6. Execução em qualquer lugar:** O Trino é otimizado para ambientes locais e em nuvem, incluindo Amazon, Azure, Google Cloud e outros.
- 7. Confiável:** Grandes organizações usam o Trino para operações comerciais críticas, incluindo resultados financeiros para mercados públicos.



Como Funciona

- 1. Envio da Consulta:** O usuário envia uma consulta SQL para o coordinator.
- 2. Planejamento:** O coordinator analisa e planeja a consulta, dividindo-a em várias etapas.
- 3. Distribuição:** O coordinator distribui as tarefas entre os workers.
- 4. Execução:** Os workers executam as tarefas e processam os dados.
- 5. Coleta de Resultados:** O coordinator coleta os resultados dos workers e os retorna ao usuário.

Essa arquitetura distribuída permite que o Trino execute consultas complexas de forma rápida e eficiente, mesmo em grandes volumes de dados.

Se precisar de mais detalhes ou tiver outras perguntas, estou à disposição!

Arquivo docker-compose com os serviços necessários

```
8  minio:
12  ports:
14    - "9001:9001"
15  command: server /data --console-address
16  #Não esquecer de comentar o volume na h
17  volumes:
18    - /mnt/d/volume/minio:/data
19  environment:
20    MINIO_ACCESS_KEY: minio
21    MINIO_SECRET_KEY: minio123
22  networks:
23    - roussenq
24
25  trino-coordinator:
26    image: trinodb/trino:latest
27    container_name: trino-coordinator
28    environment:
29      - TRINO_ENVIRONMENT=production
30    volumes:
31      - ./etc:/usr/lib/trino/etc
32      - ./catalog:/etc/trino/catalog
33    ports:
34      - "8080:8080"
35    depends on:
```

```
38  - roussenq
39
40  trino-worker:
41    image: trinodb/trino:latest
42    container_name: trino-worker
43    environment:
44      - TRINO_ENVIRONMENT=production
45      - TRINO_DISCOVERY_URI=http://trino-coordinator:8080
46    volumes:
47      - ./etc:/usr/lib/trino/etc
48      - ./catalog:/etc/trino/catalog
49    depends_on:
50      - trino-coordinator
51    networks:
52      - roussenq
53
```

```
54  hive-metastore:
55    image: 'apache/hive:4.0.0'
56    hostname: hive-metastore
57    container_name: hive
58    ports:
59      - '9083:9083'
60    volumes:
61      - ./conf/core-site.xml:/opt/hadoop/etc/hadoop/core-site.xml:ro
62      - ./conf/metastore-site.xml:/opt/apache-hive-metastore-4.0.0-bin/conf/meta
63    environment:
64      SERVICE_NAME: metastore
65      METASTORE_DB_HOSTNAME: minio
66      HIVE_AUX_JARS_PATH: /opt/hadoop/share/hadoop/tools/lib/hadoop-aws-3.3.6.ja
67    depends_on:
68      - minio
69    networks:
70      - roussenq
71
```

Arquivos de catalogo responsaveis pelo conexao do trino com os serviços

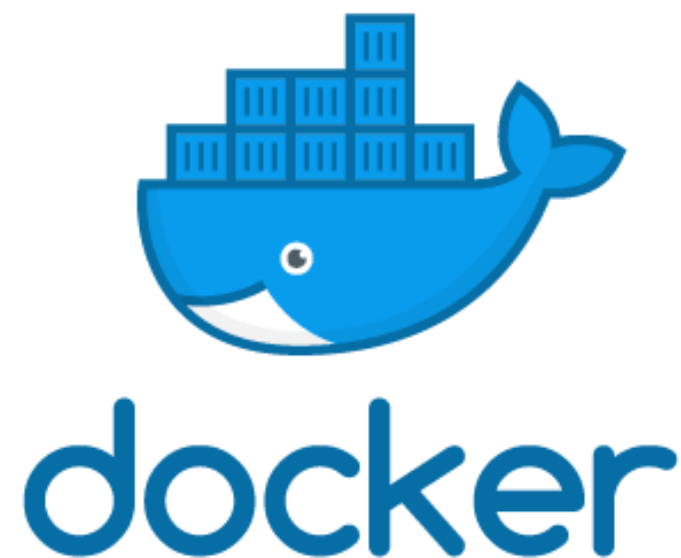


The screenshot shows the Trino Explorer interface. The top bar displays the title 'EXPLORER: TRINO [WSL:...' and several icons for file operations. Below the top bar, there are tabs for different property files: 'minio.properties', 'oracle.properties', 'postgresql.properties', and 'sqlserver.properties'. The 'catalog' directory is expanded on the left, showing a list of these property files. The 'sqlserver.properties' file is selected and its contents are displayed in the main editor area. The file contains four lines of configuration for connecting to a Microsoft SQL Server.

```
catalog > sqlserver.properties
1 connector.name=sqlserver
2 connection-url=jdbc:sqlserver://mssqlserver:1433;databaseName=master;encrypt=false
3 connection-user=sa
4 connection-password=SqlServer2019!
```


