Data Sinergy: Excel to Power BI through ETL and SQL Cycle

This document aims to present the architecture of the service **"Data Synergy: Excel for Power BI through the ETL and SQL Cycle"**, in a clear and objective way, in order to simplify the understanding of all its components and operation. This service is responsible for providing Business Intelligence solutions, attending to the details of learning more about the data lifecycle.
Throughout this document, technical details about the service, its structure and how each part relates to the others will be presented. The documentation will be useful both for developers working on the service and those who want to better understand how the technology can be applied in their industry.
Furthermore, the documentation is designed to meet the needs of different audiences, from those with a more technical knowledge in technology to those with limited knowledge in this area.

Topics

1. First Contact

1.1. Scenario and Challenges

A medium-sized company, it is growing and is looking for ways to streamline its human resources management and data analysis processes. Today, they rely on Excel spreadsheets to analyze and share employee information, resulting in difficulties with data integrity and efficient collaboration.

They decided to collaborate with business intelligence consulting firm Data Ape to improve their approach to data management and enable more effective analytics. The main objective is to carry out a "Proof of Concept" (POC) to demonstrate how process transformation using BI can bring efficiency, deeper insights and the ability to make informed decisions.

1.2. Tools

The **tools used are:**

- Excel;
- Visual Studio 2019 - (SSIS / Database Project / SSRS);
- SQL Server;
- Power BI;
- Power BI Report Server.

2. Excel

2.1. Sheets

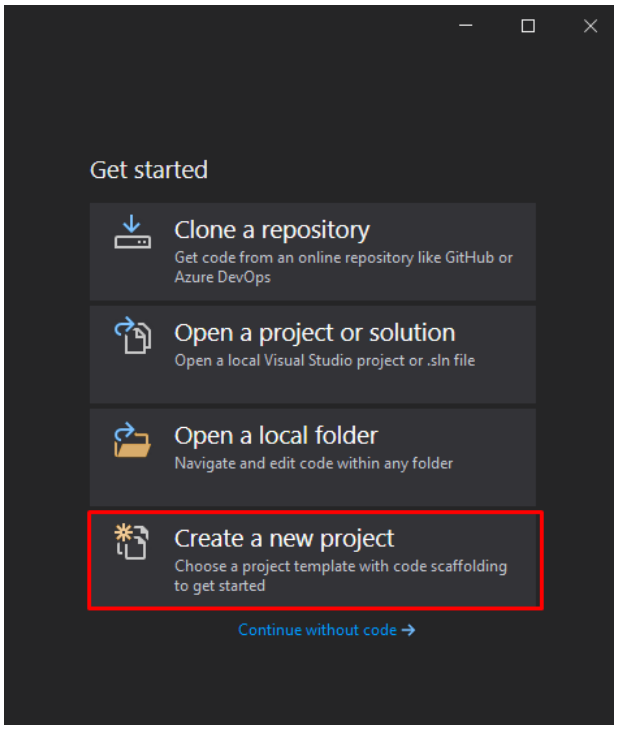
Two worksheets were sent:

- **Cargos.xlsx -** Spreadsheet containing information on all positions in the company and their respective codes
- **Employees.xlsx -** Spreadsheet that contains all the employees that passed through the company and the complete information of each employee

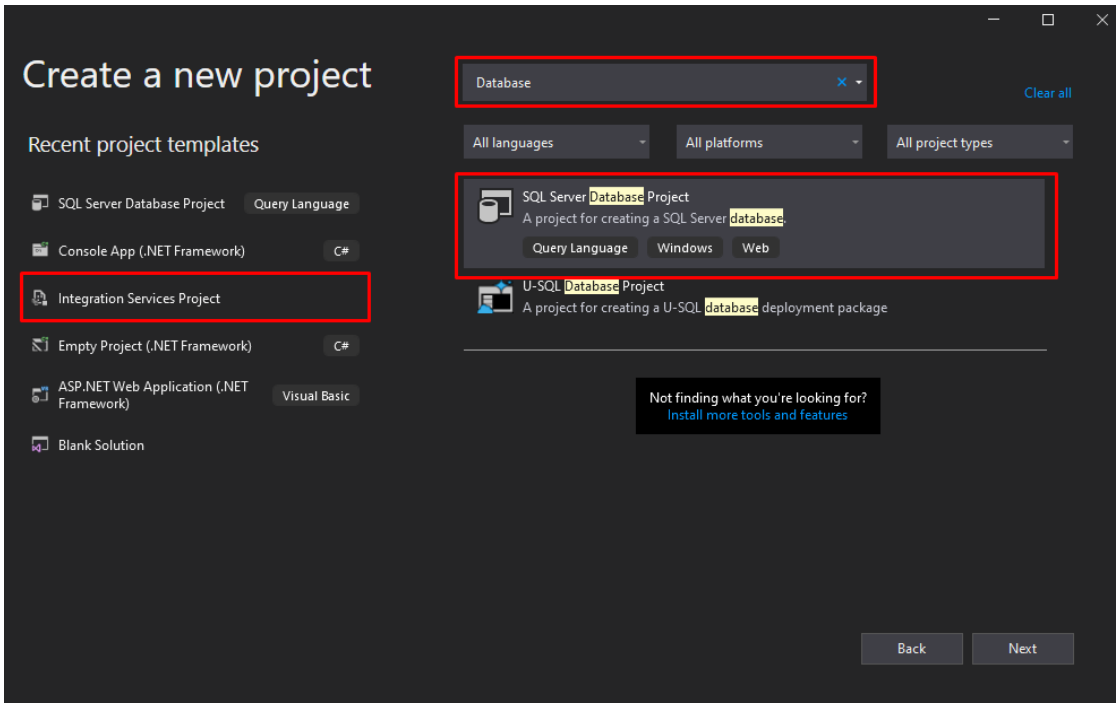
3. ETL e SQL

3.1. Creating Services in Visual Studio Visual Studio - (Database, SSIS e SSRS)

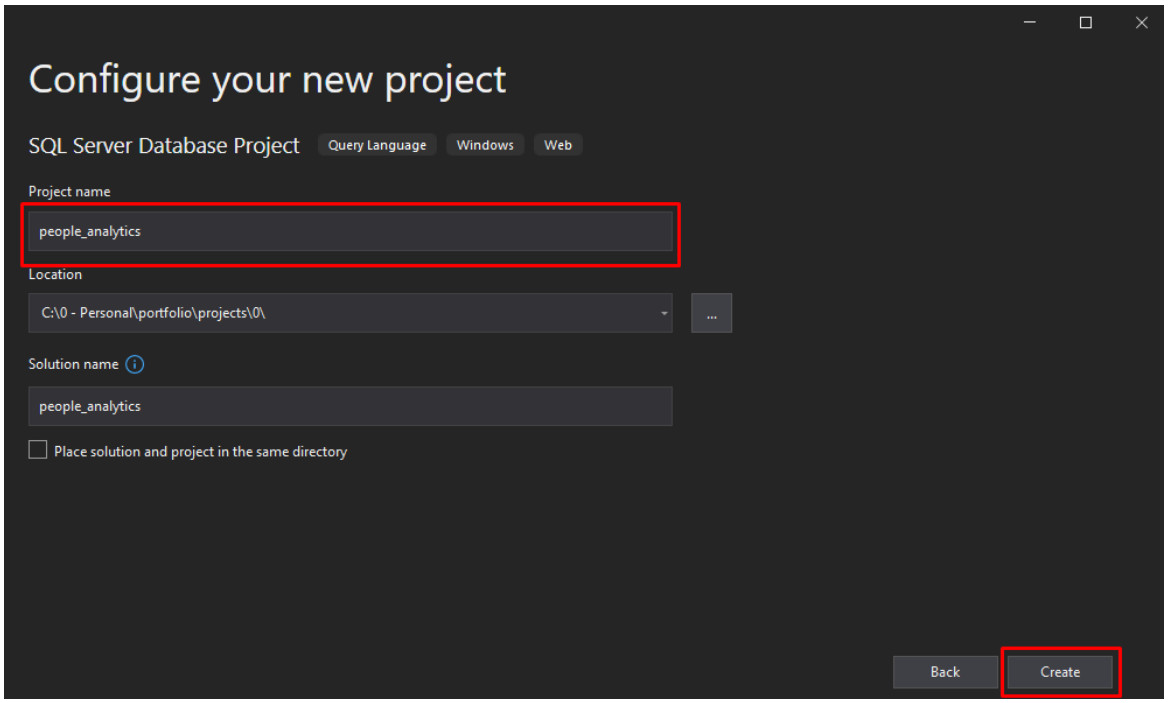
To start, let's open Visual Studio (I'm using the 2019 version) and click on “Create New Project”.

- 

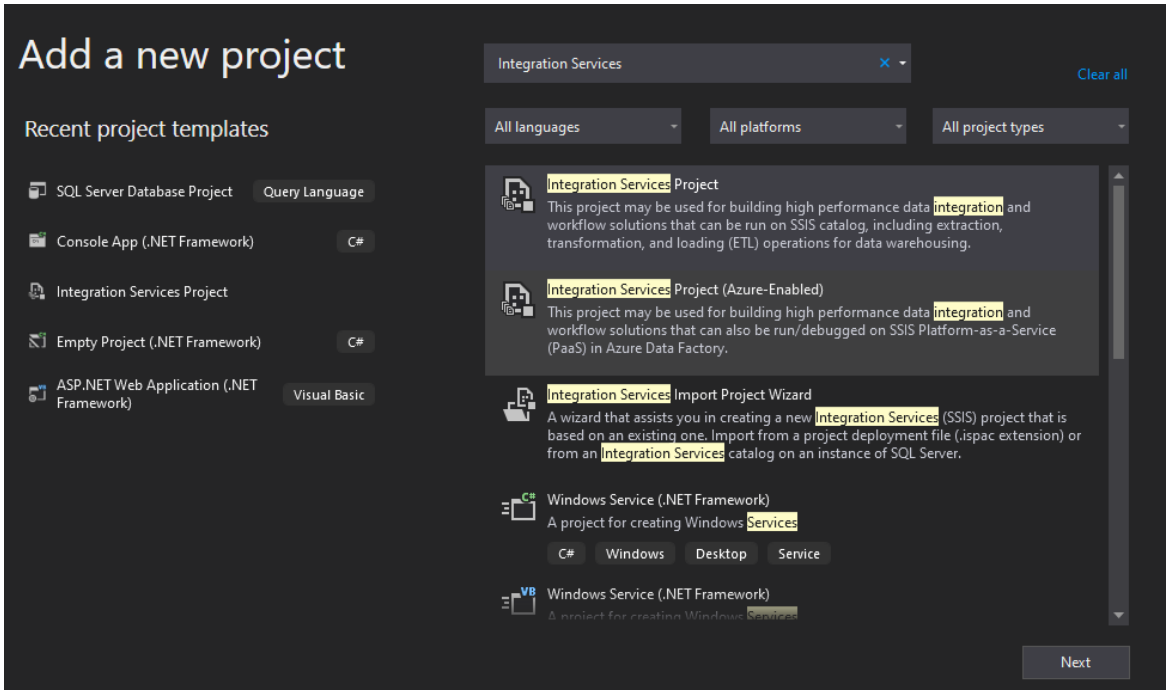
Let's start with the **Database Project**, just search and select:

- 

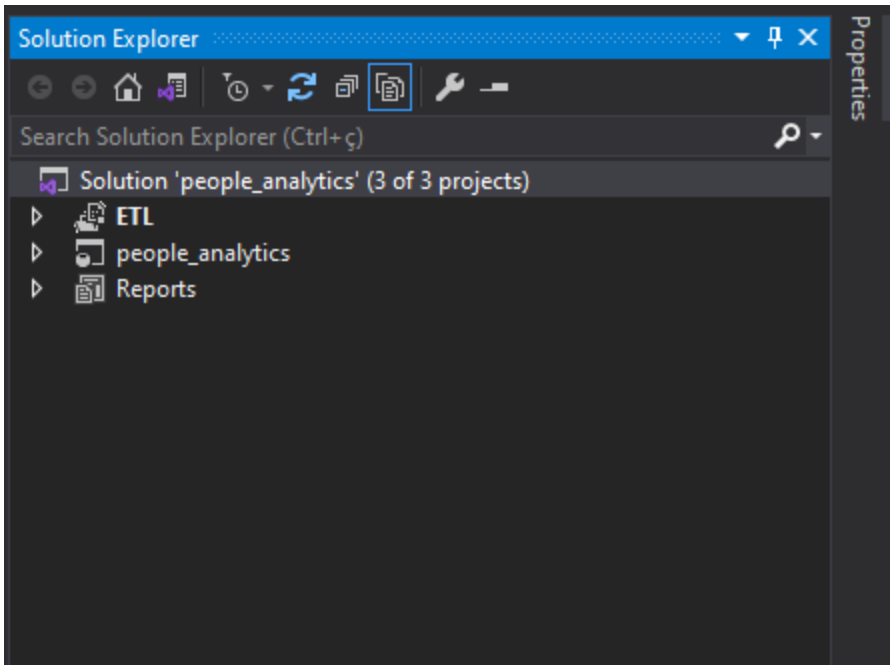
I usually add the name that will be used in the database, but you can just leave it as Database. After adding the name, click on **“Create”**

- 

To add SSIS and SSRS solutions, simply: **right-click on the solution >> Add >> New Project.** Search for the project, add the name and create.

