

# BigQuery e SQL JOINS

BigQuery é um serviço de *data warehouse* e análise de dados oferecido pelo *Google Cloud Platform* (GCP). Ele é um banco de dados totalmente gerenciado e projetado para realizar consultas SQL em grandes conjuntos de dados de maneira rápida e eficiente.

JOIN é uma função de grande importância de SQL utilizada para combinar dados de duas ou mais tabelas baseando-se em uma coluna em comum, e retornando uma nova tabela. Existem 4 tipos de JOIN mais utilizados:

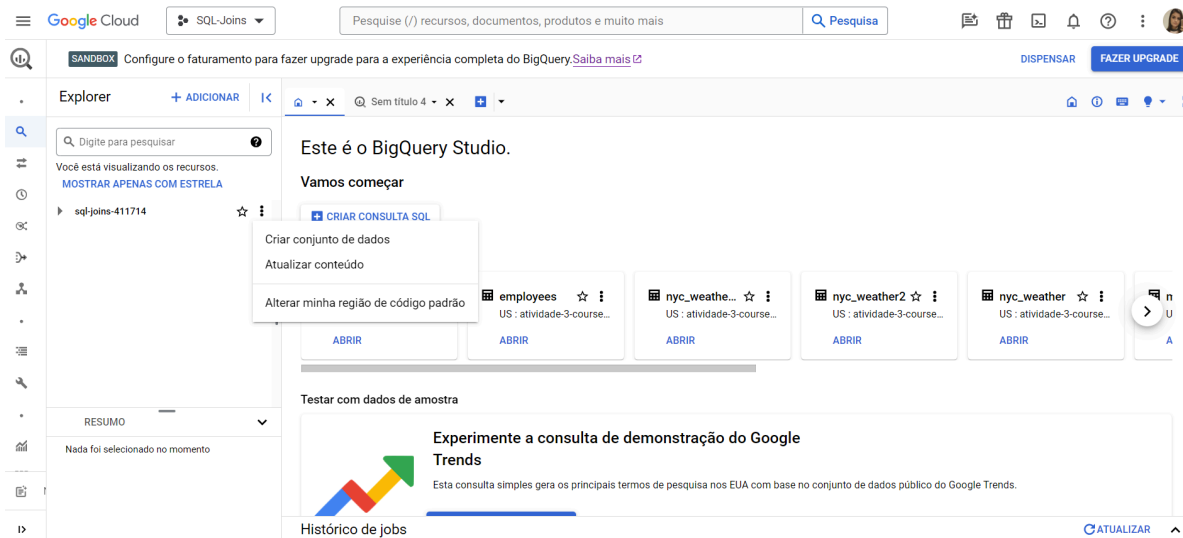
- INNER JOIN: retorna dados com valores correspondentes em ambas as tabelas
- LEFT JOIN: retorna todos os dados da tabela da esquerda (primeira mencionada) e somente os dados correspondentes da tabela da direita (segunda mencionada)
- RIGHT JOIN: retorna todos os dados da tabela da direita (segunda mencionada) e apenas os dados correspondentes da tabela da esquerda (primeira mencionada).
- OUTER JOIN: uma função que combina RIGHT JOIN e LEFT JOIN para retornar todos os registros correspondentes em ambas as tabelas.

A figura abaixo mostra a representação de cada um utilizando diagrama de Venn:



Neste arquivo trago parte de um exercício que utiliza BigQuery para treinar o aprendizado em SQL realizado durante os cursos da certificação *Google Data Analytics* oferecido pela Google em parceria com a plataforma Coursera. A seguir é mostrado o passo a passo do exercício.

1. Após criar um novo projeto no ambiente do BigQuery, criar um novo *dataset*.



O *dataset* foi denominado “*employee\_data*”.

### Criar conjunto de dados

ID do projeto  
sql-joins-411714

MUDAR

Código do conjunto de dados \*

employee\_data

Letras, números e sublinhados são permitidos

Tipo de local ?

☐ Região

Especifique uma região para colocar seus conjuntos de dados com outros serviços do Google Cloud.

☒ Multirregional

Permita que o BigQuery selecione uma região dentro de um grupo para atingir limites de cota mais altos.

Multirregional \*

US (várias regiões nos Estados Unidos)

Expiração da tabela padrão

☐ Ativar expiração da tabela ?