Demonstração

Software necessários

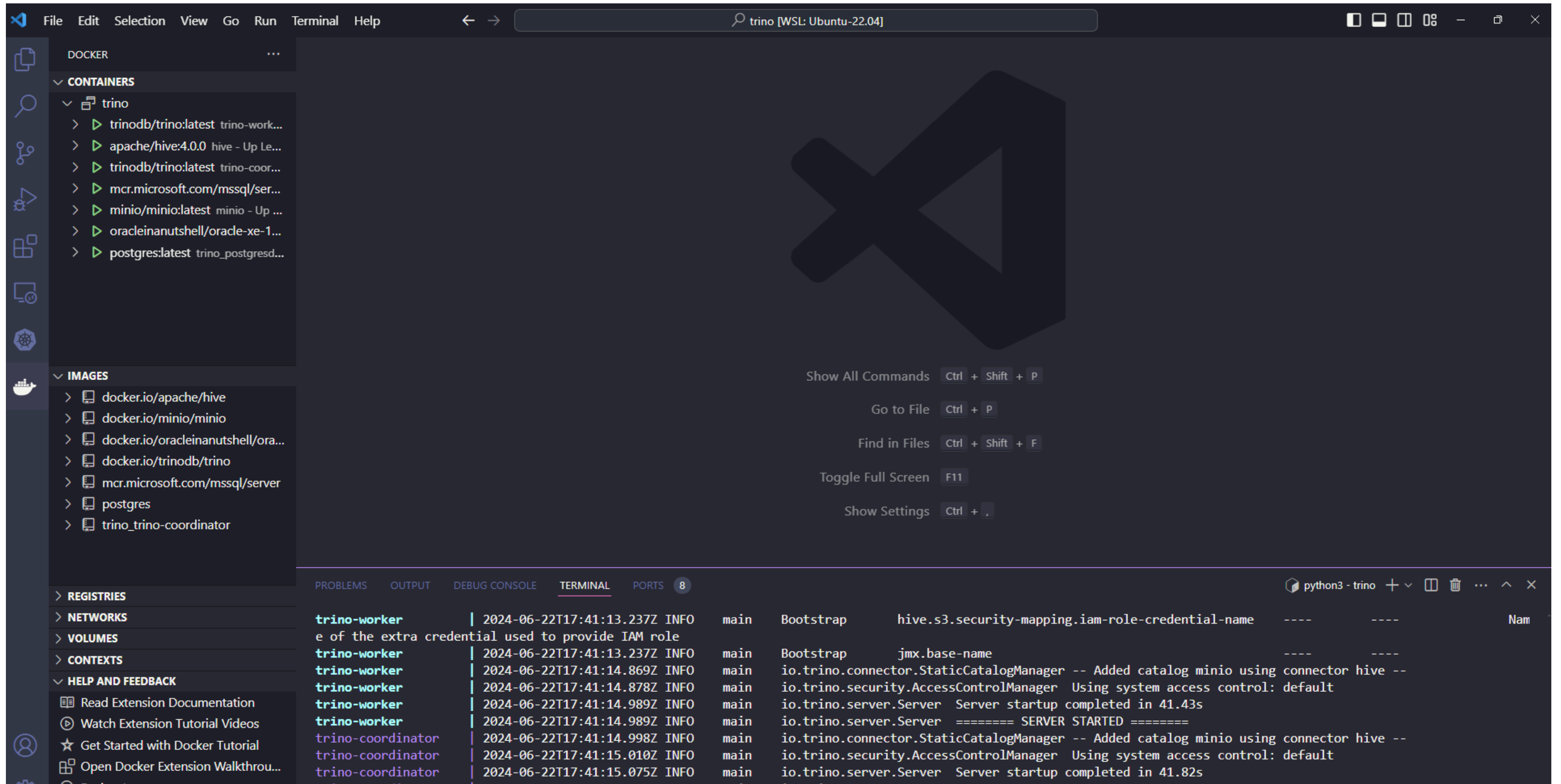


DBeaver



<https://github.com/roussenqtb/trino>

docker-compose up --build



<http://localhost:9001/login>

usuário = minio
senha = minio123

MINIO

OBJECT STORE

AGPL3

LICENSE

minio

.....

Login

Create Bucket

Bucket Name*

broze

View Bucket Naming Rules

Features

Versioning

Object Locking

Quota

OFF

ON

OFF

ON

OFF

ON

Clear

Create Bucket

://

Choose or create a new path

Current Path:

bronze

New Folder Path*

csv

Clear

Create

Refresh

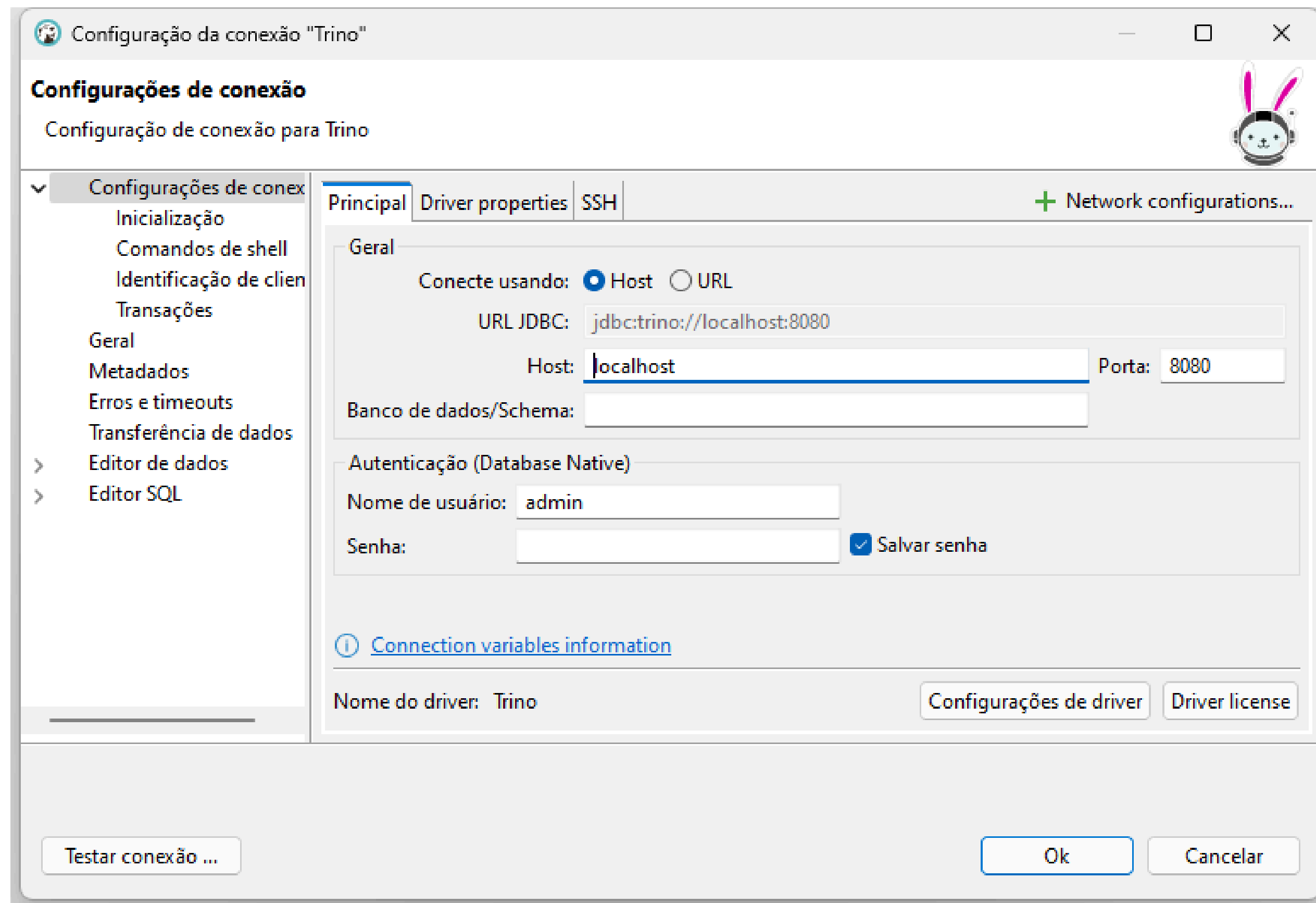
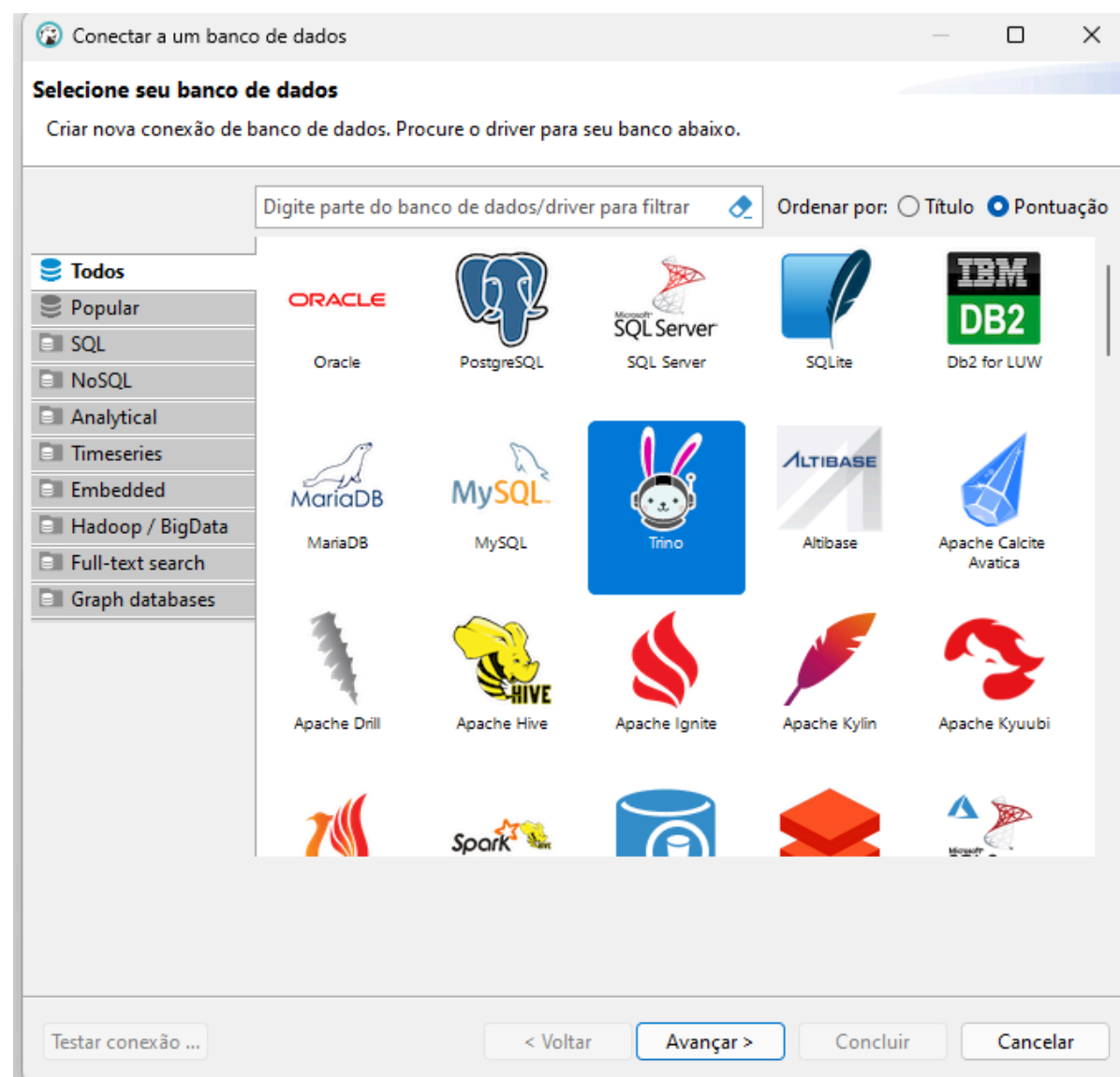
Upload