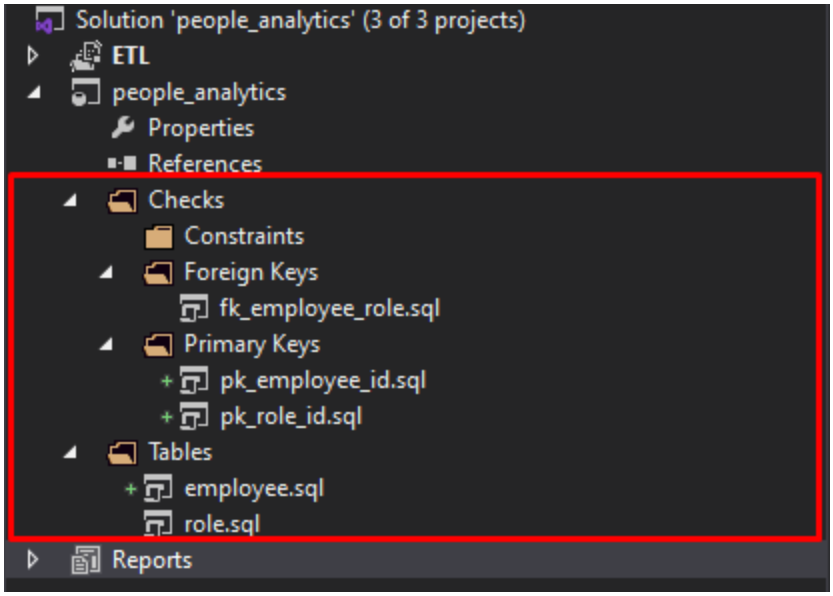
- 

We ended the construction of solutions in the following model:

- 

3.2. Creation of Tables

Let's leave the structure of the tables ready in our solution, for that, we need to analyze the fields of the worksheets and create tables referencing these fields.
I like to organize the database project into folders and inside the folders I add what I need. Example:

- 

Let's create the tables and keys:

- **Employee Table:**
  - ```sql
    CREATE TABLE [dbo].[employee]
    (
        [id] [int] NOT NULL, -- ID do Funcionário
        [name] [nvarchar](100) NOT NULL, -- Nome do Funcionário
        [gender] [nvarchar](25) NOT NULL, -- Sexo do Funcionário
        [hire_date] [date] NOT NULL, -- Data de Admissão
        [role_id] [int] NOT NULL, -- ID do Cargo
        [role_date] [date] NOT NULL, -- Data do Cargo
        [contract_id] [int] NOT NULL, -- ID do Contrato
        [contract_type] [nvarchar](50) NOT NULL, -- Tipo de Contrato
        [marital_status_id] [int] NOT NULL, -- ID do Estado Civil
        [marital_status] [nvarchar](50) NOT NULL, -- Estado Civil
        [education_id] [int] NOT NULL, -- ID da Escolaridade
        [education] [nvarchar](50) NOT NULL, -- Escolaridade
        [birth_date] [date] NOT NULL, -- Data de Nascimento
        [nationality_id] [int] NOT NULL, -- ID da Nacionalidade
        [nationality] [nvarchar](50) NOT NULL, -- Nacionalidade
        [race_id] [int] NOT NULL, -- ID da Raça
        [race] [nvarchar](50) NOT NULL, -- Raça
        [salary] [money] NOT NULL, -- Raça
        [status_id] [int] NOT NULL, -- ID da Situação
        [status] [nvarchar](50) NOT NULL, -- Situação
        [absence_cause] [nvarchar](50) NOT NULL, -- Motivo do Afastamento
        [absence_date] [date] NULL, -- Data do Afastamento
        [work] [nvarchar](50) NOT NULL -- Forma de Trabalho
    )
    ```
    ```

- **Role Table:**
  - ```
    CREATE TABLE [dbo].[role]
    (
        [id] [int] NOT NULL, -- Id do Cargo
        [name] [nvarchar](100) NOT NULL  -- Nome do Cargo
    )
    ```

OBS: As it is a POC, we are going to create a simple model, mirroring only the tables we received, but through the employee table we can identify some dimension tables.

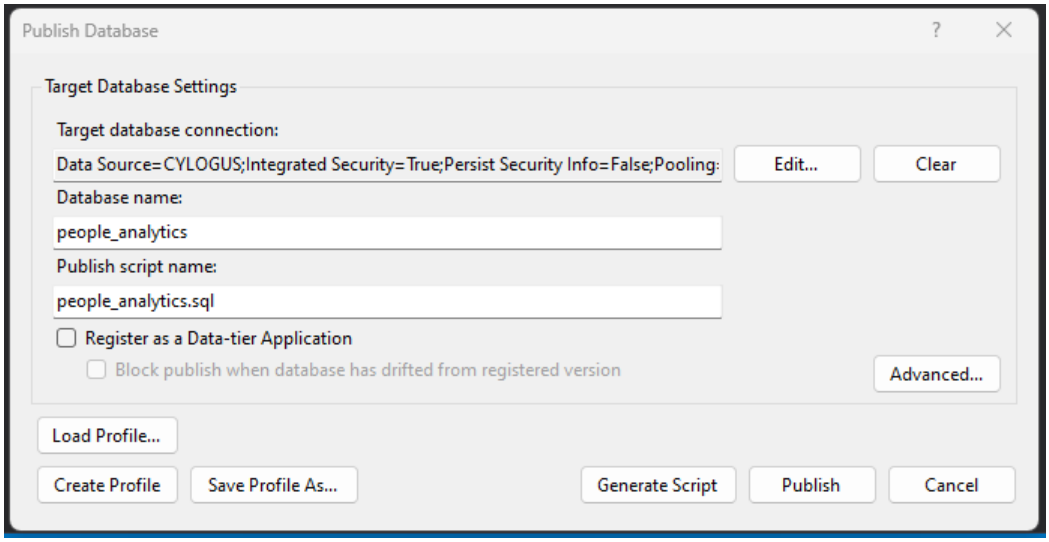
- **Primary Keys**
  - ```
    -- Tabela: role
    ALTER TABLE [dbo].[role]
        ADD CONSTRAINT [pk_role_id]
        PRIMARY KEY ([id]);
    ```
  - ```
    -- Tabela: employee
    ALTER TABLE [dbo].[employee]
        ADD CONSTRAINT [pk_employee_id]
        PRIMARY KEY ([id]);
    ```

