dbt

Treinamento básico de dbt

Preparação de Treinamento

Pré-requisitos

- Docker
- Python 3.8+
- DBeaver ou outro SQL Client de preferencia
- Oracle Database

Oracle

Script para criação de schemas (estudantes) "dbt_XX"

```
ALTER SESSION SET "_ORACLE_SCRIPT"=true;

CREATE USER dbt_XX
IDENTIFIED BY senha_XX
DEFAULT TABLESPACE users
QUOTA UNLIMITED on users;

GRANT create session TO dbt_XX;
GRANT resource TO dbt_XX;
GRANT create view TO dbt_XX;
GRANT create materialized view TO dbt_XX;
```

```
CREATE TABLE dbt_XX.tipo_cliente(
tipo char(1) NOT NULL PRIMARY KEY ,
descricao varchar2(50) NOT NULL
);

INSERT INTO dbt_XX.tipo_cliente(tipo, descricao) VALUES ('V', 'Varejo');
INSERT INTO dbt_XX.tipo_cliente(tipo, descricao) VALUES ('A', 'Atacado');
INSERT INTO dbt_XX.tipo_cliente(tipo, descricao) VALUES ('C',
'Consumidor');

COMMIT;
```

Acesso ao Oracle

- *Atividade*: Conectar ao Oracle pelo DBeaver

Download do Projeto de Estudo

- *Atividade*: Descompactar o arquivo em uma pasta `dbt`

Python (ambiente virtual)

- Arquivos importantes

> - `setup.py` - Configuração de Python por setuptools
> - `requirements.txt` - Dependencias do dbt
> - `requirements.dev.txt` - Dependencias de desenvolvimento

- Power Shell (Ambiente Microsoft)

```
python -m venv venv
.\venv\Scripts/activate.bat
pip install --upgrade pip
pip install -e .[interactive]
```

- Bash (Linux, Mac, WSL)

```
python -m venv venv
. venv/bin/activate
pip install --upgrade pip
pip install -e .[interactive]
```

- Make

```
make install
```

> *Nota*: Ativar "venv"

dbt

Criação de projeto

- Linux: criar pasta `.dbt` no home do usuário

```
mkdir ~/.dbt
```

- Windows: criar pasta `.dbt` no home do usuário

```
# Obter o caminho da pasta do usuário atual usando variável de ambiente
$userProfile = $env:USERPROFILE

# Definir o nome da pasta a ser criada
```

```
$folderName = ".dbt"

# Combinar o caminho do usuário com o nome da pasta para obter o caminho
completo
$folderPath = Join-Path -Path $userProfile -ChildPath $folderName

# Criar a pasta se ela não existir
if (-Not (Test-Path -Path $folderPath)) {
    New-Item -ItemType Directory -Path $folderPath
    Write-Output "Pasta '$folderName' criada em '$userProfile' com
sucesso."
} else {
    Write-Output "A pasta '$folderName' já existe em '$userProfile'."
}
```

- Criar projeto no dbt

```
dbt init <project_name> --profiles-dir=$(shell pwd)/profiles
```

> - project: data_transformation
> - database: [1] oracle
> - protocol: tcp
> - PORT: 1521
> - threads: 1

Acesso ao Oracle

Configurar variáveis de ambiente `.env`

```
DBT_ORACLE_HOST=
DBT_ORACLE_USER=dbt_XX
DBT_ORACLE_PASSWORD=senha_XX
DBT_ORACLE_SERVICE=
DBT_ORACLE_SCHEMA=dbt_XX
```

Ler o `.env` e carregar as variáveis de ambiente

- Power Shell (Ambiente Microsoft)

```
Get-Content .env | ForEach-Object {
    if ($_ -match "^\s*([^=]+)\s*=\s*(.*)\s*$") {
        [System.Environment]::SetEnvironmentVariable($matches[1],
$matches[2])
    }
}
```

```
dir env:
```

- Bash (Linux, Mac, WSL)

```
export $(cat .env | xargs)
```

```
env | grep DBT
```

Verificar a conexão ao Oracle no dbt

```
dbt debug --project-dir data_transformation
```

Arquivos de configuração do projeto

> - `profiles.yml` - Configurações de acesso ao BD
> - `dbt_project.yml` - Configurações do projeto