Upload File

Upload Fold...

Size

Configure o trino no dbeaver



usuário = admin

senha =

Execute o script 01 - minio.sql

The screenshot shows the DBeaver IDE interface. On the left, the 'Navegador de banco de dados' (Database Navigator) pane displays a list of database connections. The 'Trino - localhost:8080' connection is selected and expanded, showing a 'minio' schema with three tables: 'bronze', 'default', and 'information_schema'. On the right, the 'Script-3' editor pane contains the following SQL script:

```
CREATE SCHEMA IF NOT EXISTS minio.bronze
WITH (location = 's3a://bronze/');

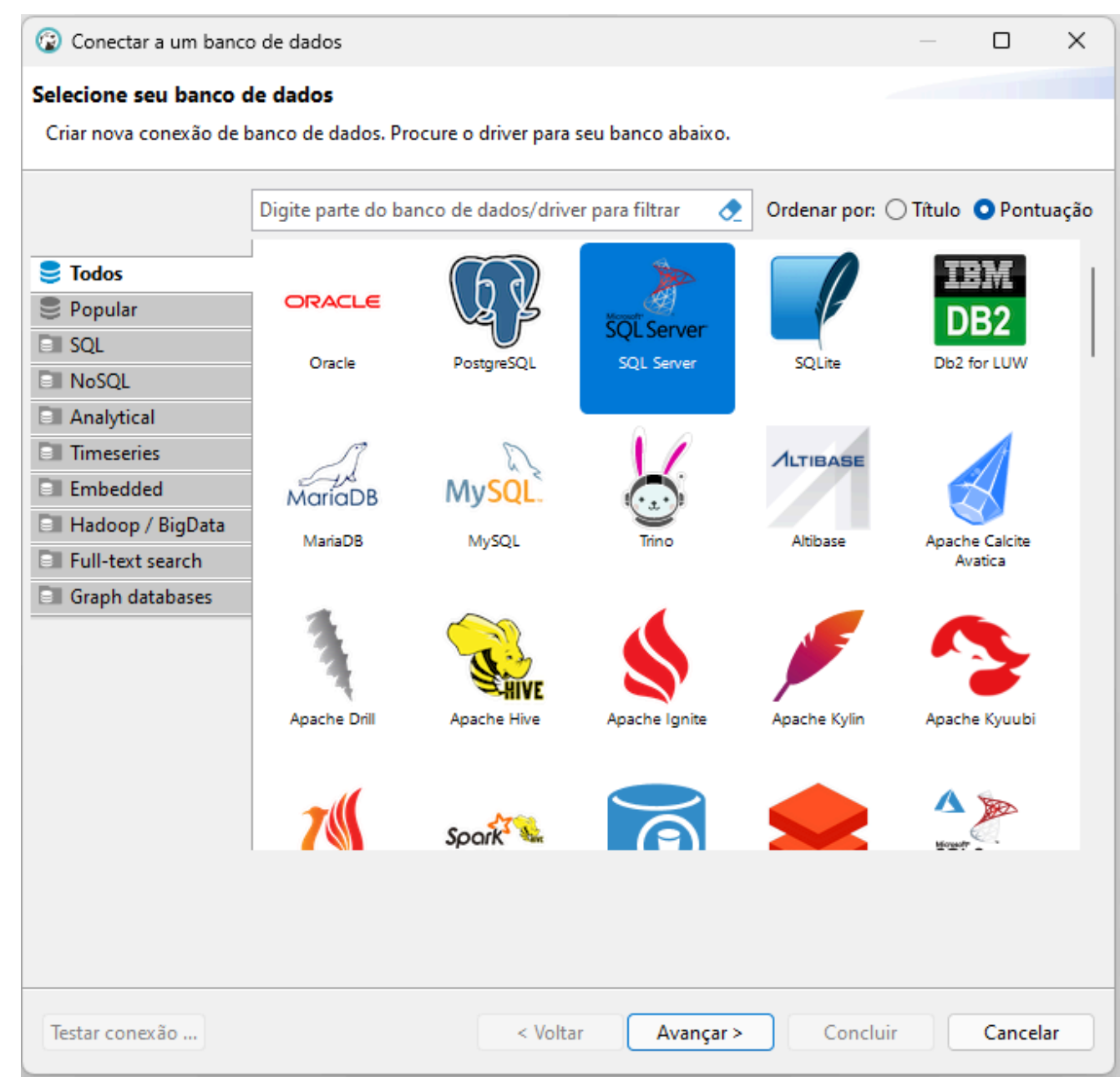
CREATE TABLE IF NOT EXISTS minio.bronze.pessoa_parquet(
  id INT,
  cpf VARCHAR,
  email VARCHAR
)
WITH (