- **Foreign Keys**
  - ```
    -- Chave Estrangeira para Role
    ALTER TABLE [dbo].[employee]
    ADD CONSTRAINT [fk_employee_role]
    FOREIGN KEY ([role_id])
    REFERENCES [dbo].[role] ([id]);
    ```

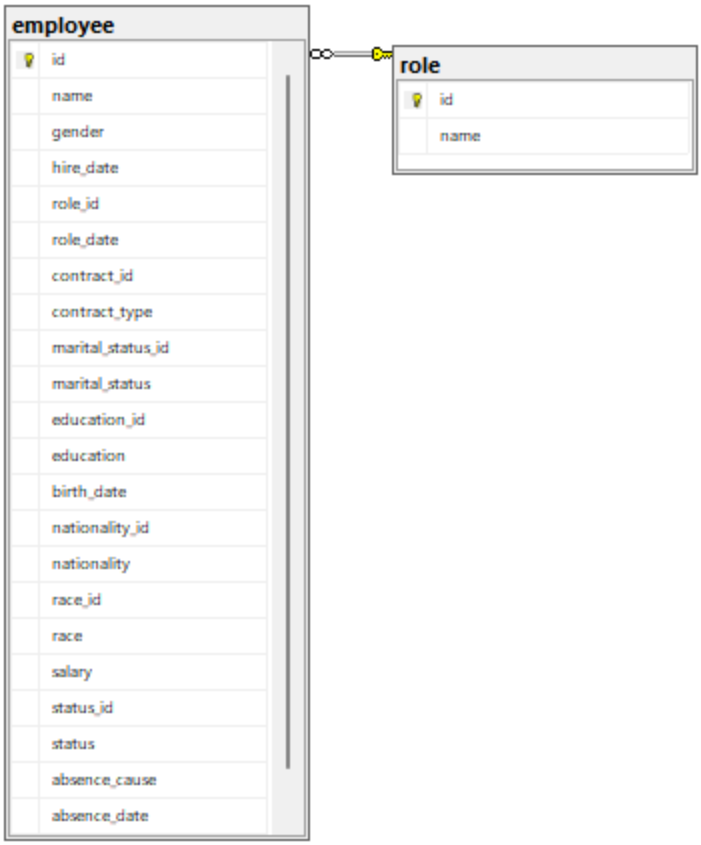
3.3. Deploy to SQL Server

Para realizar o Deploy no SQL Server, seguiremos os seguintes passos:

- **Right click on the database solution >> Publish** (It will do a Build, if there is an error it will not be possible to publish)
- **Target Database Connection** (You configure the connection with SQL)
- **Database Name**
- **Publicação**
  - **Generate Script** (A publishing script will be generated)
  - **Publish** (Will publish direct)



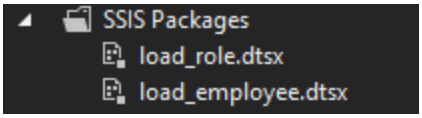
After publishing, check **SQL Server.**



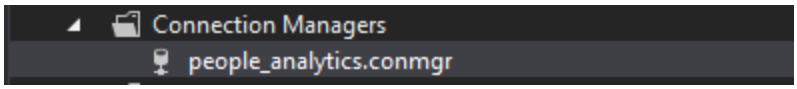
3.4. ETL

Let's make two packages:

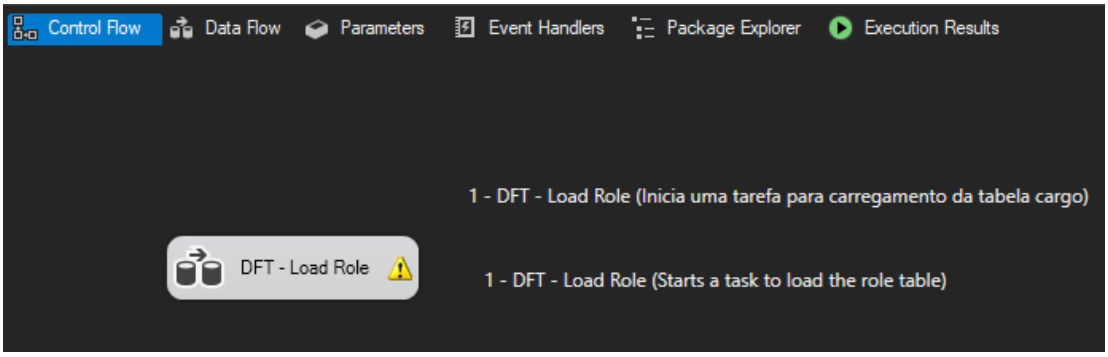
- load_employee (Loads the employee table)

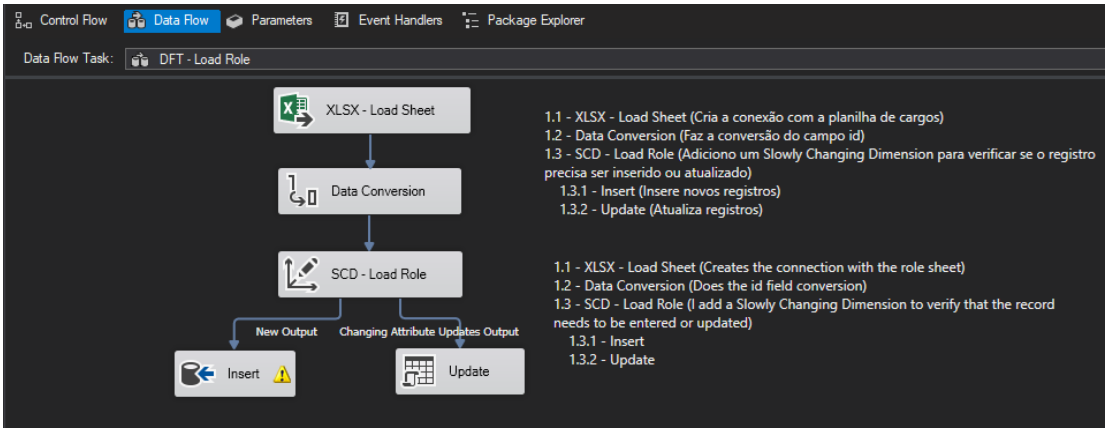
- load_role (Loads the role table)
  - 

Official connection for all packages:

- 

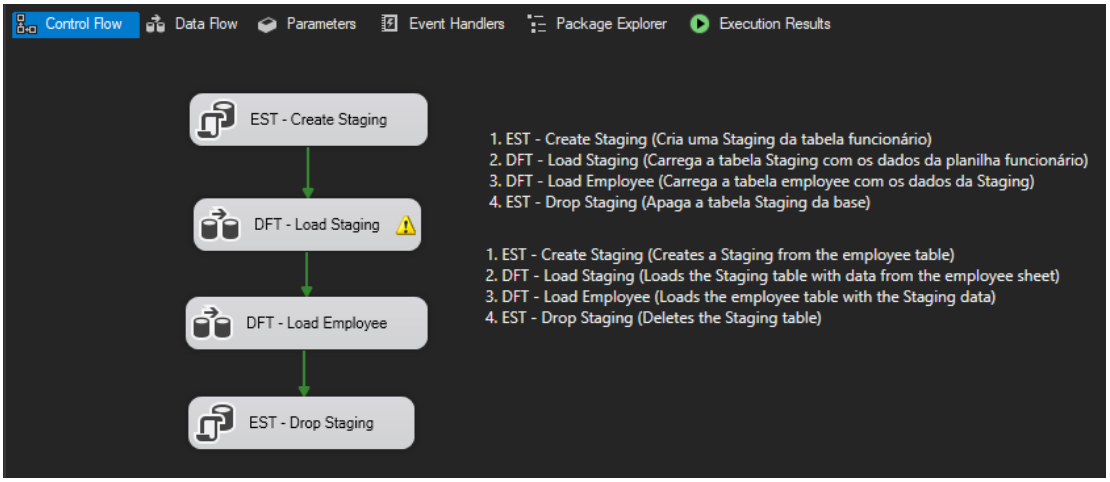
load_role - I focused on a more practical and direct approach, to avoid errors I added an SCD directly from the spreadsheet.

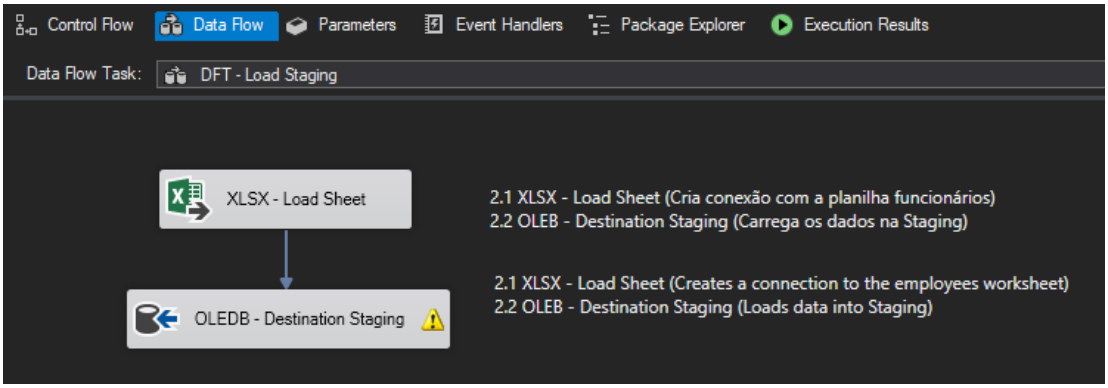
- **Control Flow**
  - 

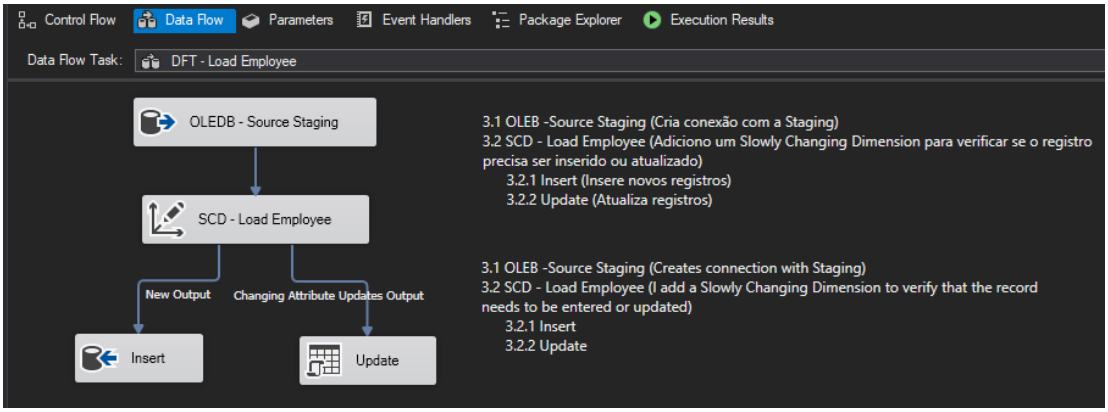
- **Data Flow**
  - 

load_employee - I applied an approach that I really like to use and I believe to be very functional (not so much in this case, but I wanted to share). I create a Staging at the beginning of the Job and load the worksheet in this Staging, the main reason is that we have a task (Data Flow Task) focused only on handling the Staging.

- **Control Flow**

- 

- **Data Flow**
  - 

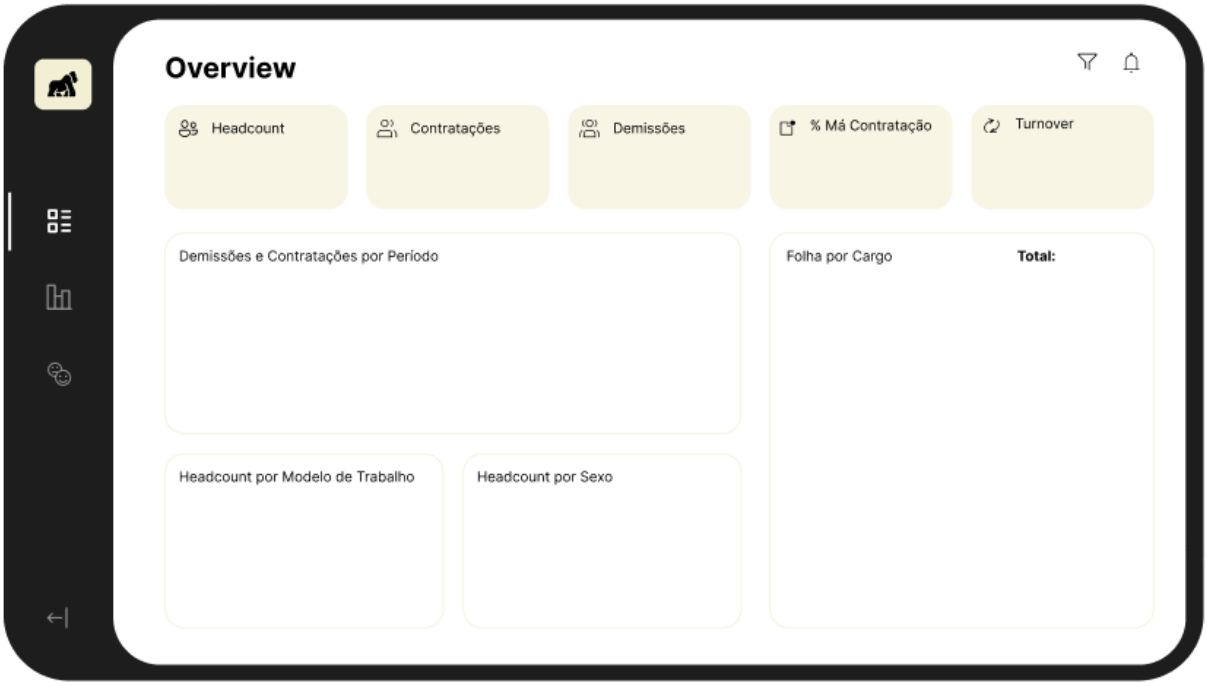
  - 

4. BI

4.1. Dashboard

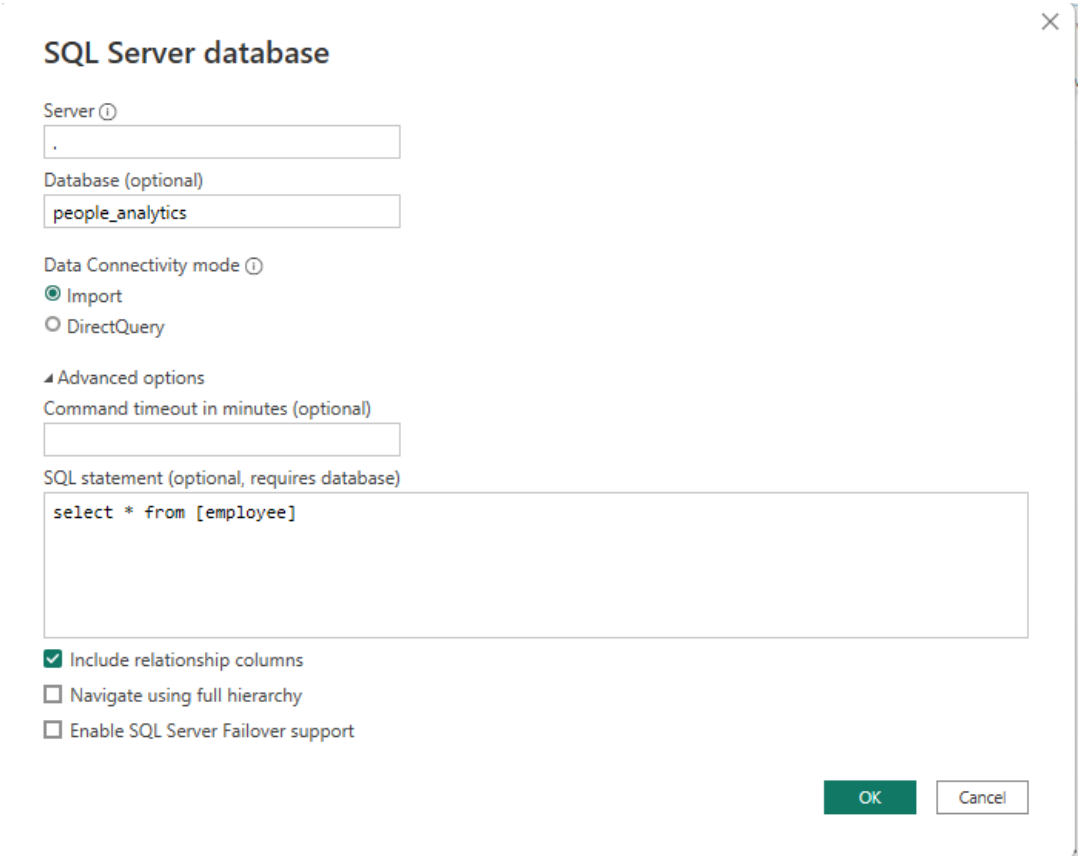
Template

The template used was created in Figma, **everything is available in the project files.**



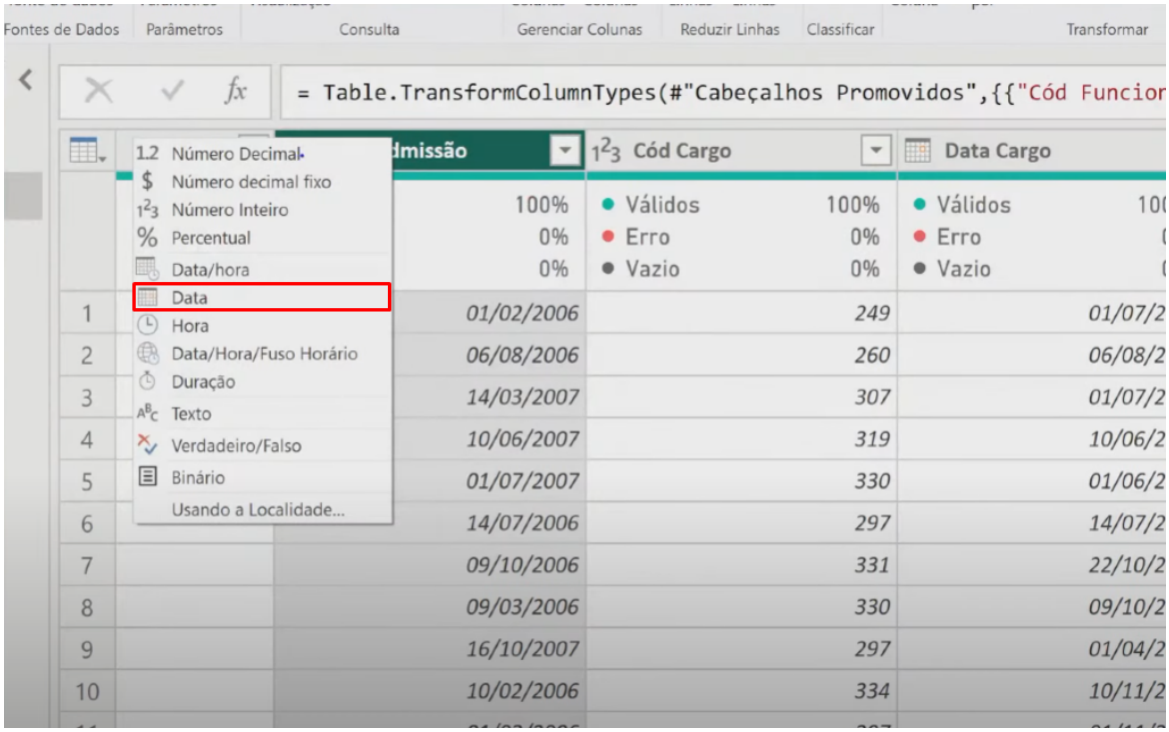
Connection

In Power BI, let's use the SQL Server Database connection and load the two tables:



Transformations

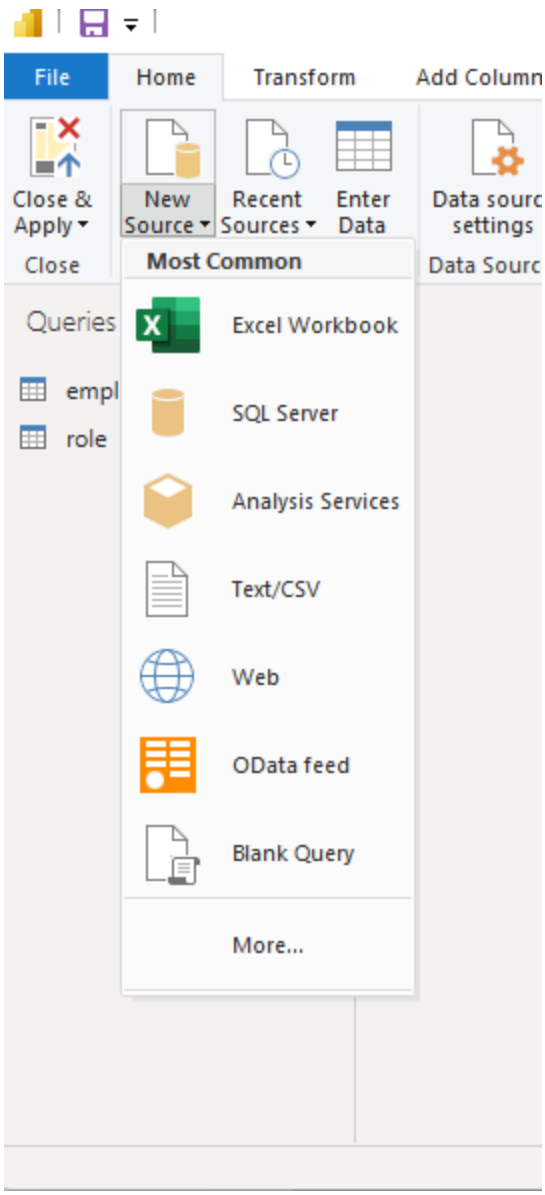
Just add **"Date"** icon in all tables which are with **"Date/Time"**

- 

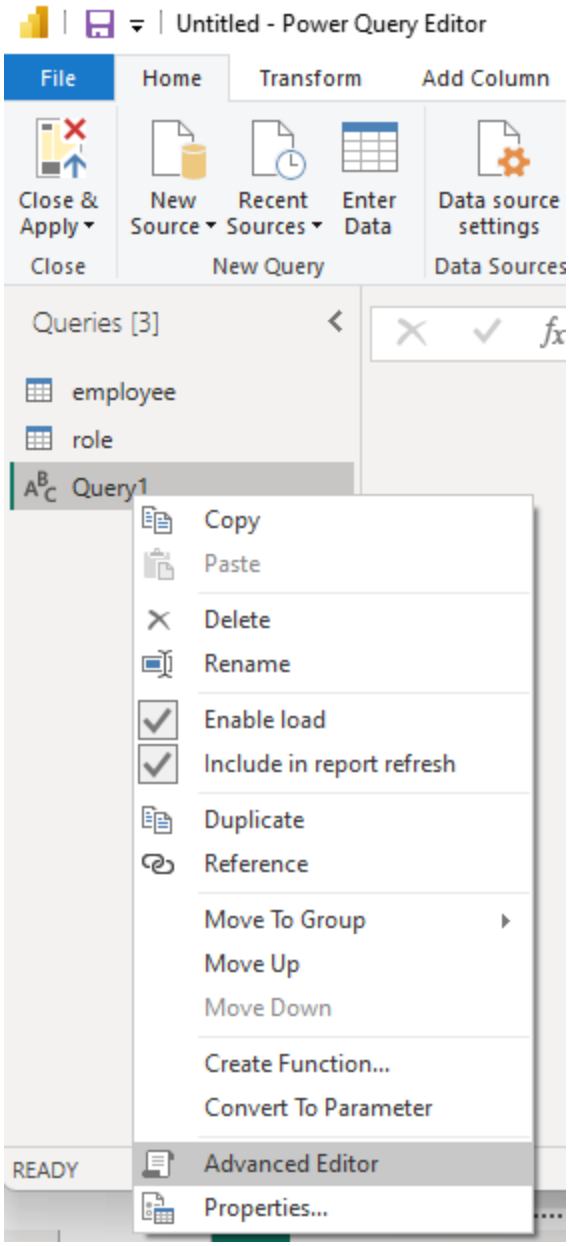
Calendar Table

https://github.com/minhasplanilhas/PowerBI/blob/master/dCalendarioCompleto

Click in **New Source + Blank Query**



Right click + Advanced Editor

- 

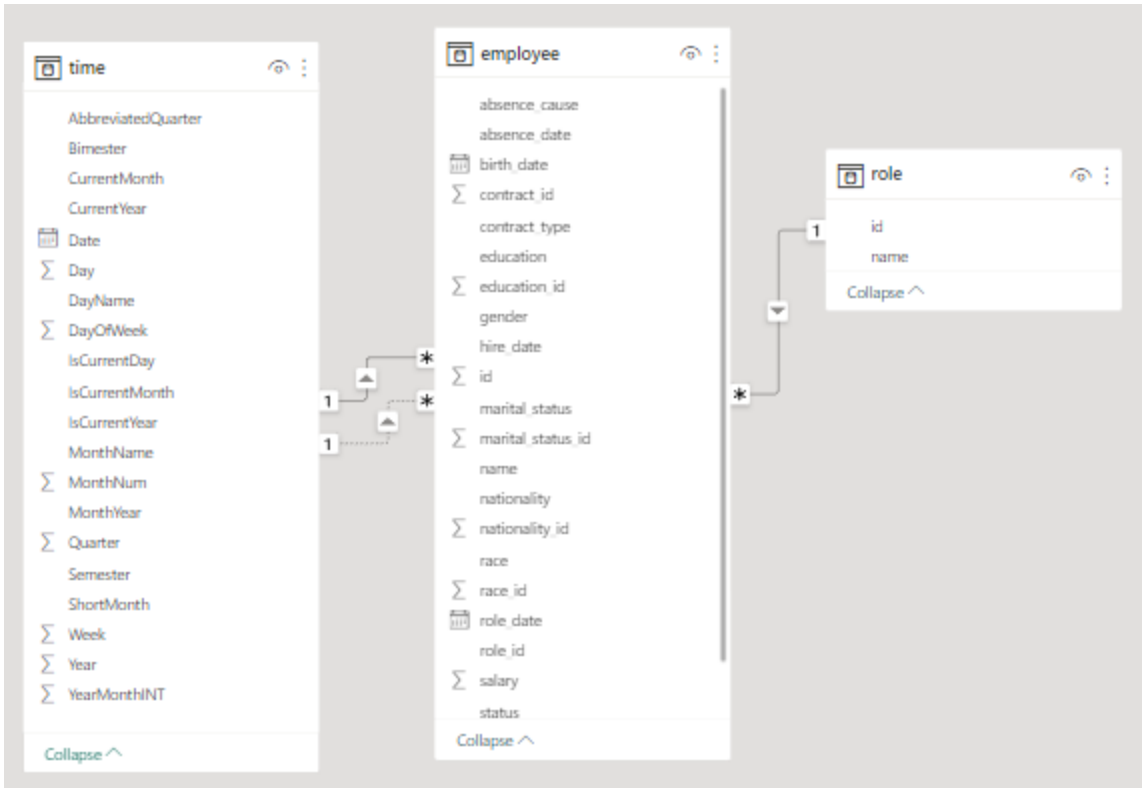
Paste the code shared by the link above and change only the print part:

In my example I change the code of the calendar table in English, you can find it in the files: tableTimePowerQuery.txt

Relationships

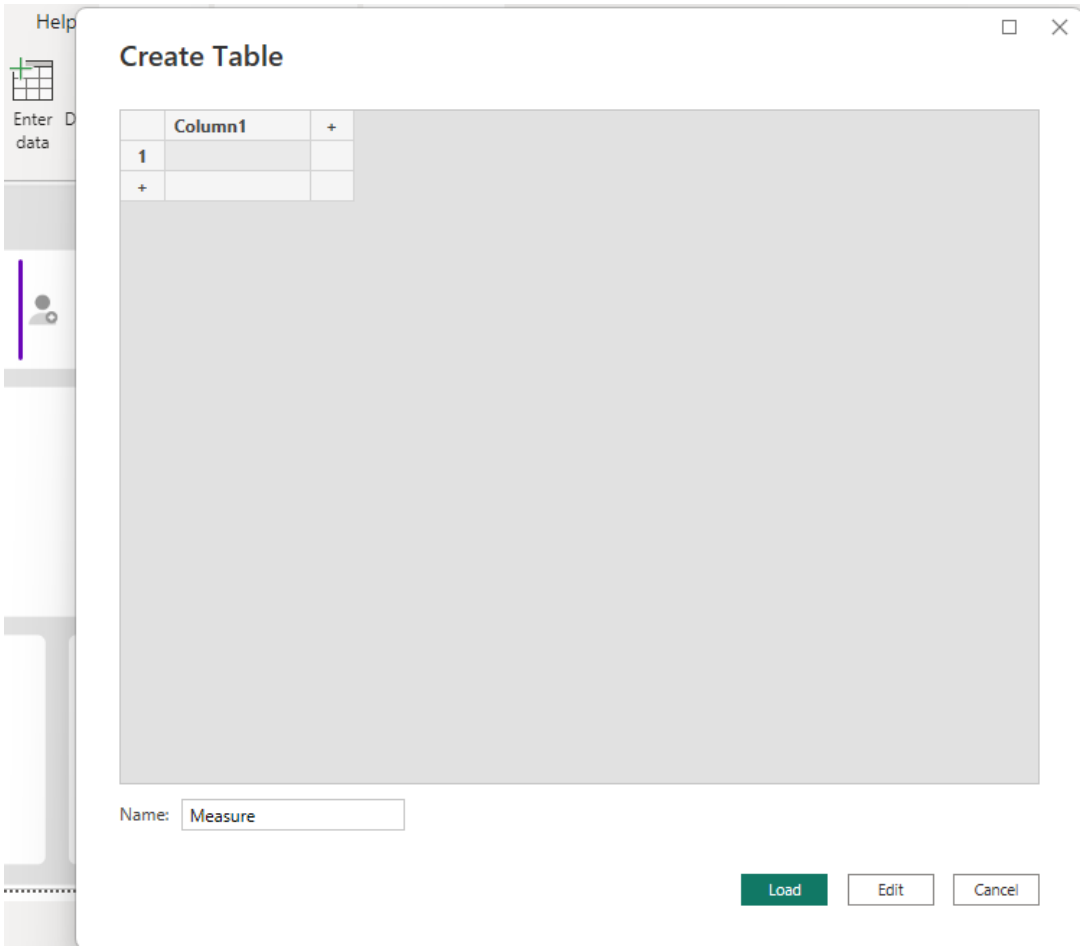
employee[role_id] --> role[id]

employee[hire_date] / employee[absence_date] --> time[Date]



Measures DAX

Create a table called **Measure** to organize the DAX measures (Measures). **Enter Data + Load**



Measure 01 - Total Hiring