Idade máxima padrão da tabela

Dias

Opções avançadas

^

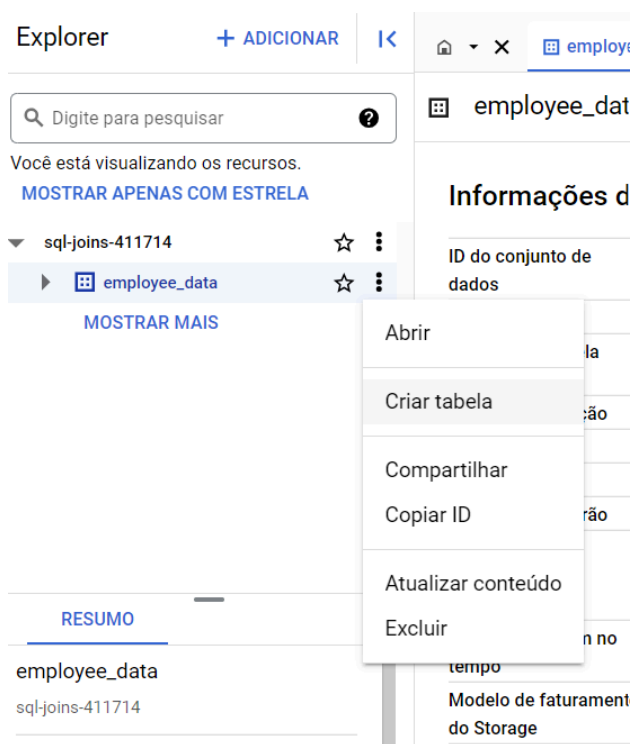
Criptografia ?

☒ Chave de criptografia gerenciada pelo Google

CRIAR CONJUNTO DE DADOS

CANCELAR

1.2 - Em seguida, foi criada a primeira tabela.



Foi chamada de “*employees*”, pois nela seriam carregados dados fictícios sobre colaboradores de uma empresa.



Visualização da tabela *employees*.

Digite para pesquisar

Você está visualizando os recursos.

MOSTRAR APENAS COM ESTRELA

sql-joins-411714

employee\_data

departments

employees

MOSTRAR MAIS

MOSTRAR MAIS

employees

CONSULTA

COMPARTILHAR

COPIAR

ESQUEMA	DETALHES	VISUALIZAÇÃO	LINHAGEM	PERFIL DE D
Linha	name	department_id	role	
1	Dave Smith	1	Product Marketing Manager	
2	Scott Tanner	1	Director of Demand Gen	
3	Margaret Lane	1	VP of Marketing	
4	Julie Jones	2	Software Engineer	
5	Ted Connors	2	Software Engineer	
6	Mary Martin	5	Receptionist	

1.3 - Por fim, a segunda tabela foi criada e denominada “*departments*”, recebendo dados sobre os departamentos da mesma empresa fictícia.

Criar tabela

Fazer upload

Selecionar arquivo \*

\_Departments-Table---Understanding-JOINS.csv

Formato do arquivo

CSV

Destino

Projeto \*

sql-joins-411714

Conjunto de dados \*

employee\_data

Tabela \*

departments

O tamanho máximo do nome é de 1.024 bytes UTF-8. Letras U

Tipo de tabela

Tabela nativa

Esquema

☒ Detectar automaticamente

O esquema será gerado automaticamente.

CRIAR TABELA

CANCELAR

Visualização da tabela *departments*.

Digite para pesquisar

Você está visualizando os recursos.

MOSTRAR APENAS COM ESTRELA

sql-joins-411714

employee\_data

departments

employees

MOSTRAR MAIS

MOSTRAR MAIS

departments

CONSULTA

COMPARTILHAR

ESQUEMA

DETALHES

VISUALIZAÇÃO

LINHAGEM

Linha	name	department_id
1	Marketing	1
2	Engineering	2
3	Accounting	3
4	Sales	4

2 - Após o passo 1, o *dataset* já está organizado e pronto para consultas. Apenas com o dataset *employee\_data* selecionado foi criada uma consulta utilizando INNER JOIN.

employee\_data

\*Sem título 4

Sem título 4

EXECUTAR

SALVAR

FAZER O DOWNLOAD

COMPAR

```
1 SELECT
2   employees.name AS employee_name,
3   employees.role AS employee_role,
4   departments.name AS department_name
5 FROM
6   `sql-joins-411714.employee_data.employees` AS employees
7 INNER JOIN
8   `sql-joins-411714.employee_data.departments` AS departments
9 ON
10  employees.department_id = departments.department_id
```

Resultados da consulta

INFORMAÇÕES DO JOB

RESULTADOS

GRÁFICO

PRÉ-VISUALIZAÇÃO

JSON

DET/

Linha	employee_name	employee_role	department_name
1	Dave Smith	Product Marketing Manager	Marketing
2	Scott Tanner	Director of Demand Gen	Marketing
3	Margaret Lane	VP of Marketing	Marketing
4	Julie Jones	Software Engineer	Engineering
5	Ted Connors	Software Engineer	Engineering

### Explicando o código:

- Apenas as colunas “name” e “role” da tabela *employees* e a coluna “name” da tabela *departments* foram selecionadas. Todas foram renomeadas utilizando o comando AS para facilitar a consulta;
- Na cláusula FROM foi utilizada a tabela *employees* como primeira tabela (da esquerda). Sua localização também foi renomeada com o comando AS, já que seria necessário utilizar novamente a localização posteriormente na cláusula de JOIN;
- Em seguida, foi realizado o INNER JOIN utilizando a localização da tabela *departments*. Especificando que a coluna em comum seria “department\_id” com o comando ON.