  external_location = 's3a://bronze/parquet/',
  format = 'PARQUET'
);

CREATE TABLE IF NOT EXISTS minio.bronze.pessoa_csv(
  id VARCHAR,
  nome VARCHAR,
  cpf VARCHAR,
  email VARCHAR
)
WITH (
  external_location = 's3a://bronze/csv/',
  format = 'csv',
  csv_separator = ',',
  csv_escape = '"',
  skip_header_line_count = 1
);

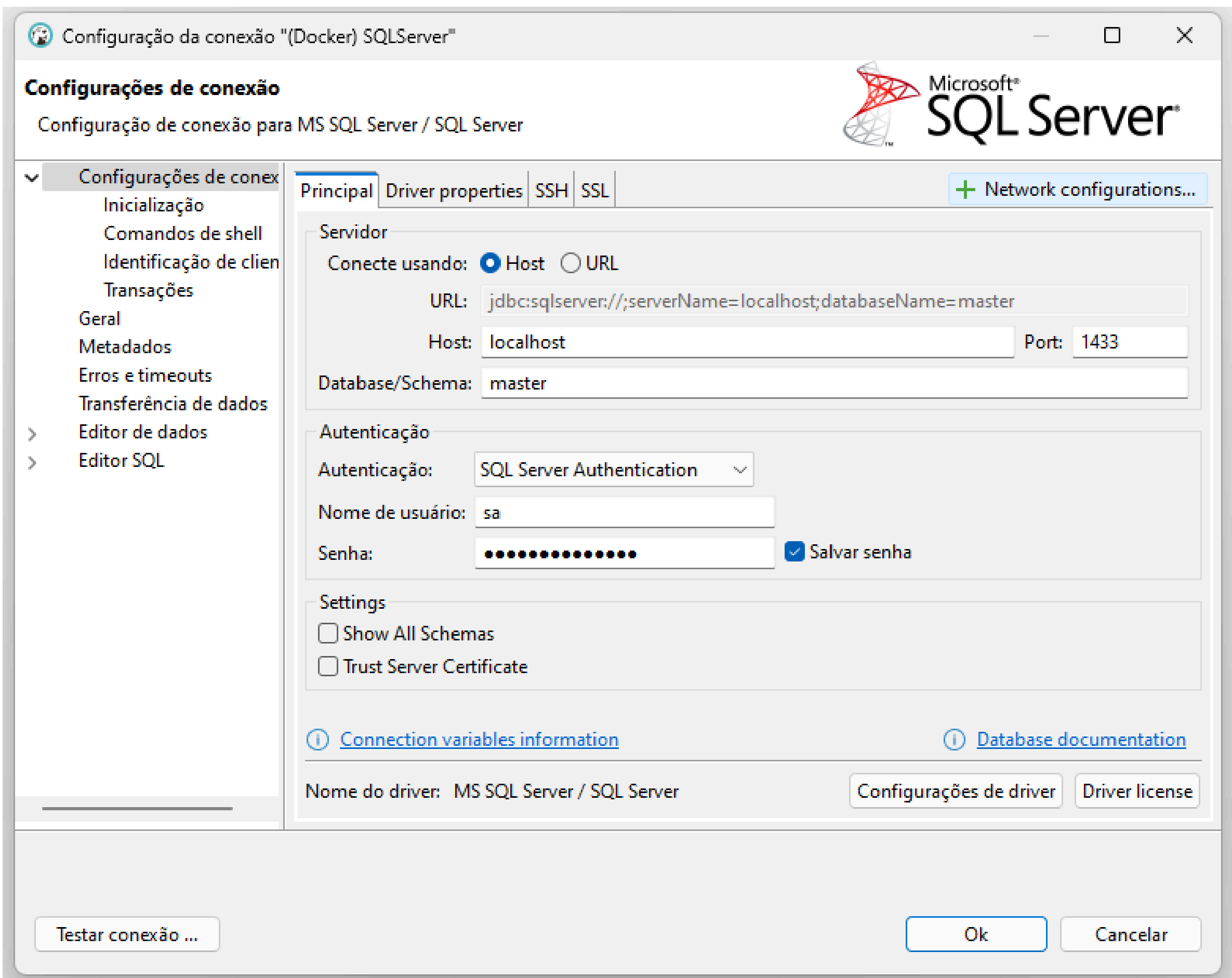
CREATE TABLE IF NOT EXISTS minio.bronze.pessoa_json(
  id int,
  nome VARCHAR,
  cpf VARCHAR,
  email VARCHAR
)
WITH (
  external_location = 's3a://bronze/json/',
  format = 'JSON'
);
```

A tooltip 'Executar script SQL (Alt+X)' is visible over the 'Executar' (Execute) button in the script editor toolbar.

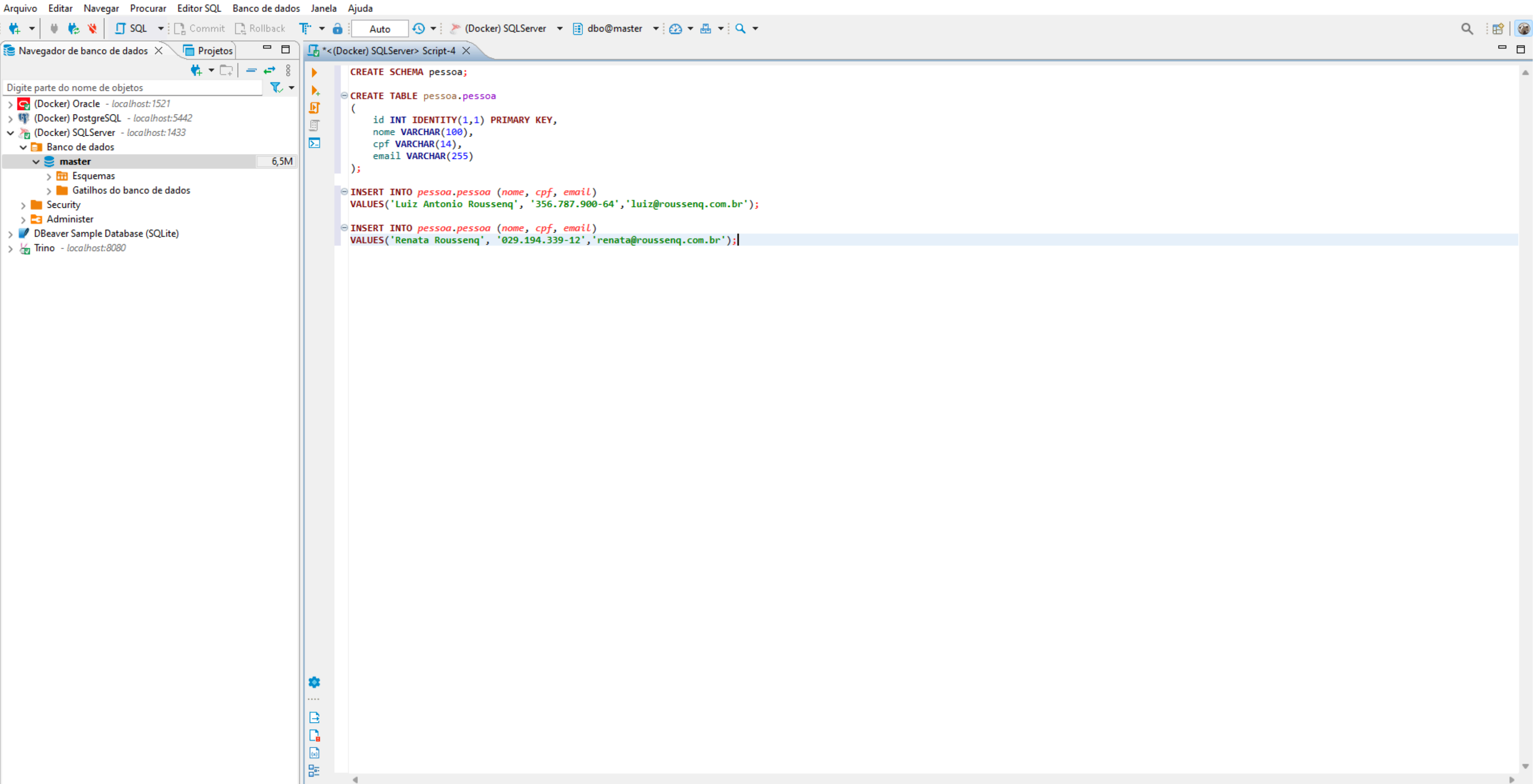
Configure a conexão com banco SQL Server