```
Total Hiring = COUNTROWS(employee)
```

Measure 02 - Layoffs

```
layoffs =
CALCULATE(
    [Total Hiring],
    employee[status_id] = 7,
    USERELATIONSHIP('time'[Date], employee[absence_date])
)
```

Measure 03 - Headcount (Accumulated from employees)

```
Headcount =
VAR _Active =
CALCULATE(
    [Total Hiring] - [layoffs],
    FILTER(ALL('time'),
    'time'[Date] <= MAX('time'[Date])
    )
)
RETURN
IF(_Active <> 0, _Active)
```

Measure 04 - Tempo de Retenção (Employee retention is a strategy companies use to prevent or decrease employee turnover.).

Note: Let's use the DAX in a new column in the employees table

```
retention_time = DATEDIFF(employee[hire_date], employee[absence_date], DAY)
```

Measure 05 - Total Bad Hires (Each company analyzes its measurement as a standard, in this model retention below 60 was added)

```
Tot Bad Hire =
CALCULATE(
    [Total Hiring],
    employee[retention_time] < 60,
    employee[retention_time] <> BLANK())
```

Measure 06 - % Bad Hires (We just calculated the percentage to use as a measure)

```
% Bad Hire =
DIVIDE([Tot Bad Hire], [Total Hiring])
```

Measure 07 - Turnover (Turnover is the employee turnover rate, which measures the number of employees leaving an organization over a period of time)

```
Turnover =
VAR _Headcount =
CALCULATE(
    [Headcount],
    PREVIOUSMONTH('time'[Date])
)
VAR _Calculation = DIVIDE([Total Hiring] + [layoffs], 2)
RETURN
DIVIDE(_Calculation, _Headcount)
```

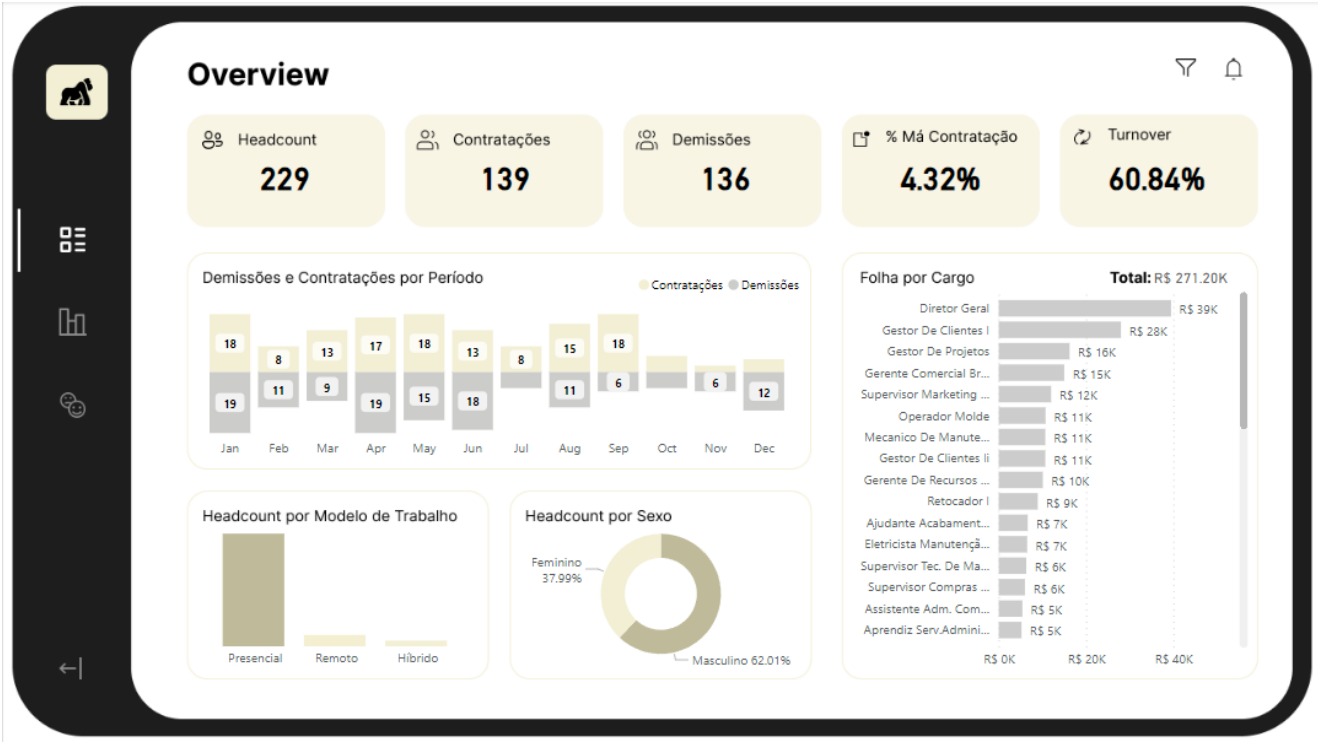
Measure 08 - Payroll Value (A total of how much is spent on each employee, excluding those fired)

