**2. Joining Data in SQL**

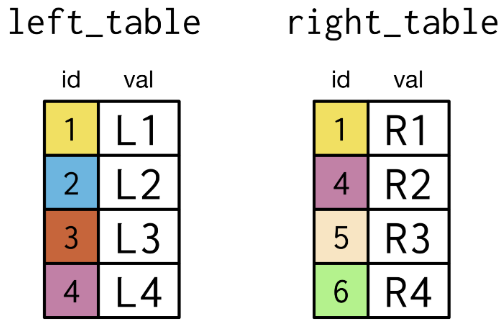
Juntar duas ou mais tabelas em uma única tabela.

INNER JOIN e LEFT JOIN são as junções mais comuns.

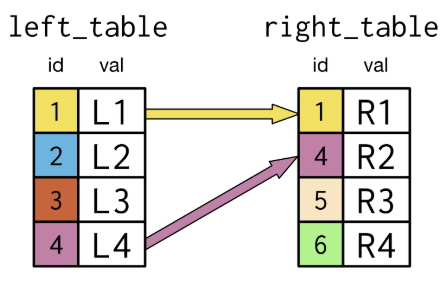
Vamos começar com o layout de um dado e como o INNER JOIN pode ser aplicado a esse dado.

Nos próximos exercícios trabalharemos com duas tabelas nomeadas de left e right. Veja que os valores de mesmo id estão coloridos com a mesma cor.

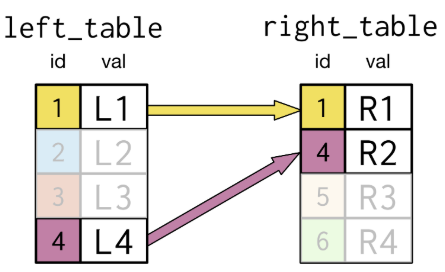
O campo “id” é conhecido como um campo-chave já q ele pode ser usado p/ se referir a um tabela para outra tabela. Ambas as tabelas left e right possuem um campo chamado “val”, que será útil p/ ver quais registros e valores são incluídos em cada join.



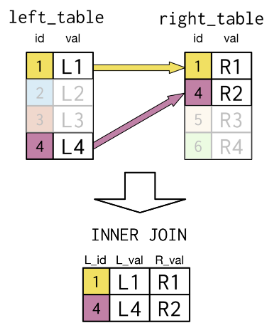
O INNER JOIN apenas inclui registros nos quais a chave está em ambas as tabelas. Veja que aqui o campo id combina apenas os valores 1 e 4. Com INNER JOIN nós procuramos por combinações na tabela right correspondendo a todas as entradas no campo-chave da tabela left.



O foco aqui muda para apenas os registros com uma correspondência em termos do campo “id”. Os regsitros que não são do interesse do INNER JOIN foram desbotados.



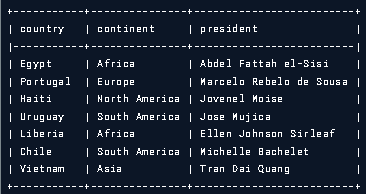
Está é a tabela única resultante da cláusula INNER JOIN que fornece o campo “val” da tabela right com registros correspondentes a apenas aqueles com o valor de id 1 e 4.



Agora que temos noção de como o INNER JOIN funciona, vamos tentar um exemplo em SQL. A tabela prime\_ministers é uma das tabelas da base de dados dos líderes. Note que os países estão inclusos. Suponha que vc esteja interessado em determinar nações que possuem ambos um primeiro ministro e presidente e colocar os resultados em uma tabela só.

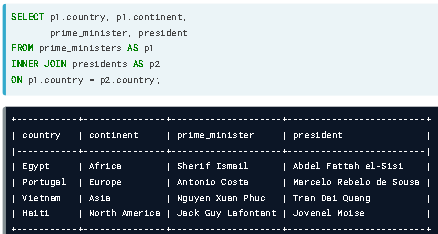


Agora vamos ver a tabela dos presidents. Há países que aparecem em ambas as tabelas. Com tabelas curtas como estas é fácil de ver que Egito, Portugal, Vietnan e Haiti aparecem em ambas as tabelas. Para tabelas grandes, não dá para fazer isso visualmente.

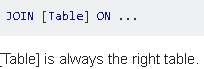


Então usamos a sintaxe do SQL para pegar os países que possuem prime ministers e presidentes dessas duas tabelas para uma.

A sintaxe para completar um INNER JOIN d tabela prime\_ministers para a tabela presidents baseado no campo-chave country é mostrada abaixo. A declaração SELECT é usada para selecionar campos específicos das duas tabelas. Nesse caso, já que countries existe nas duas tabelas, devemos escrever o nome da tabela “ponto” primeiro e depois o nome da coluna (p1.country) para evitar erros. Em seguida listamos a tabela a esquerda do INNER JOIN depois do FROM. E depois listamos a tabela direita depois do INNER JOIN. Por fim, especificamos as chaves das duas tabelas que gostaríamos de combinar.

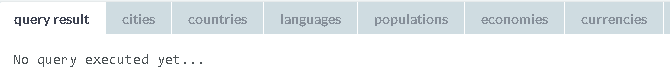


P.S.: Left Table é a primeira tabela no SELECT. Right Table sempre vai ser a tabela que vc está juntando.

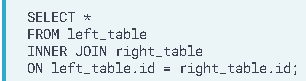


**INNER JOIN**

Vamos usar o PostgreSQL, mas esses JOINs podem ser aplicados para adiferentes formas de SQL. Ao longo do curso vamos usar as bases de dados countries, que contém informações das cidades mais populosas do mundo. Estas são as tabelas que usaremos.