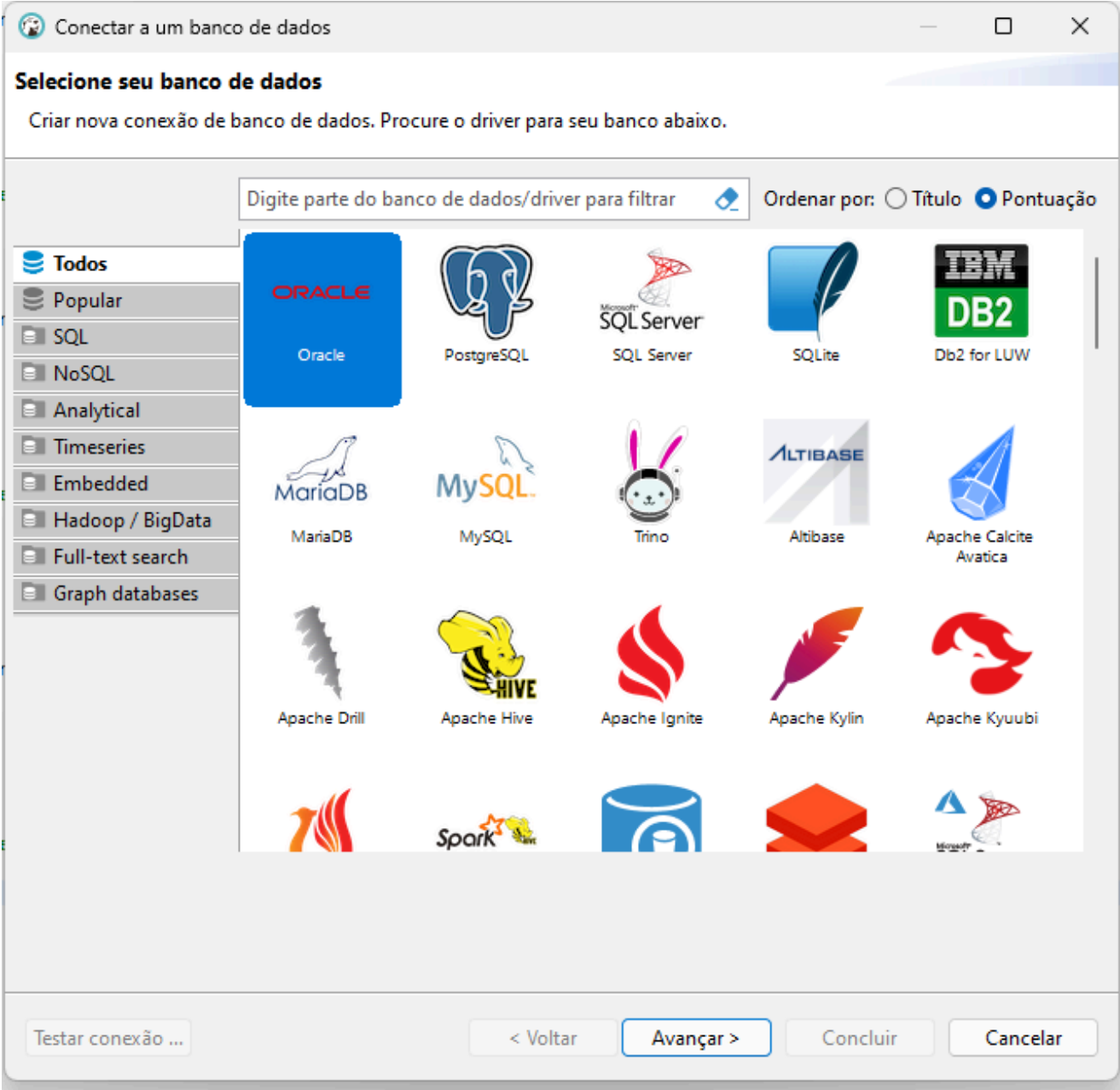
usuário = sa
senha = SqlServer2019!