```
Payroll Value =
CALCULATE(
    SUM(employee[salary]),
    employee[status_id] <> 7)
```

Measure 09 - Layoffs Chart (A trick to make the value negative, used for visualization purposes)

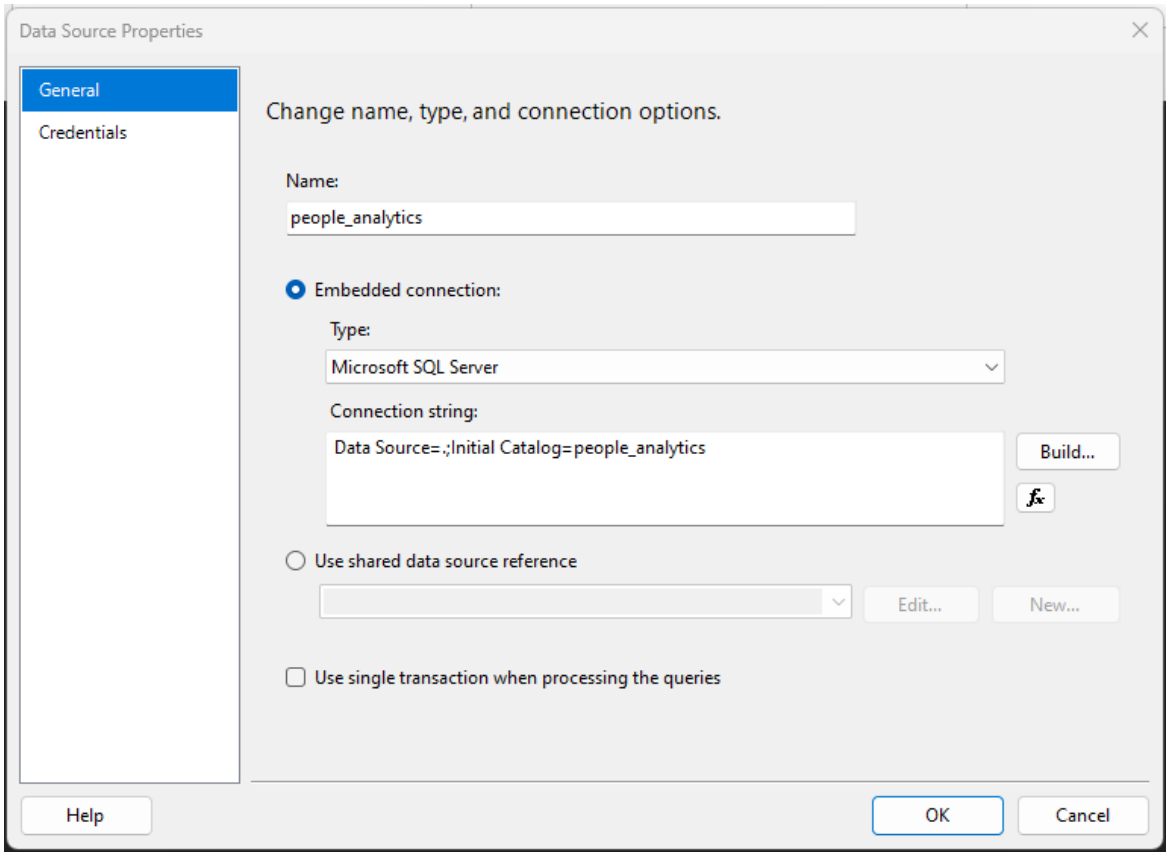
```
Layoffs Chart = - [layoffs]
```

Final Dashboard

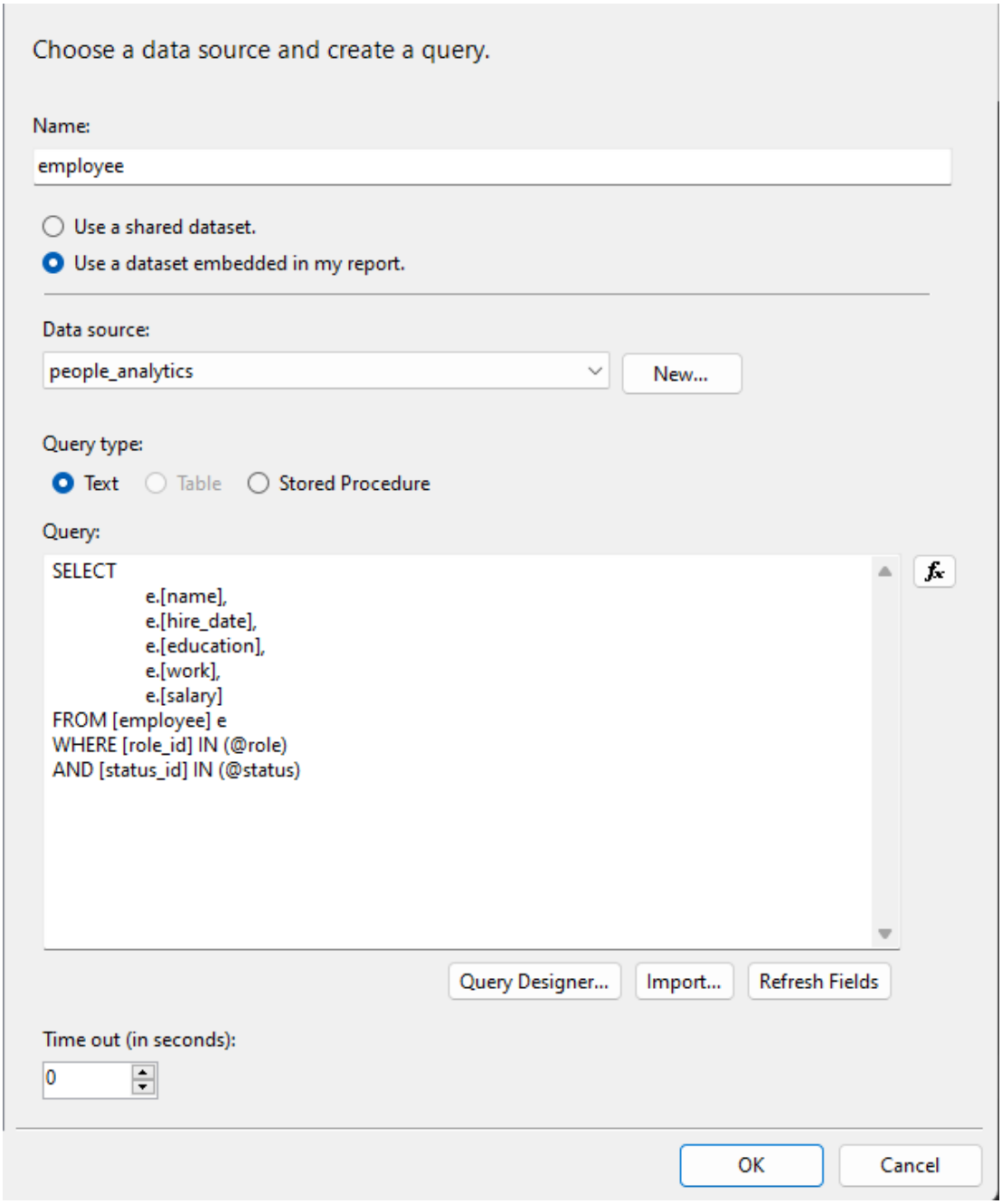


4.2. Report

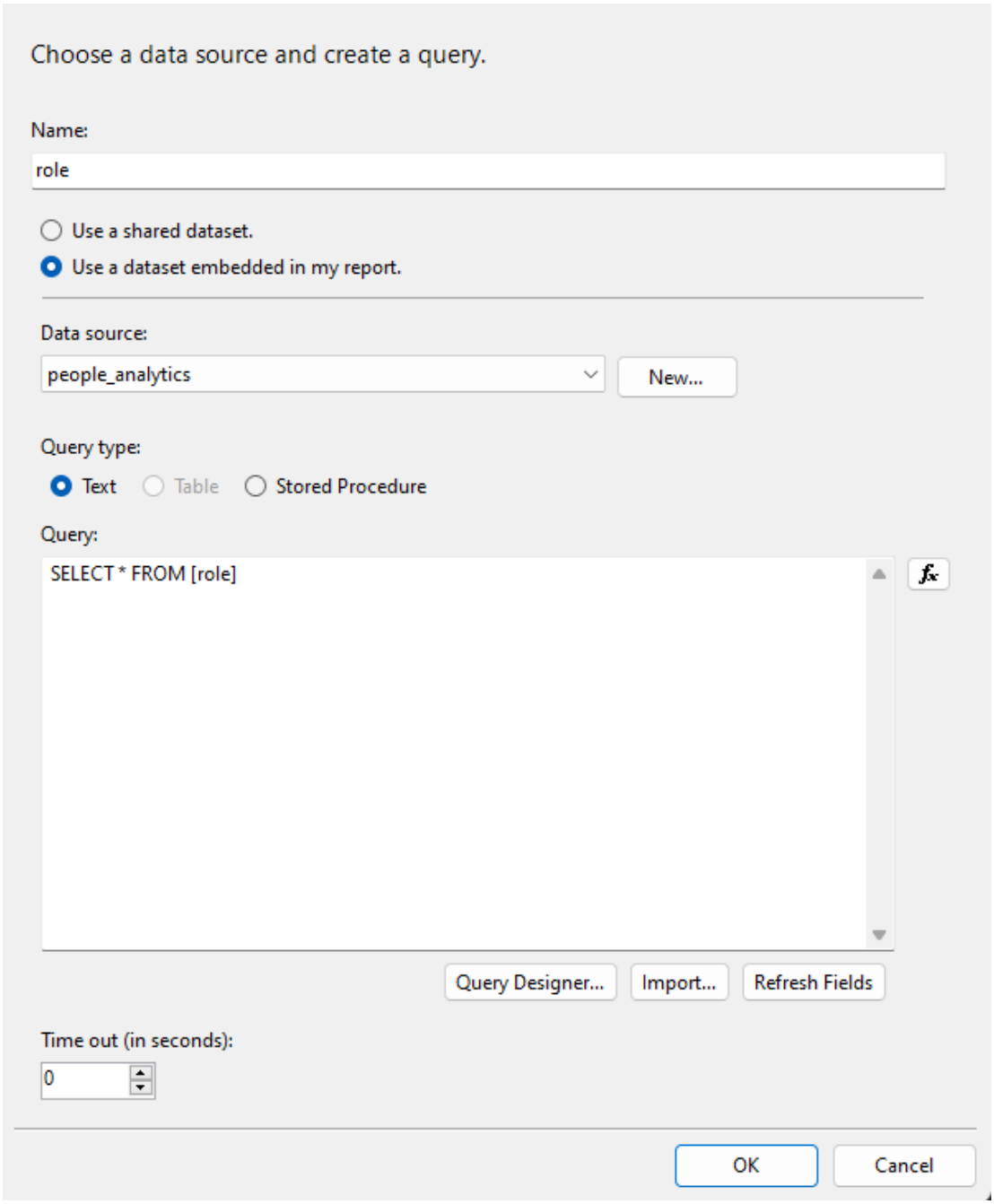
Data Sources



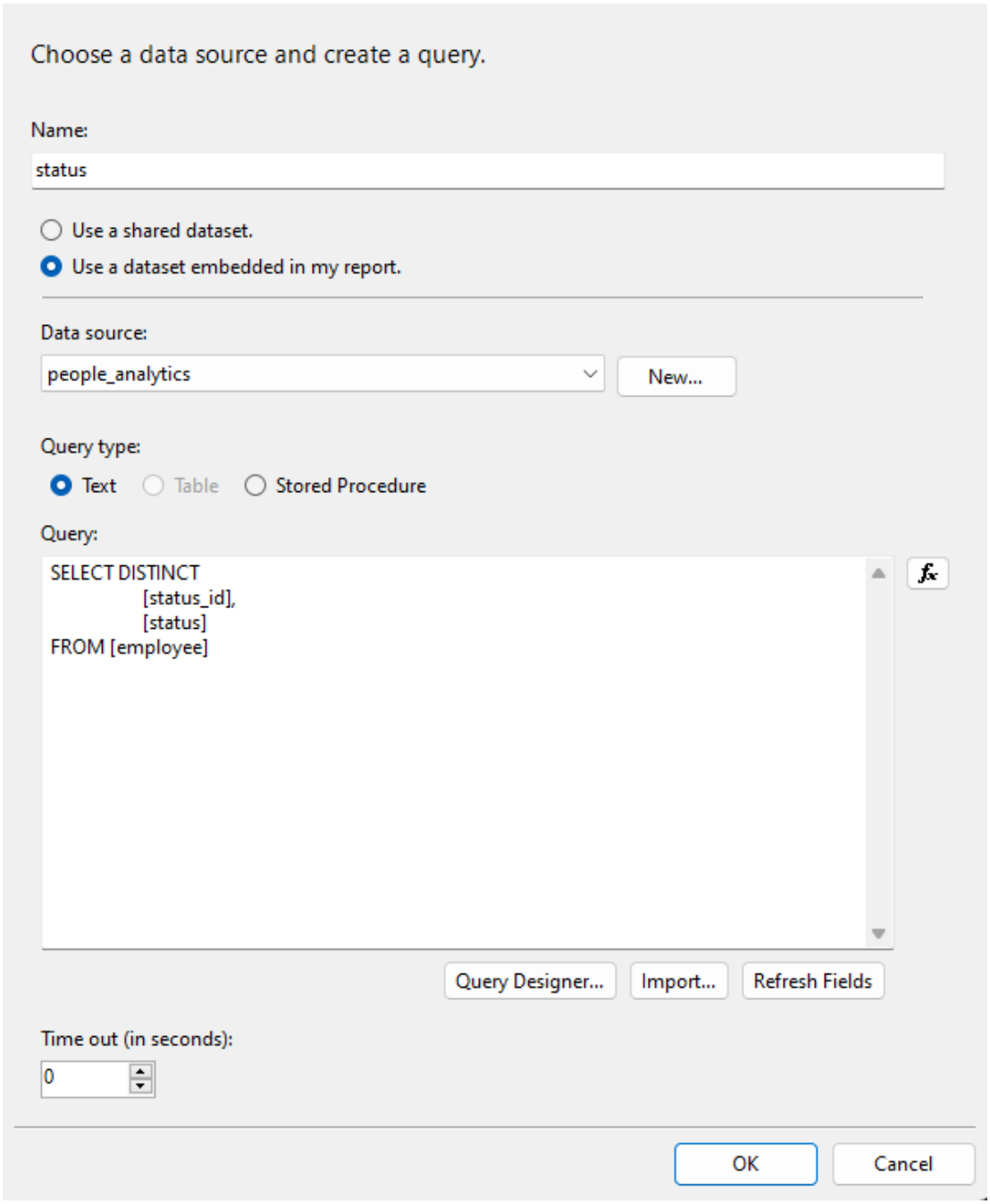
Datasets

- employee
  - 

- role

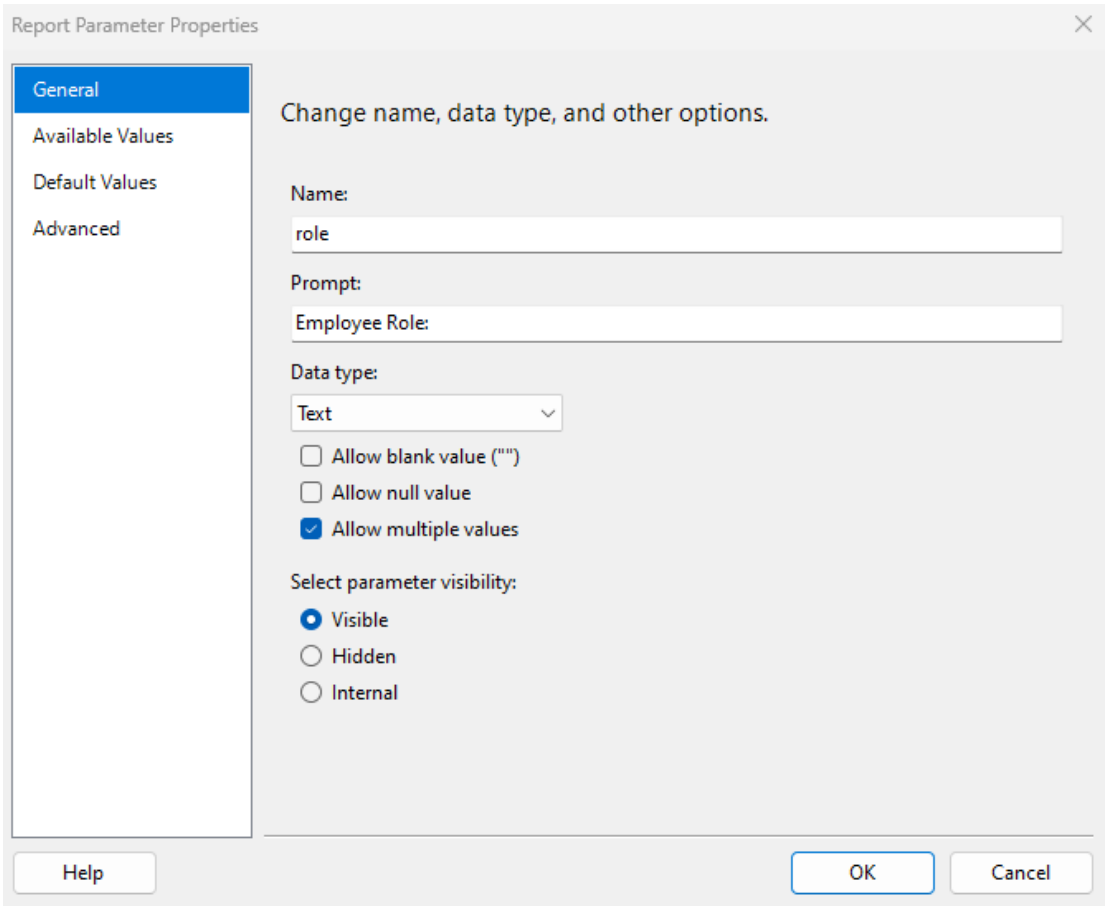
- 
Choose a data source and create a query.
Name:
role
Use a shared dataset.
Use a dataset embedded in my report.
Data source:
people_analytics
New...
Query type:
Text
Table
Stored Procedure
Query:
SELECT * FROM [role]
Query Designer...
Import...
Refresh Fields
Time out (in seconds):
0
OK
Cancel

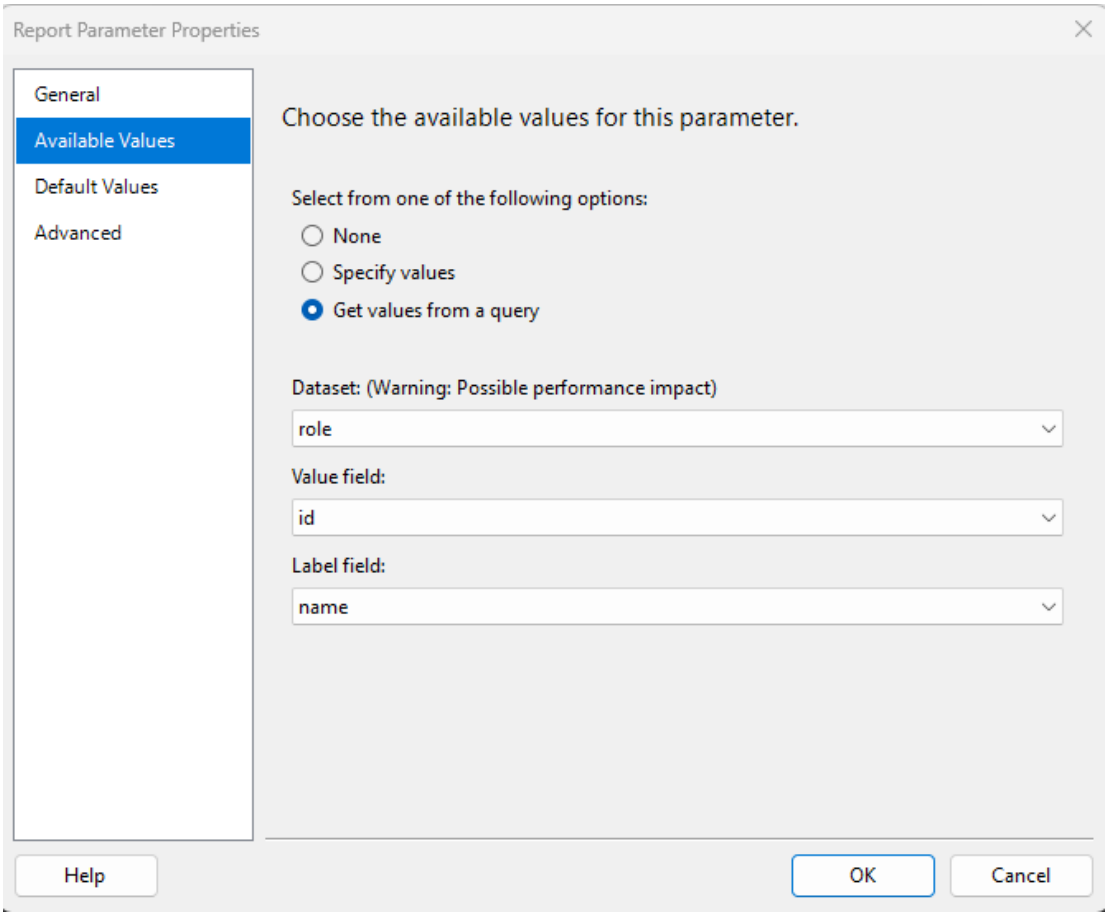
- status

- 

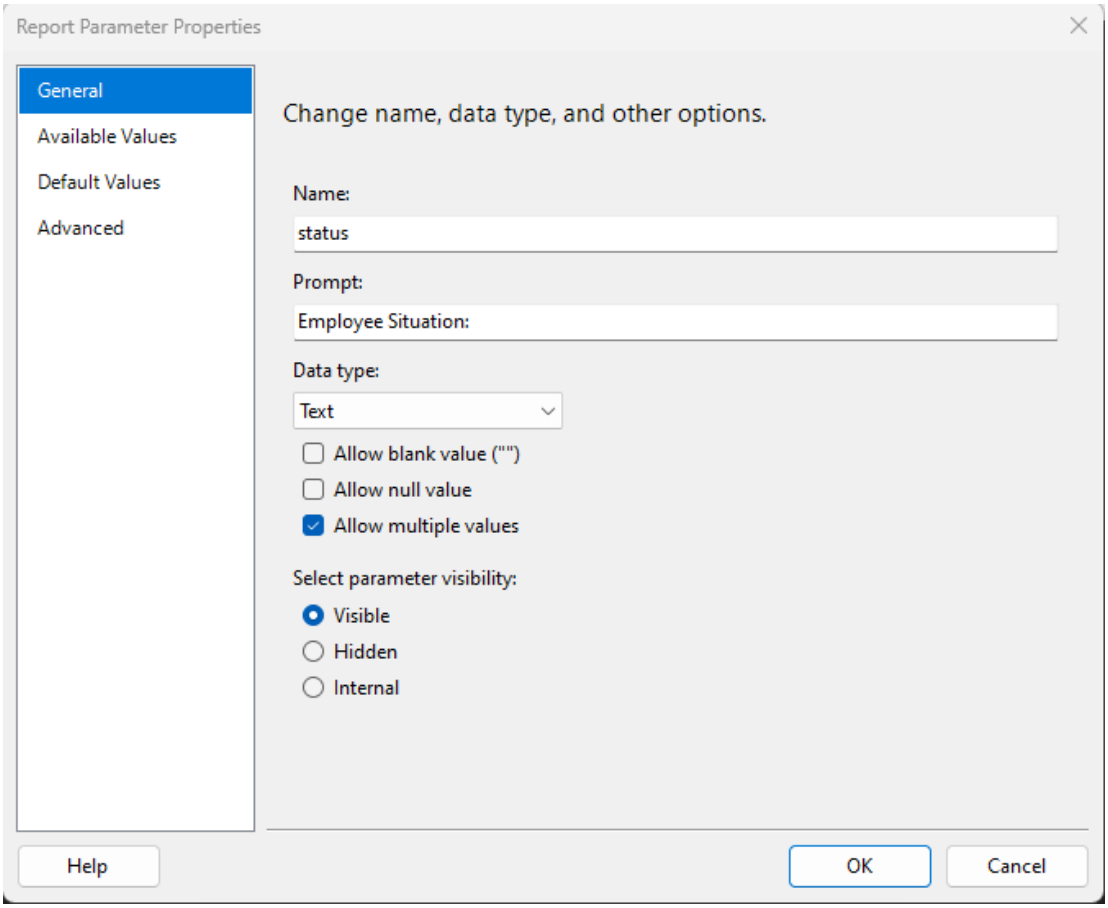
Parameters

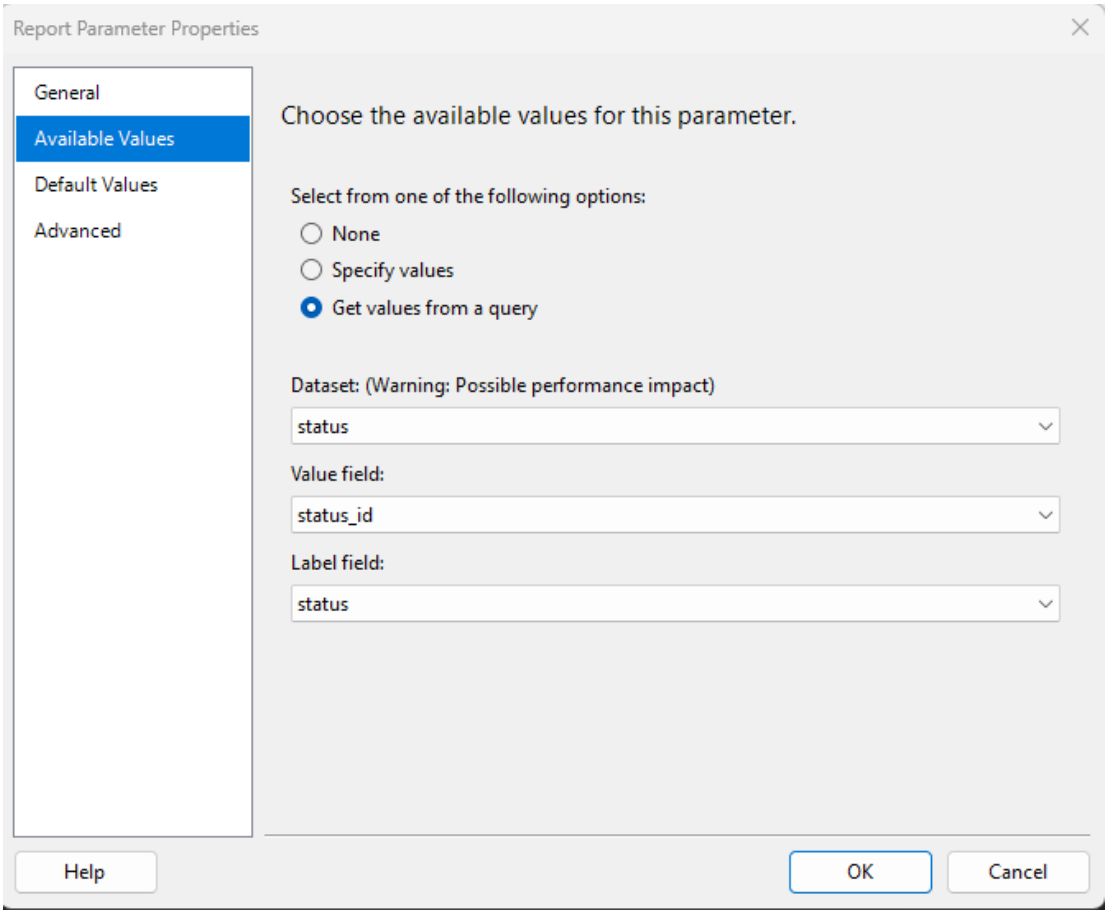
- Employee Role


Report Parameter Properties
General
Available Values
Default Values
Advanced
Change name, data type, and other options.
Name:
role
Prompt:
Employee Role:
Data type:
Text
Allow blank value ("")