### Explicando o resultado:

- Como resultado, tem-se uma tabela contendo apenas os dados referentes aos valores que deram *match* nas colunas em comum.

3. Em seguida, a consulta é modificada para se fazer um novo JOIN.

employee\_data x \*Sem título 4 x

Sem título 4 EXECUTAR SALVAR FAZER O DOWNLOAD COMPARTILHAR

```
1 SELECT
2   employees.name AS employee_name,
3   employees.role AS employee_role,
4   departments.name AS department_name
5 FROM
6   `sql-joins-411714.employee_data.employees` AS employees
7 LEFT JOIN
8   `sql-joins-411714.employee_data.departments` AS departments
9   ON
10  employees.department_id = departments.department_id
```

### Resultados da consulta

INFORMAÇÕES DO JOB RESULTADOS GRÁFICO PRÉ-VISUALIZAÇÃO JSON DETALHES

Linha	employee_name	employee_role	department_name
1	Dave Smith	Product Marketing Manager	Marketing
2	Scott Tanner	Director of Demand Gen	Marketing
3	Margaret Lane	VP of Marketing	Marketing
4	Julie Jones	Software Engineer	Engineering
5	Ted Connors	Software Engineer	Engineering
6	Mary Martin	Receptionist	null

#### Explicando o código:

- Dessa vez foi utilizado um LEFT JOIN, todo o restante da estrutura do código permaneceu o mesmo.

#### Explicando o resultado:

- A tabela resultante contém todos os dados relacionados com os valores que deram *match* nas colunas em comum, além dos valores que não deram *match* da primeira tabela (*employees*).

4. Após isso, foi realizada outra consulta, utilizando RIGHT JOIN.

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'employee\_data' and '\*Sem título 4'. Below the tabs, there are buttons for 'EXECUTAR', 'SALVAR', 'FAZER O DOWNLOAD', and 'COMPARAR'. The query text is as follows:

```
1 SELECT
2   employees.name AS employee_name,
3   employees.role AS employee_role,
4   departments.name AS department_name
5 FROM
6   `sql-joins-411714.employee_data.employees` AS employees
7 RIGHT JOIN
8   `sql-joins-411714.employee_data.departments` AS departments
9 ON
10  employees.department_id = departments.department_id
```

Below the query, the 'Resultados da consulta' section is visible. It has tabs for 'INFORMAÇÕES DO JOB', 'RESULTADOS', 'GRÁFICO', 'PRÉ-VISUALIZAÇÃO', 'JSON', and 'DETA'. The 'RESULTADOS' tab is selected, showing a table with 7 rows and 4 columns: 'employee\_name', 'employee\_role', and 'department\_name'. The first three rows show data for Marketing, and the last four rows show data for Engineering, Accounting, and Sales, with some rows having null values for employee\_name and employee\_role.

Linha	employee_name	employee_role	department_name
1	Dave Smith	Product Marketing Manager	Marketing
2	Scott Tanner	Director of Demand Gen	Marketing
3	Margaret Lane	VP of Marketing	Marketing
4	Julie Jones	Software Engineer	Engineering
5	Ted Connors	Software Engineer	Engineering
6	null	null	Accounting
7	null	null	Sales