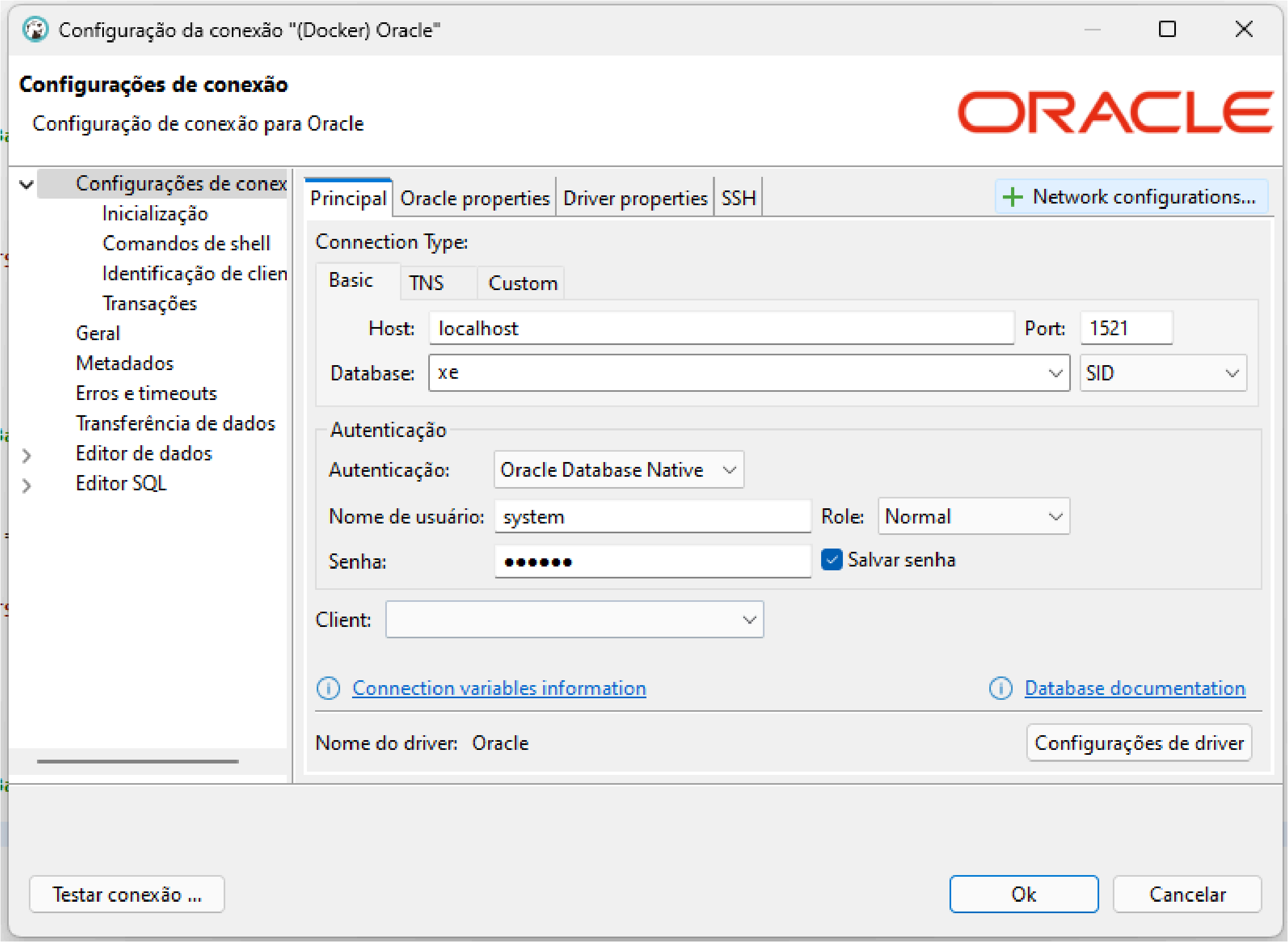
Execute o script 02 - sqlserver.sql



Configure a conexão com banco oracle



usuário = system
senha = oracle



Execute o script 03 - oracle.sql

Arquivo

Editar

Navegar

Procurar

Editor SQL

Banco de dados

Janela

Ajuda

SQL

Commit

Rollback

Auto

(Docker) Oracle

ANONYMOUS

Executar script SQL (Alt+X)

Navegador de banco de dados

Projetos

Digite parte do nome de objetos

(Docker) Oracle - localhost:1521

Esquemas

ANONYMOUS

APEX_040000

APEX_PUBLIC_USER

FLows_FILES

HR

XSSNULL

Metadado Global

Armazenamento

Segurança

Administrador

(Docker) PostgreSQL - localhost:5442

(Docker) SQLServer - localhost:1433

DBeaver Sample Database (SQLite)

Trino - localhost:8080

*<(Docker) Oracle> Script

CREATE TABLESPACE ROUSSENQ
DATAFILE 'ROUSSENQ.dbf' SIZE 100M;

CREATE SEQUENCE pessoa_seq
START WITH 1
NOCACHE
NOCYCLE;

CREATE USER ESTUDO IDENTIFIED BY 123;
ALTER USER ESTUDO DEFAULT TABLESPACE ROUSSENQ;
ALTER USER ESTUDO QUOTA UNLIMITED ON ROUSSENQ;

CREATE TABLE ESTUDO.PESSOA (
ID_PESSOA NUMBER(38,0) NOT NULL,
NOME VARCHAR2(100) NOT NULL,
CPF VARCHAR2(14) NULL,
EMAIL VARCHAR2(100) NULL,
CONSTRAINT PESSOAS_PK PRIMARY KEY (ID_PESSOA)
)
TABLESPACE ROUSSENQ;

CREATE OR REPLACE TRIGGER trg_pessoa_id
BEFORE INSERT ON ESTUDO.PESSOA
FOR EACH ROW
BEGIN
SELECT pessoa_seq.NEXTVAL
INTO :new.id_pessoa
FROM dual;
END;

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Alice', '123.456.789-01', 'alice@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Bob', '987.654.321-09', 'bob@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Charlie', '111.222.333-44', 'charlie@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('David', '555.666.777-88', 'david@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Eva', '999.888.777-66', 'eva@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Fernando', '444.333.222-11', 'fernando@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Gabriela', '777.888.999-00', 'gabriela@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Hugo', '222.333.444-55', 'hugo@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Isabela', '666.555.444-33', 'isabela@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('João', '888.777.666-55', 'joao@example.com');

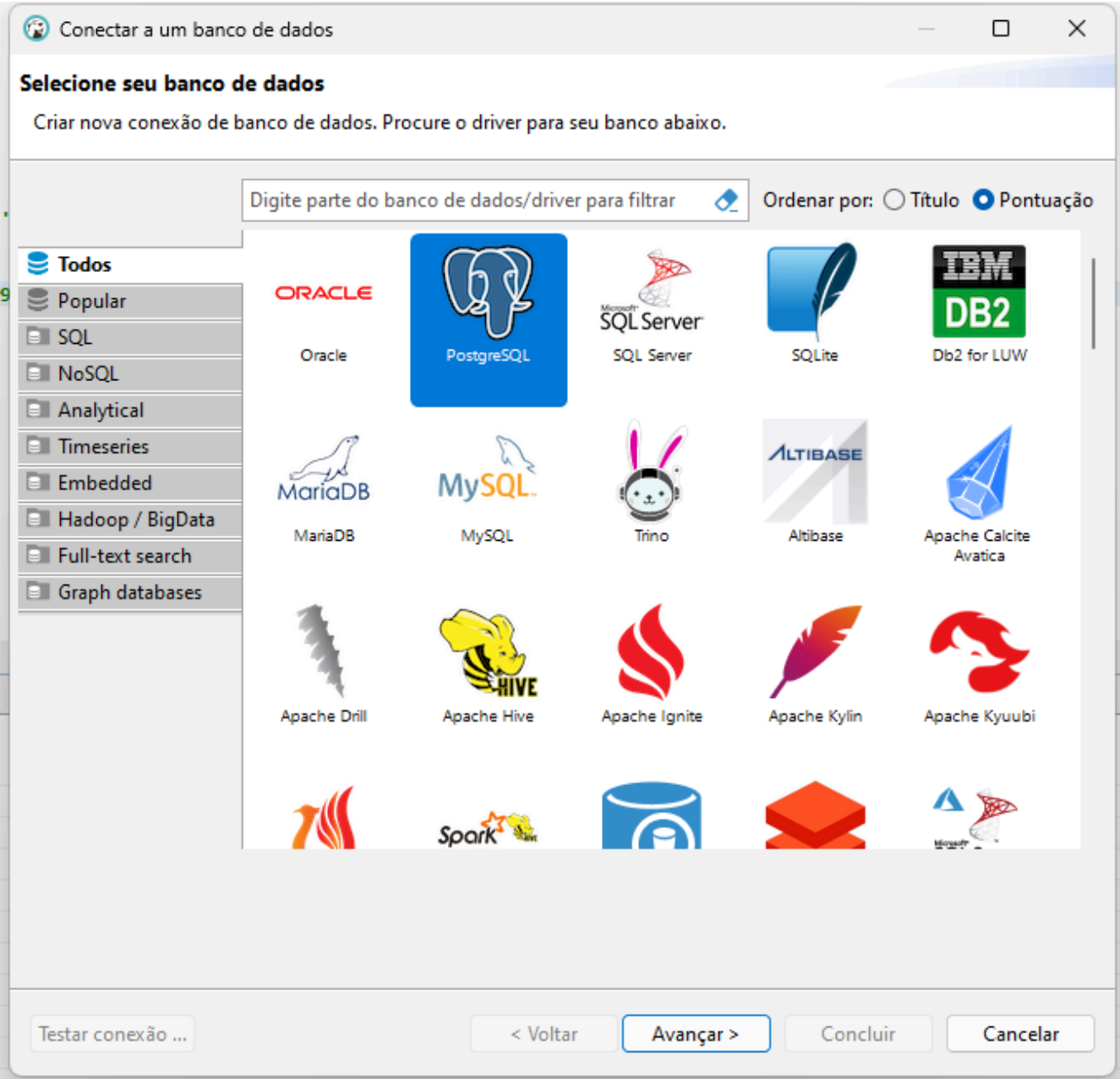
INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Karen', '444.555.666-77', 'karen@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Lucas', '111.222.333-44', 'lucas@example.com');