Allow null value
Allow multiple values
Select parameter visibility:
Visible
Hidden
Internal
Help
OK
Cancel

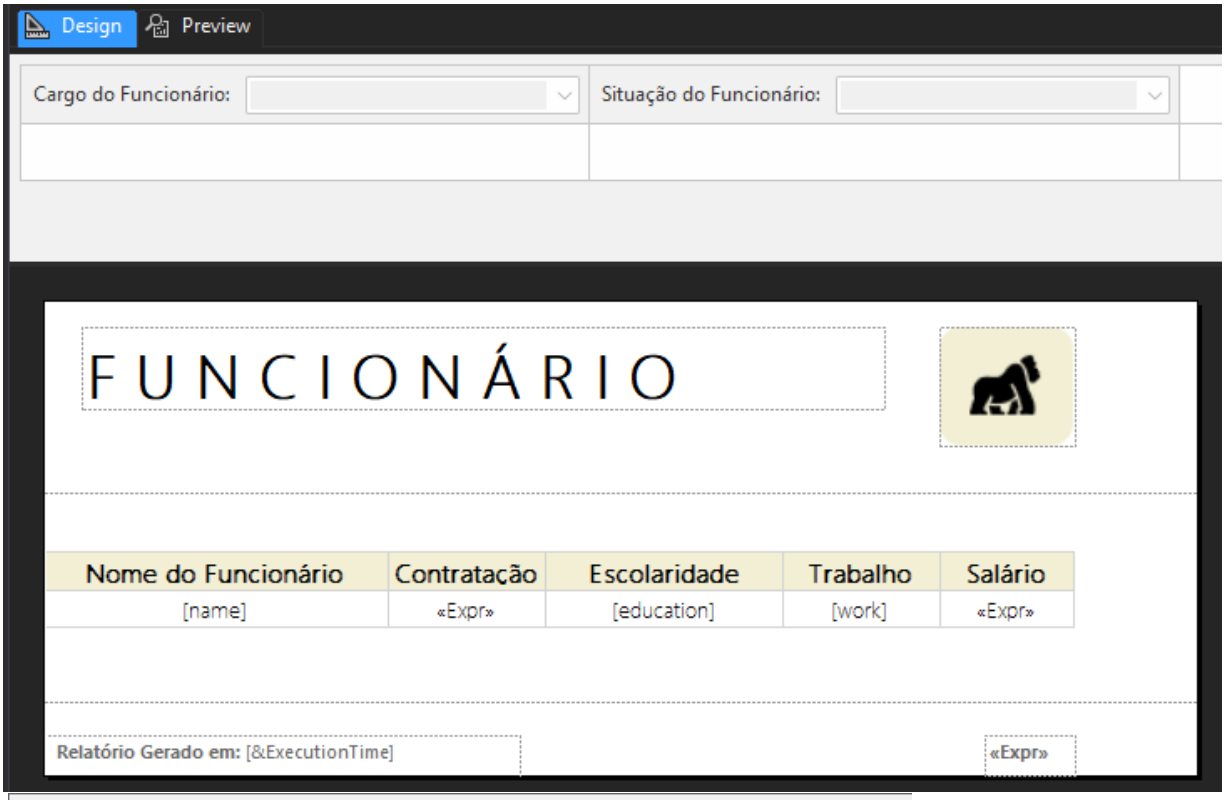
- 

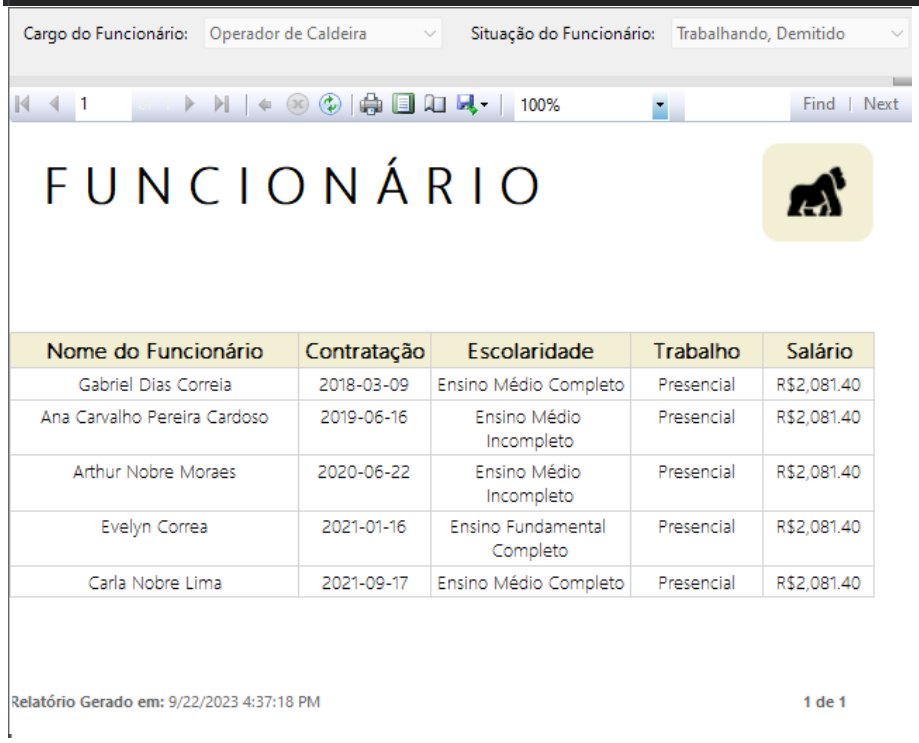
- Employee Status


Report Parameter Properties
General
Available Values
Default Values
Advanced
Change name, data type, and other options.
Name:
status
Prompt:
Employee Situation:
Data type:
Text
Allow blank value ("")
Allow null value
Allow multiple values
Select parameter visibility:
Visible
Hidden
Internal
Help
OK
Cancel

- 

Final Report





4.2. Deploy Reports e Dashboards

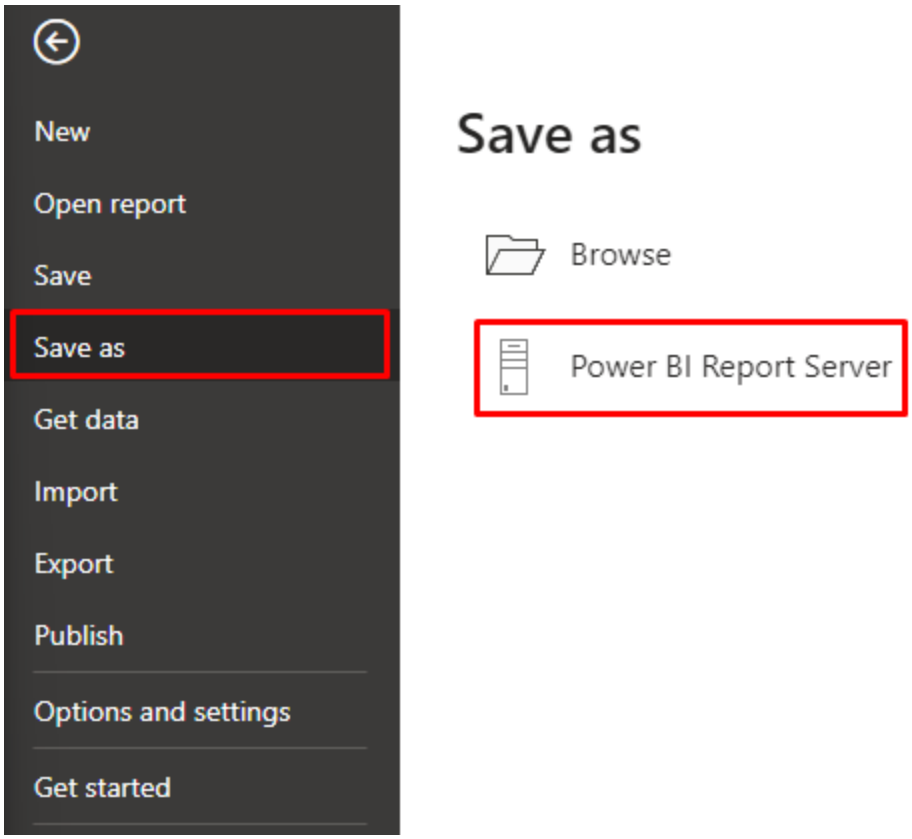
First of all, you need to setup Power BI Report Server, you can deploy both in Power BI Service as well, but our example is focused on something cost 0 or less.

Good link: *https://spgeeks.devoworx.com/install-and-configure-power-bi-report-server/*

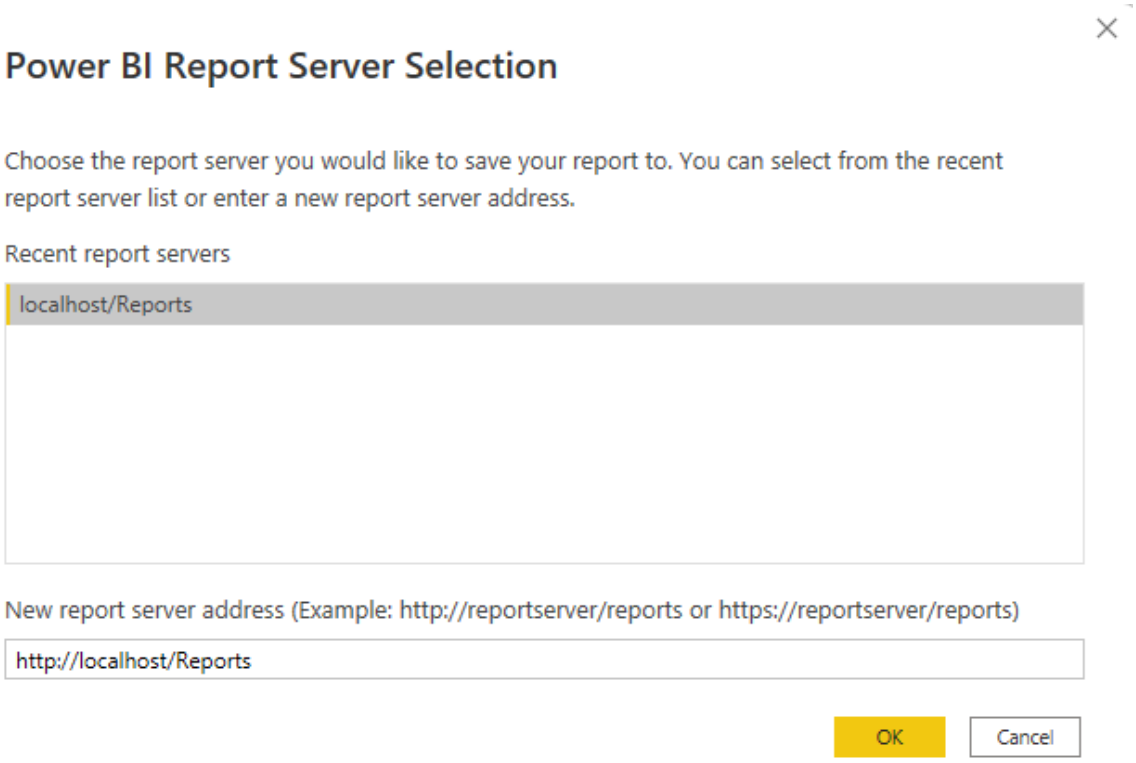
OBS: From here we will use the model in Portuguese

Deploy the Dashboard

Click in File >> Save as >> Power BI Report Server

- 

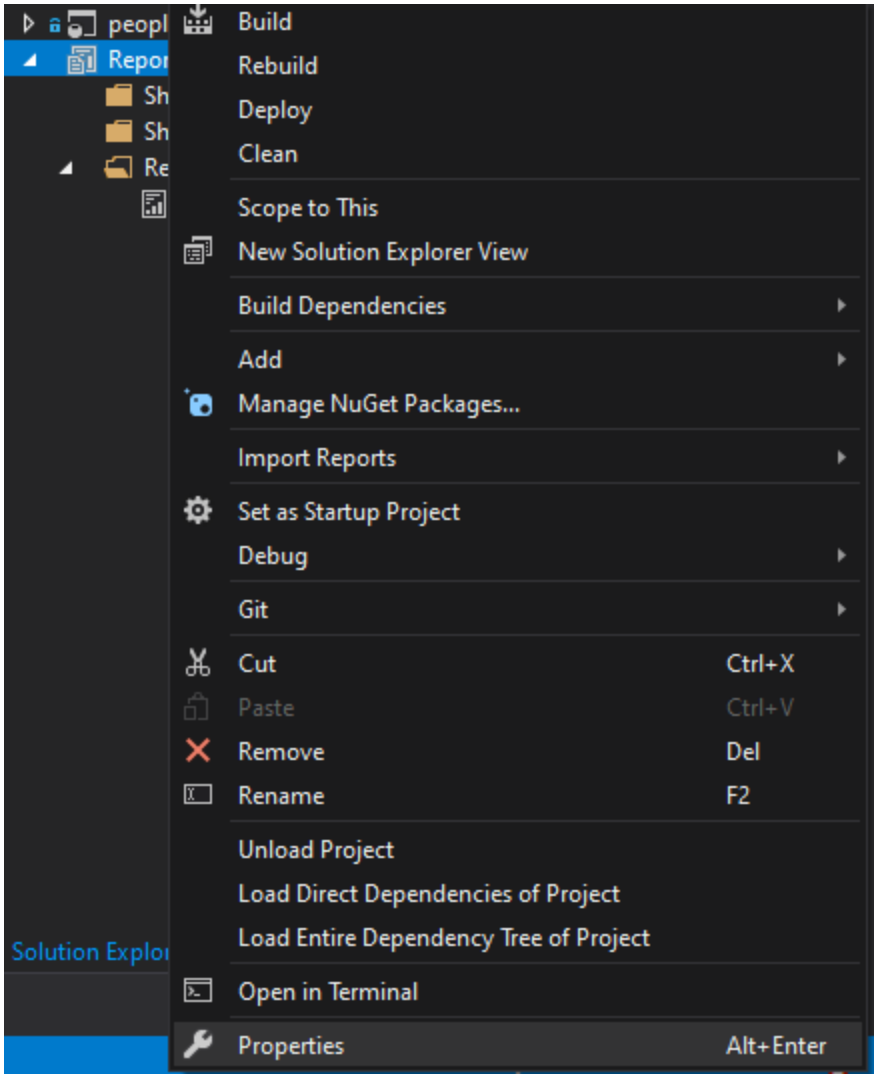
Add the Report Server URL >> Ok >> Ok

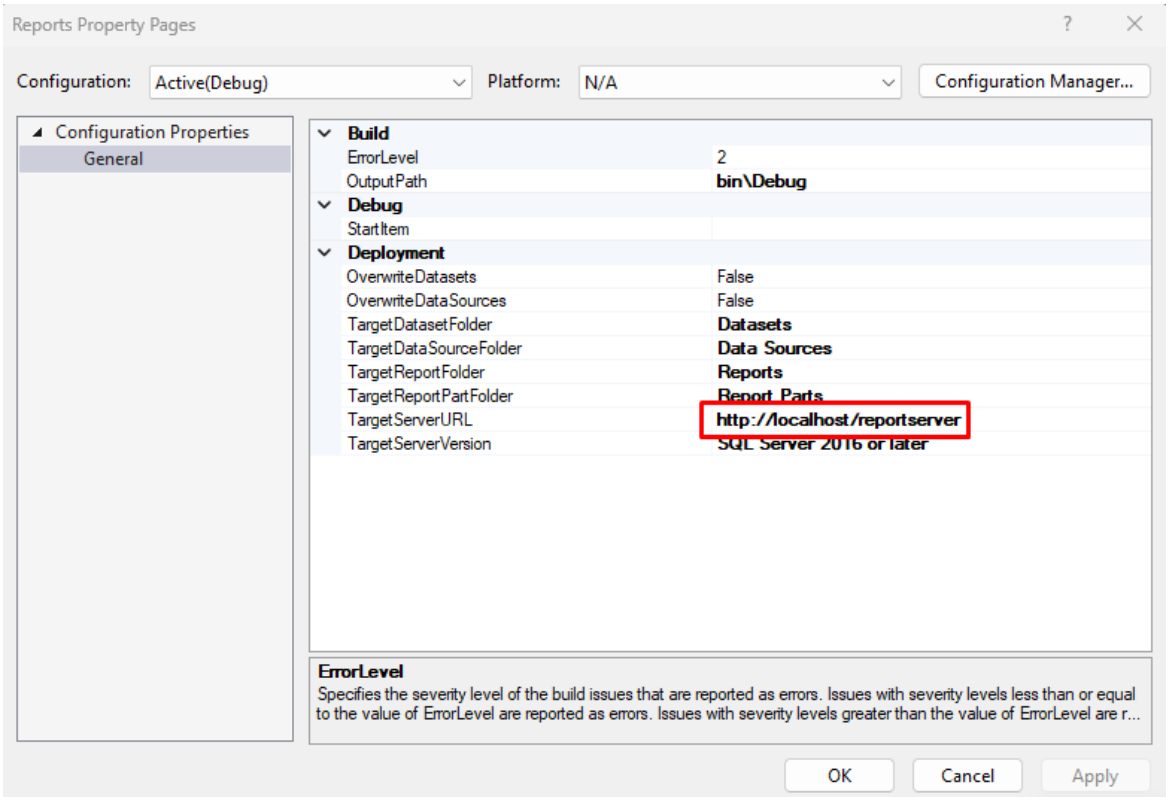


-