- *Atividade*: Identificar o path dos arquivos
- *Atividade*: Identificar as variáveis de ambiente (Template Jinja)
- *Atividade*: Mover `profiles.yml` para pasta do projeto `data_transformation`

```
dbt debug --project-dir data_transformation --profiles-dir
data_transformation
```

dbt commands (CLI)

- https://docs.getdbt.com/reference/dbt-commands

> *Importante*: Acessar a pasta do projeto `cd data_transformation`

- Aplicar as alterações no Oracle

```
dbt run
```

> - *Erro*: Verificar `data_transformation/logs/dbt.log`
> - *Importante*: Verificar `data_transformation/target/run/*`
> - *Nota*: Verificar as tabelas e views criadas no schema

- Debugar

```
dbt run --debug
```

- Testar validações built in
- https://docs.getdbt.com/docs/build/data-tests

```
dbt test
```

> *Importante*: Verificar `data_transformation/logs/dbt.log*`

- Compilar o projetos

```
dbt compile
```

> *Importante*: Verificar `data_transformation/target/compiled/*`

- Executar o CLI fora da pasta (usado pelo Airflow)

```
dbt run --project-dir data_transformation --profiles-dir
data_transformation
```

- Documentação e Data Lineage
- https://docs.getdbt.com/reference/commands/cmd-docs

```
dbt docs generate --profiles-dir data_transformation --project-dir
data_transformation
dbt docs serve --profiles-dir data_transformation --project-dir
data_transformation
```

> *Navegar*: http://localhost:8080

Seeds

Arquivos estáticos

> Pasta: `data_transformation/seeds`

```
dbt seed
```

> *Nota*: Verificar as tabelas

- *Atividade*: Criar um arquivo `data_transformation/seeds/raw_clientes.csv` e gerar a tabela no Oracle

```
id,nome,uf,tipo_cliente,limite_credito,data_cadastro
1,Marta,SP,V,-1,
2,Pelé,RJ,A,5300,03/05/24
3,Maradona,MG,C,,03/05/2024
4,Messi,MG,,18300
5,Raí,SR,V,,
```

```
dbt seed --select raw_clientes
```

> *Importante*: Verificar `data_transformation/target/run/*`

- *Atividade*: Criar um arquivo `data_transformation/seeds/produtos.csv` e gerar a tabela no Oracle
- *Atividade*: Adicionar uma coluna no arquivo `data_transformation/seeds/produtos.csv` e gerar a tabela no Oracle

Models/CTE

Tabelas ou views com/sem schemas a partir de CTE/Common Table Expression

```
WITH nome_expressão [( nome_colunas [,...n] )]
AS
(CTE_definição_da_consulta)
```

```
WITH clientes AS (
    SELECT * FROM RAW_CLIENTES
), ufs AS (
    SELECT * FROM UNIDADES_FEDERACAO
)
SELECT c.id, c.nome, c.uf, ufs.nome AS estado
FROM clientes c JOIN ufs ufs ON c.UF = ufs.sigla
```

- *Atividade*: Criar `data_transformation/models/dim/dim_clientes.sql` para o CTE acima

```
dbt run --select dim_clientes
```

> - *Observação*: Qual objeto foi criado no BD?
> - *Observação*: Como está a documentação?

Models/References

Referenciar objetos com uso de `{{ ref("object_name") }}`

- https://docs.getdbt.com/reference/dbt-jinja-functions/ref
- *Atividade*: Alterar `data_transformation/models/dim/dim_clientes.sql`

```
WITH clientes AS (
    SELECT * FROM {{ ref("raw_clientes") }}
), ufs AS (
    SELECT * FROM {{ ref("unidades_federacao") }}
)
SELECT c.id, c.nome, c.uf, ufs.nome AS estado
FROM clientes c LEFT JOIN ufs ufs ON c.UF = ufs.sigla
```

```
dbt run --select dim_clientes
```

> - *Observação*: Como está a documentação? (Data Lineage)

Models/Sources

Mapear objetos existentes que não foram criados pelo dbt

- https://docs.getdbt.com/docs/build/sources

`data_transformation/models/sources.yml`

```
version: 2

sources:
  - name: raw
    schema: dbt_XX
    tables:
      - name: raw_tipo_cliente
        identifier: tipo_cliente
```