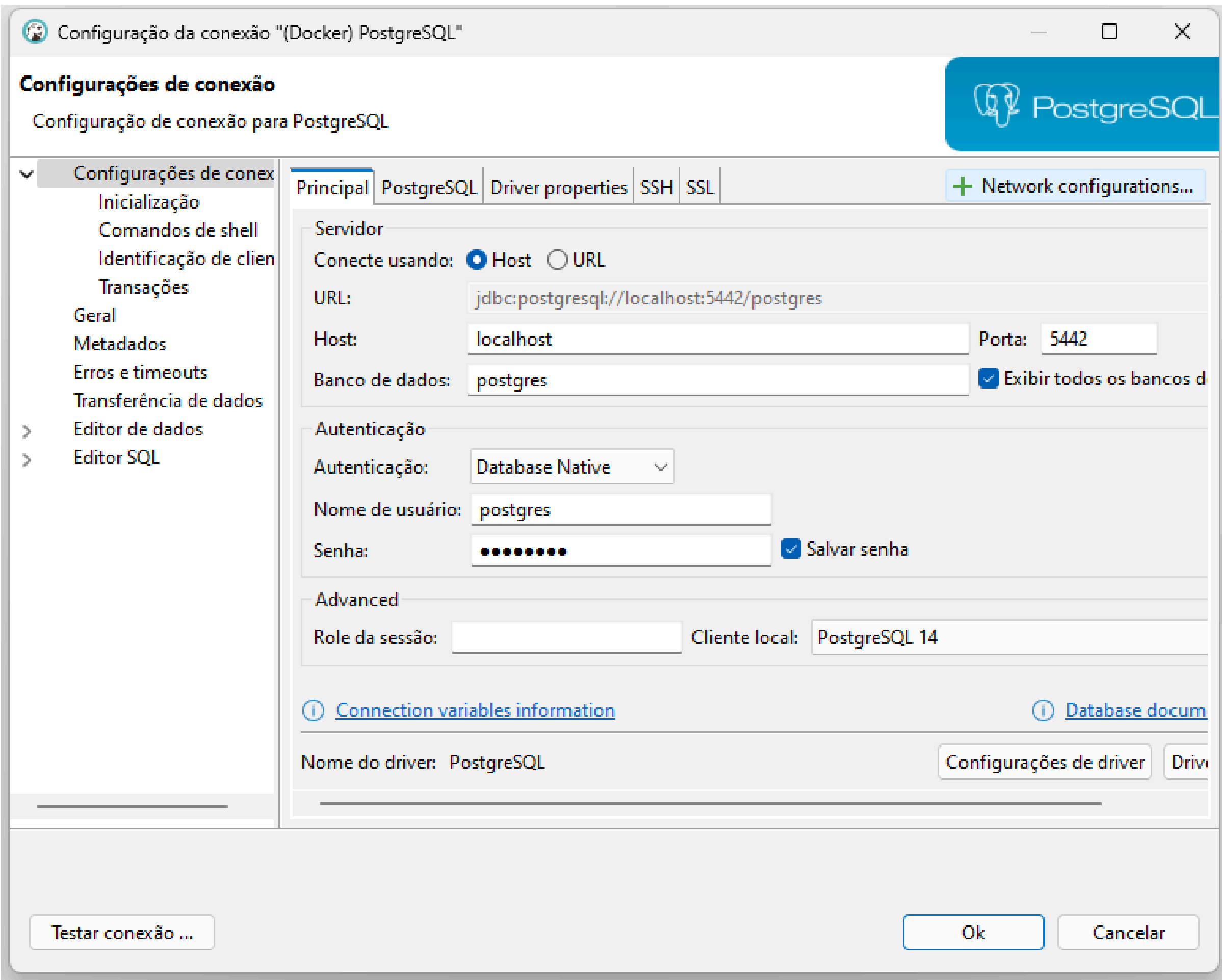
INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Mariana', '555.444.333-22', 'mariana@example.com');

INSERT INTO ESTUDO.PESSOA (nome, cpf, email)
VALUES('Nelson', '999.888.777-66', 'nelson@example.com');

Configure a conexão com banco oracle



usuário = postgres
senha = postgres



Execute o script 04 - postgres.sql

DBEaver 24.1.0 - <(Docker) PostgreSQL> Script-4

Arquivo Editar Navegar Procurar Editor SQL Banco de dados Janela Ajuda

SQL

Commit

Rollback

Auto

(Docker) PostgreSQL

public@estudo

Navegador de banco de dados

Projetos

Digite parte do nome de objetos

> (Docker) Oracle - localhost:1521

> (Docker) PostgreSQL - localhost:5442

> Bancos de dados

> estudo

> postgres

> Administrar

> Informações do sistema

> (Docker) SQLServer - localhost:1433

> DBEaver Sample Database (SQLite)

> Trino - localhost:8080

*<(Docker) PostgreSQL> Script-4

create schema pessoa;

CREATE TABLE pessoa.pessoa (

id bigSERIAL PRIMARY KEY,

nome VARCHAR(100),

cpf VARCHAR(18),

email VARCHAR(150)

);

INSERT INTO pessoa.pessoa (nome, cpf, email)

VALUES('Marcos', '125.486.789-01', 'marcos@example.com');