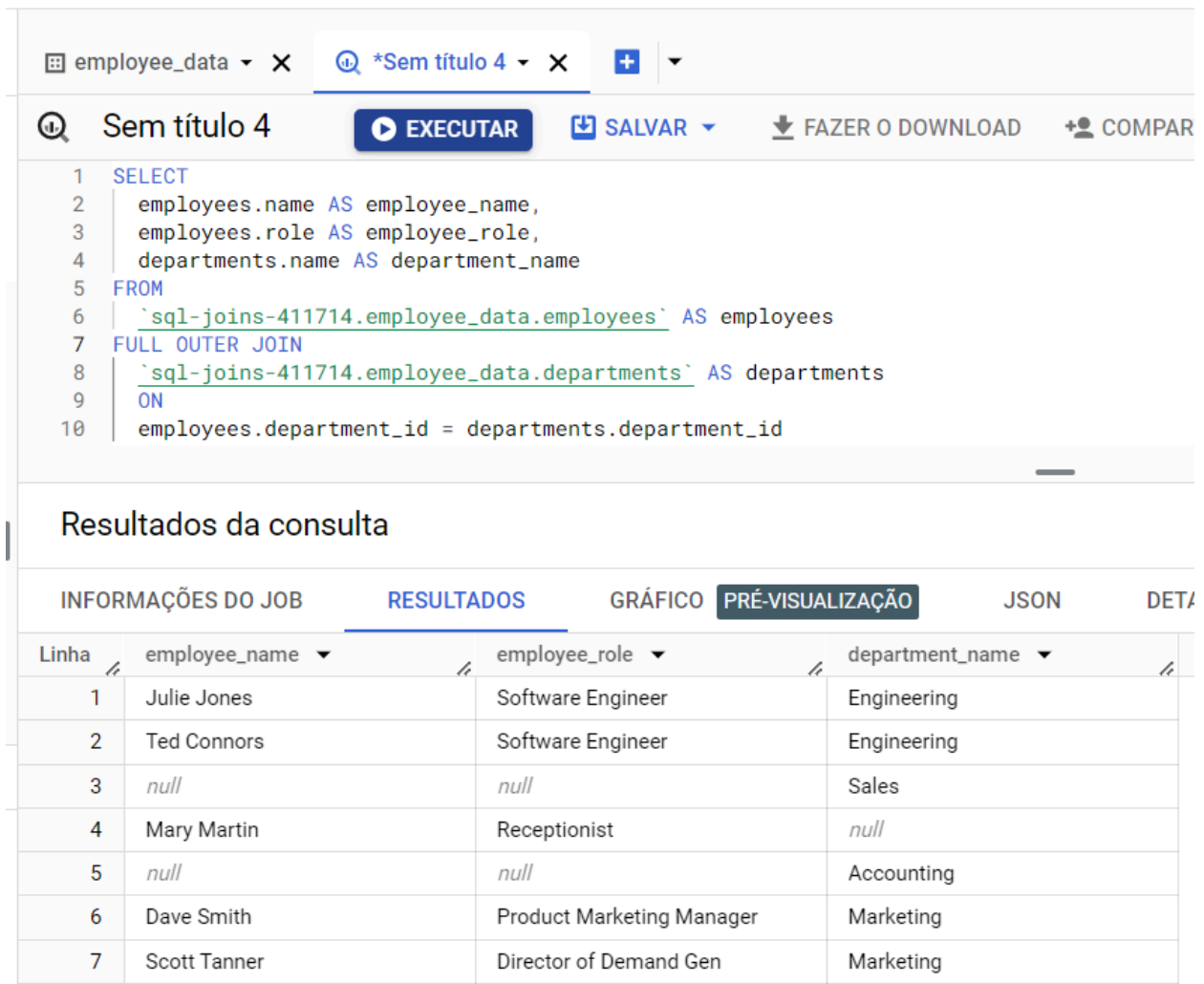
#### Explicando o código:

- Novamente, a estrutura inicial do código se manteve, porém foi usando um RIGHT JOIN.

#### Explicando o resultado:

- A tabela resultante apresenta todos os dados relacionados com os valores que deram *match* nas colunas em comum, além dos valores que não deram *match* da segunda tabela (*departments*).

5. Por fim, a consulta foi modificada novamente para utilizar um outro JOIN.



The screenshot shows a SQL query editor interface. At the top, there's a tab labeled "Sem título 4" with a search icon and a "EXECUTAR" button. Below the tab, the query is displayed in a monospace font. The query is a SELECT statement that joins the 'employees' and 'departments' tables from a database named 'sql-joins-411714.employee\_data' using a FULL OUTER JOIN on the 'department\_id' column. Below the query, there's a section titled "Resultados da consulta" with a tabbed interface. The "RESULTADOS" tab is active, showing a table with 7 rows and 4 columns: 'Linha', 'employee\_name', 'employee\_role', and 'department\_name'. The data is as follows:

Linha	employee_name	employee_role	department_name
1	Julie Jones	Software Engineer	Engineering
2	Ted Connors	Software Engineer	Engineering
3	null	null	Sales
4	Mary Martin	Receptionist	null
5	null	null	Accounting
6	Dave Smith	Product Marketing Manager	Marketing
7	Scott Tanner	Director of Demand Gen	Marketing

#### Explicando o código:

- A estrutura inicial do código foi mantida, e um RIGHT JOIN foi utilizado.

#### Explicando o resultado:

- A tabela resultante tem todos os dados relacionados com os valores que deram *match* nas colunas em comum, além dos valores que não deram *match* de ambas as tabelas.