- *Atividade*: Alterar `data_transformation/models/dim/dim_clientes.sql`

```
WITH clientes AS (
    SELECT *
    FROM {{ ref("raw_clientes") }}
), ufs AS (
    SELECT
        uf.sigla,
        uf.nome as estado
```

```
    FROM {{ ref("unidades_federacao") }} uf
), tipos_cliente AS (
    SELECT *
    FROM {{ source ("raw", "raw_tipo_cliente") }}
)
SELECT
    c.id,
    c.nome,
    c.uf,
    ufs.estado,
    tc.descricao as tipo_cliente
FROM
    clientes c LEFT JOIN ufs ufs ON c.UF = ufs.sigla
    LEFT JOIN tipos_cliente tc on tc.tipo = c.tipo_cliente
```

```
dbt compile --select dim_clientes
```

> - *Importante*: Verificar `data_transformation/target/run/*`

```
dbt run --select dim_clientes
```

Models/Materializations

- https://docs.getdbt.com/docs/build/materializations
- *Atividade*: Alterar `data_transformation/dbt_project.yml`

```
models:
  data_transformation:
    # Config indicated by + and applies to all files under models/example/
    example:
      +materialized: view
    dim:
      +materialized: table
```

```
dbt run --select dim_clientes
```

> - *Importante*: Verificar `data_transformation/target/run/*`

Models/Materializations/Table

- table -> `create table as`

```
{{ config(materialized='table') }}
```

Models/Materializations/View

- view -> `create view as`

```
{{ config(materialized='view') }}
```

Models/Materializations/Incremental

- incremental -> `merge`

```
{{ config(materialized='incremental') }}
```

https://docs.getdbt.com/docs/build/incremental-models

https://docs.getdbt.com/docs/build/incremental-strategy

- _Não tem o "merge" para Oracle

Models/Materializations/Ephemeral

- ephemeral -> `cte`

```
{{ config(materialized='ephemeral') }}
```

Models/Materializations/Ephemeral

- materialized view

```
{{ config(materialized='materialized_view') }}
```

Schemas e validações no dbt

- *Atividade*: Criar o schema `data_transformation/models/dim/dim_clientes.yml`

```
version: 2

models:
  - name: dim_clientes
```

```
    description: Repositório de dados de clientes
    config:
      contract:
        enforced: true
    columns:
      - name: id
        data_type: number
        description: "The primary key for this table"
      - name: nome
        data_type: varchar2(4000)
        description: "Nome do cliente"
      - name: uf
        data_type: varchar2(4000)
        description: "Unidade de Federação (sigla)"
      - name: estado
        data_type: varchar2(4000)
        description: "Unidade de Federação"
      - name: tipo_cliente
        data_type: varchar2(50)
        description: "Tipo de cliente"
      - name: coluna_nova_inexistente
        data_type: varchar2(100)
        description: "coluna_nova_inexistente"
```

```
dbt run --select dim_clientes
```

> *Importante*: Verificar o erro causado pela validação do schema

Macros

- *Atividade*: Criar a macro `data_transformation/macros/dash_min_max.sql`

```
{% macro dash_min_max(model, column, identifier, min, max) %}
    SELECT
    '{{ model }}' AS tabela,
    'min_max' AS criterio,
    '{{ column }}' AS campo,
    'WHERE {{ identifier }} = ''' || {{ identifier }} || '''' AS detalhe
    FROM {{ model }}
    WHERE {{ column }} NOT BETWEEN {{ min }} AND {{ max }}
{% endmacro %}
```

- *Atividade*: Criar a macro `data_transformation/macros/dash_not_null.sql`

```
{% macro dash_not_null(model, column, identifier) %}
    SELECT
    '{{ model }}' AS tabela,
```

```
    'not null' AS criterio,
    '{{ column }}' AS campo,
    'WHERE {{ identifier }} = ''' || {{ identifier }} || '''' AS detalhe
    FROM {{ model }}
    WHERE {{ column }} IS NULL
{% endmacro %}
```

- *Atividade*: Criar a model `data_transformation/models/dq/dash_errors.sql`

```
WITH errors AS (
    {{ dash_not_null('RAW_CLIENTES', 'TIPO_CLIENTE', 'ID' ) }}
    UNION ALL
    {{ dash_min_max('RAW_CLIENTES', 'LIMITE_CREDITO', 'ID', 0, 9999999 ) }}
)
SELECT * FROM ERRORS
```

```
dbt run --select dash_errors
```

> - *Importante*: Verificar a view ou table criada no Oracle.