INSERT INTO pessoa.pessoa (nome, cpf, email)

VALUES('Alexandre', '927.634.391-09', 'alexandre@example.com');

INSERT INTO pessoa.pessoa (nome, cpf, email)

VALUES('Marcelo', '611.822.933-11', 'marcelo@example.com');

INSERT INTO pessoa.pessoa (nome, cpf, email)

VALUES('Luiz Antonio Roussenq', '356.787.900-64', 'luiz@roussenq.com.br');

RRT

nt RR

Gravável

Inserção Inteligente

17 · 74 · 602

Sel: 0 | 0

Execute o script 05- consulta.sql

DBeaver 24.1.0 - <Trino> Script-3

Arquivo Editar Navegar Procurar Editor SQL Banco de dados Janela Ajuda

Navegador de banco de dados

Projetos

Digite parte do nome de objetos

> (Docker) Oracle - localhost:1521

> (Docker) PostgreSQL - localhost:5442

> (Docker) SQLServer - localhost:1433

> DBeaver Sample Database (SQLite)

> Trino - localhost:8080

> minio

> bronze

> Tabelas

> Visualização

> Índices

> Procedimentos

> Tipos de dados

> default

> information_schema

> oracle

> postgresql

> sqlserver

> system

*<Trino> Script-3

```
select id_pessoa,nome,cpf,email,'ORACLE'as origem from oracle.estudo.pessoa
union
select id,nome,cpf,email,'POSTGRES'as origem from postgresql.pessoa.pessoa
union
select id,nome,cpf,email,'SQLSERVER'as origem from sqlserver.pessoa.pessoa
union
select id,nome,cpf,email,'PARQUET'as origem from minio.bronze.pessoa_parquet
union
select cast(id as integer)as id,nome,cpf,email,'CSV'as origem from minio.bronze.pessoa_csv
union
select id,nome,cpf,email,'JSON'as origem from minio.bronze.pessoa_json
order by cpf
```

Resultados 1

select id_pessoa,nome,cpf,email,'ORACLE'as origem from oracle.est

Insira uma expressão SQL para filtrar os resultados (use Ctrl+Espaço)

	123 id_pessoa	ABC nome	ABC cpf	ABC email	ABC origem
7	8	Hugo	222.333.444-55	hugo@example.com	ORACLE
8	5	Leticia	313.416.280-60	leticia@example.com	CSV
9	3	Carlos Souza	333.444.555-66	carlos.souza@example.com	JSON
10	15	Olivia	333.444.555-66	olivia@example.com	ORACLE
11	1	Luiz Antonio Roussenq	356.787.900-64	luiz@roussenq.com.br	SQLSERVER
12	16	Luiz Antonio Roussenq	356.787.900-64	luiz@roussenq.com.br	ORACLE
13	4	Luiz Antonio Roussenq	356.787.900-64	luiz@roussenq.com.br	POSTGRES
14	3	Luiz A Roussenq	356.787.900-64	luiz@roussenq.com.br	CSV
15	1	Luiz Antonio Roussenq	356.787.900-64	luiz@roussenq.com.br	JSON
16	2	Luiz	356.787.900-64	Luiz@example.com	PARQUET
17	4	Christina Roussenq	383.651.420-66	chris@roussenq.com.br	JSON
18	6	Fernando	444.333.222-11	fernando@example.com	ORACLE
19	11	Karen	444.555.666-77	karen@example.com	ORACLE
20	1	Gabriel	464.782.490-00	Gabriel@example.com	PARQUET
21	13	Mariana	555.444.333-22	mariana@example.com	ORACLE
22	4	David	555.666.777-88	david@example.com	ORACLE
23	3	Marcelo	611.822.933-11	marcelo@example.com	POSTGRES
24	9	Isabela	666.555.444-33	isabela@example.com	ORACLE
25	3	Leonardo	710.361.460-14	leonardo@example.com	PARQUET
26	6	Manoel Souza	732.318.450-92	manoel.souza@example.com	JSON
27	5	Patricia Oliveira	732.318.450-92	patricia.oliveira@example.co	JSON
28	6	Lidiane	741.127.400-34	lidiane@example.com	PARQUET
29	4	Ana	771.734.820-97	ana@example.com	PARQUET
30	7	Gabriela	777.888.999-00	gabriela@example.com	ORACLE
31	5	Keli	854.801.610-83	keli@example.com	PARQUET
32	2	Maria Oliveira	888.777.666-55	maria.oliveira@example.com	JSON
33	10	João	888.777.666-55	joao@example.com	ORACLE
34	2	Alexandre	927.634.391-09	alexandre@example.com	POSTGRES
35	4	Ana	947.919.170-90	ana@example.com	CSV

Atualizar

Salvar

Cancelar

Exportar dados

200

39

39 linha(s) recuperada(s) - 0,741s (0,375s recuperado), em 2024-06-22 às 15:11:29

Execute a apptrino

```
python
gerarparquet.py
apptrino.py X
apptrino.py > ...
8     user='admin',
9     catalog='',
10    schema=''
11 )
12
13 sql_query = '''
14     SELECT ID,NOME,CPF,EMAIL,'PARQUET'AS ORIGEM FROM MINIO.BRONZE.PESSOA_PARQUET      "ORIGEM": Unknown word.
15     UNION
16     SELECT CAST(ID AS INTEGER)AS ID,NOME,CPF,EMAIL,'CSV'AS ORIGEM FROM MINIO.BRONZE.PESSOA_CSV      "ORIGEM": Unknown word.
17     UNION
18     SELECT ID,NOME,CPF,EMAIL,'JSON'AS ORIGEM FROM MINIO.BRONZE.PESSOA_JSON      "ORIGEM": Unknown word.
19     ORDER BY CPF
20     '''
21
22 cur = conn.cursor()
23 cur.execute(sql_query)
24
25 rows = cur.fetchall()
26 for row in rows:
27     print(row)
28
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Code

```
[1, 'Luiz Antonio Roussenq', '356.787.900-64', 'luiz@roussenq.com.br', 'JSON']
[4, 'Christina Roussenq', '383.651.420-66', 'chris@roussenq.com.br', 'JSON']
[1, 'Gabriel', '464.782.490-00', 'Gabriel@example.com', 'PARQUET']
[3, 'Leonardo', '710.361.460-14', 'leonardo@example.com', 'PARQUET']
[5, 'Patricia Oliveira', '732.318.450-92', 'patricia.oliveira@example.com', 'JSON']
[6, 'Manoel Souza', '732.318.450-92', 'manoel.souza@example.com', 'JSON']
[6, 'Lidiane', '741.127.400-34', 'lidiane@example.com', 'PARQUET']
[4, 'Ana', '771.734.820-97', 'ana@example.com', 'PARQUET']
[5, 'Keli', '854.801.610-83', 'keli@example.com', 'PARQUET']
[2, 'Maria Oliveira', '888.777.666-55', 'maria.oliveira@example.com', 'JSON']
[4, 'Ana', '947.919.170-90', 'ana@example.com', 'CSV']
[2, 'Maria', '987.654.321-00', 'maria@example.com', 'CSV']

[Done] exited with code=0 in 1.341 seconds
```

docker-compose down