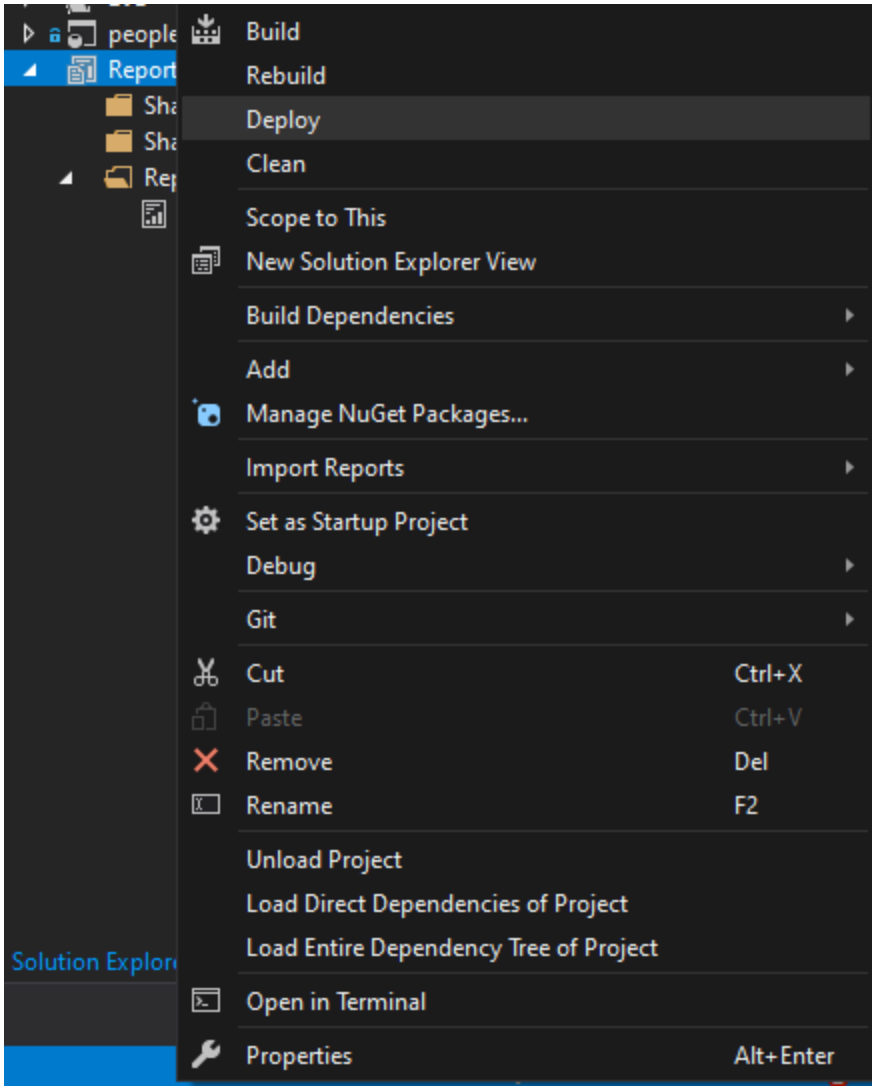
Deploy the Reports

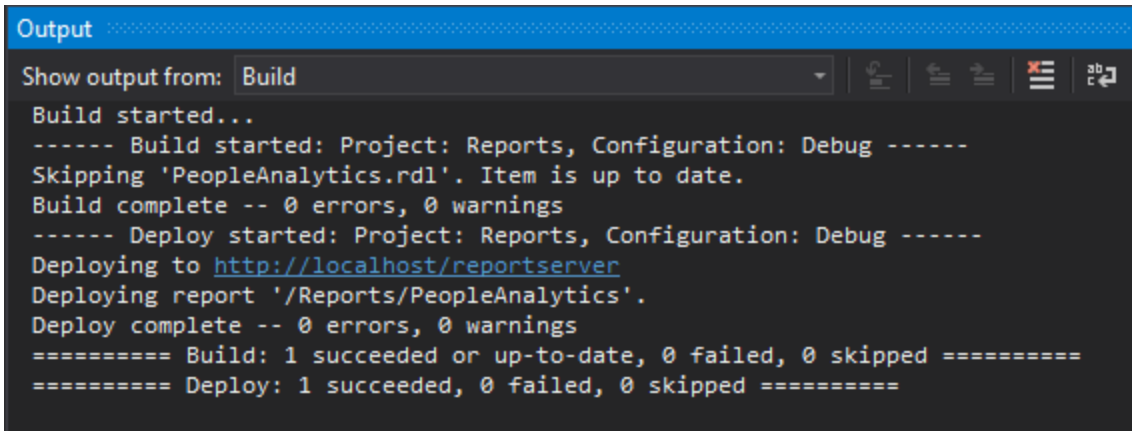
Configure the Report Server URL first. **Right Click in SSRS solution >> Properties >> Target Server URL**

- 
Build
Rebuild
Deploy
Clean
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Import Reports
Set as Startup Project
Debug
Git
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies of Project
Load Entire Dependency Tree of Project
Open in Terminal
Properties Alt+Enter
Solution Explor

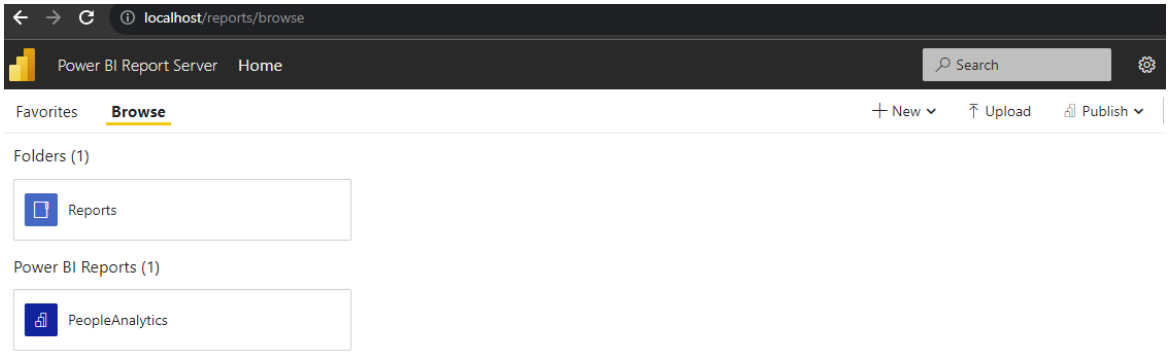


- **Right Click in SSRS solution >> Deploy >> Wait build and deploy finish**

- 

- 

Check Report Server



4.4. Embed Reports and Dashboards

In this step I want to show how we can embedded reports and Dashboards into our website or share them with people in your organization.

Of course, in the Report Server we have less functionality and less support from Microsft, but anyway, I believe that a good part of the companies only use the basics and that basics the Report Server delivers very well.

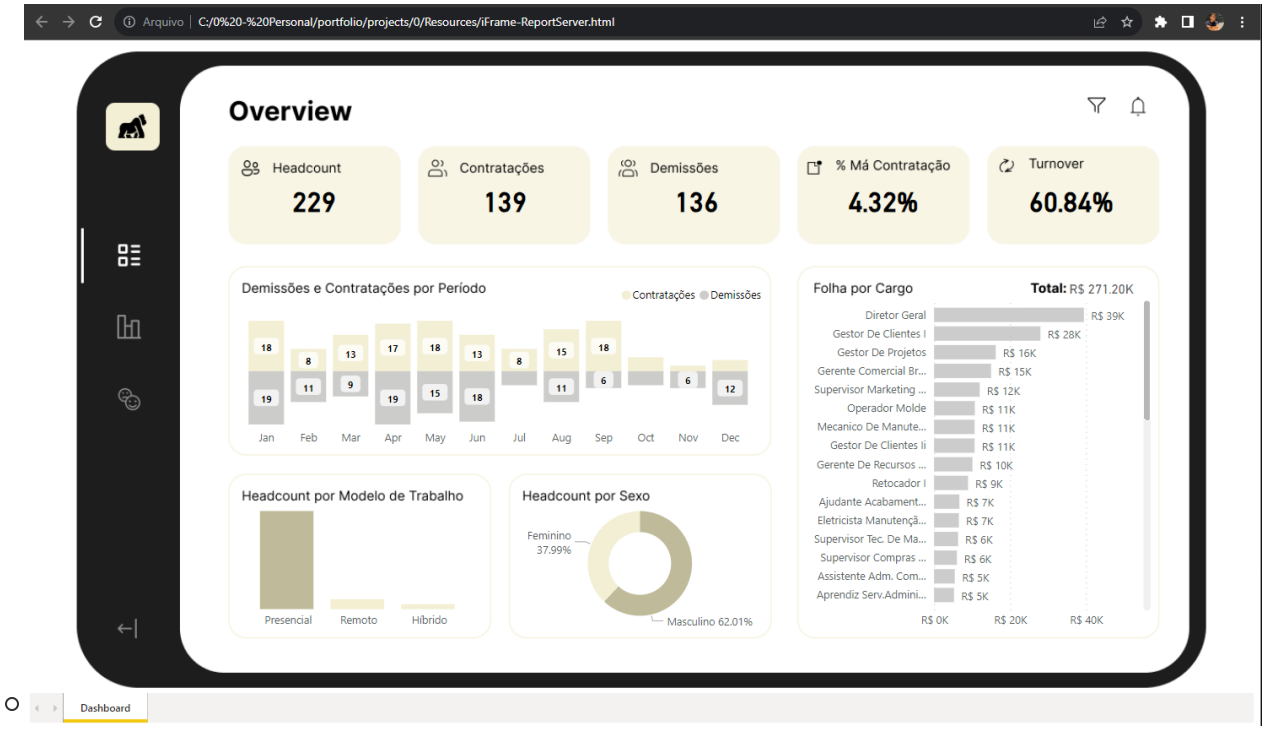
Embed Dashboard

- URL: http://cylogus/Reports/powerbi/PeopleAnalytics**?rs:embed=true**
- iFrame example:

```
<html>
<head>
    <title> "Teste PBIRS" </title>
</head>
<body>

<iframe src="http://localhost/Reports/powerbi/PeopleAnalytics?rs:embed=true" width="100%" height="800" frameborder="0"><iframe>

</body>
</html>
```



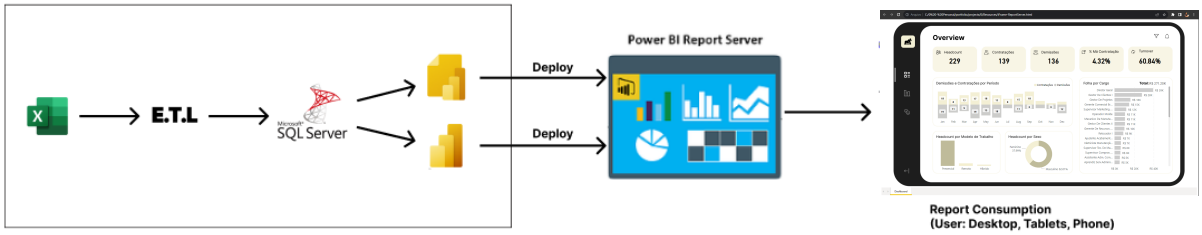
The same logic is used in the Report

You can embed this in an Iframe on your website or share the report server URL. You can also customize the report server with your branding, such as a server for your organization.

Link below for brand: https://learn.microsoft.com/en-us/sql/reporting-services/branding-the-web-portal?view=sql-server-ver16

5. Architecture

5.1. Conceptual Example of Architecture



Link to Figma: https://www.figma.com/file/MseE0ZwKpp5sgZTgiEwPx7/Untitled?type=design&node-id=0%3A1&mode=design&t=mcfuQi1Y9VIoD6JH-1