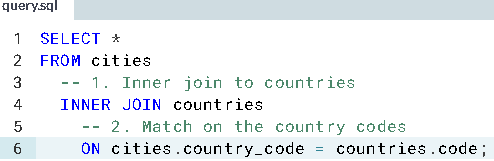


Sintaxe do INNER JOIN:

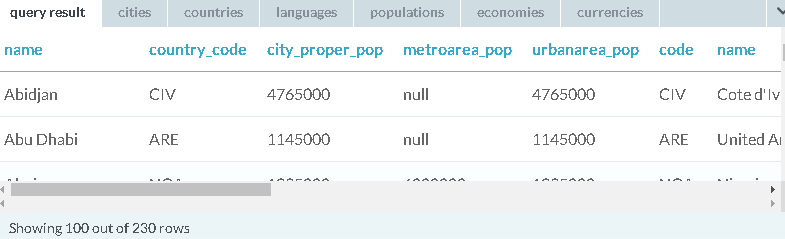


Você começará com uma instrução SELECT e criará uma inner join (junção interna) com as tabelas de cidades e países:

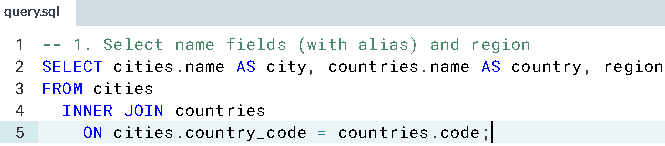
Passo 1: INNER JOIN (associe) a tabela de cities à esquerda a tabela de countries à direita, mantendo todos os campos nas duas tabelas. Você deve corresponder as tabelas no campo country\_code nas cidades e o campo code nos países. Não aliase suas tabelas aqui ou na próxima etapa. Usar cidades e países está bom por enquanto.

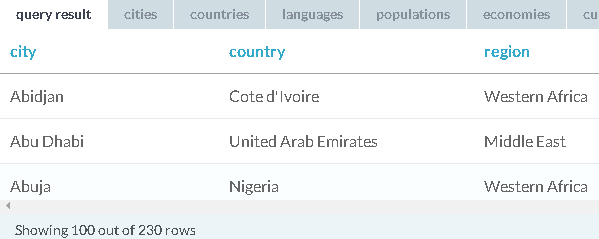


O resultado é a junção da tabela countries a tabela cities.



Passo 2: Modifique a instrução SELECT para manter apenas o nome da cidade, o nome do país e o nome da região em que o país reside. Lembre-se do nosso curso Introdução ao SQL for Data Science que você pode criar alias de campos usando o AS. Aliase o nome da cidade como cidade e o nome do país como país.

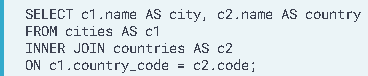




Observe que ao selecionar a coluna region não precisei falar de qual tabela, pois só há essa coluna em uma das tabelas, então o SQL não se confundirá.

**INNER JOIN (2)**

Em vez de escrever o nome completo da tabela, você pode usar o alias da tabela como atalho. Para tabelas, você também usa o AS para adicionar o alias imediatamente após o nome da tabela com um espaço. Confira o alias de cidades e países abaixo.

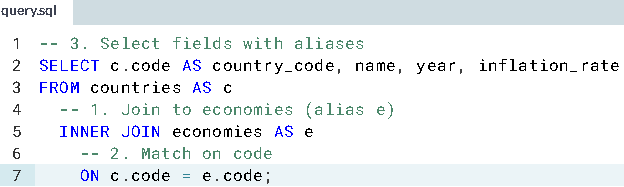


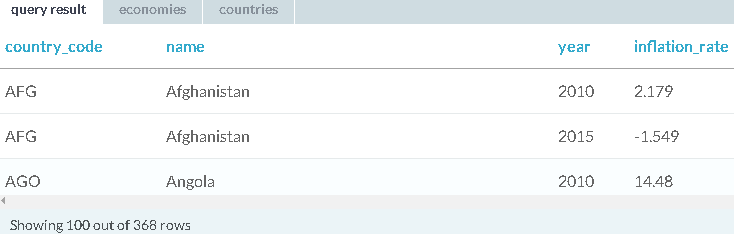
Observe que para selecionar um campo em sua consulta que apareça em várias tabelas, você precisará identificar a que tabela / alias de tabela se refere usando um “.” na sua instrução SELECT.

Agora, você explorará uma maneira de obter dados das tabelas countries e economies para examinar a taxa de inflação em 2010 e 2015.

P.S.: Às vezes, é mais fácil escrever código SQL fora de ordem: você escreve a instrução SELECT depois de fazer JOIN.

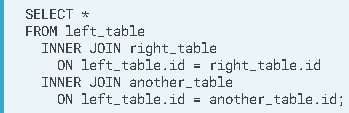
Ex. Junte-se às tabelas countries (esquerda) e economies (direita) aliasing countries AS c e economies AS e. Especifique o campo para corresponder às tabelas ON. Nesta junção, SELECT: c.code, aliased como country\_code. E name, year e inflation\_rate, sem alias:





**INNER JOIN (3)**

A capacidade de combinar várias junções (JOIN) em uma única consulta é um recurso poderoso do SQL, por exemplo:



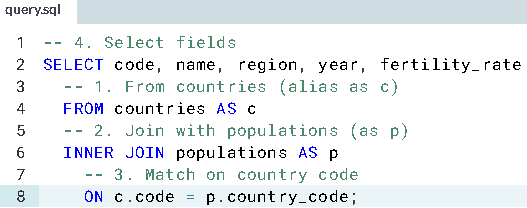
Como você pode ver aqui, torna-se entediante escrever continuamente nomes longos de tabela em junções. Por isso é útil criar um alias para cada tabela usando a primeira letra do nome (por exemplo, countries AS c)! É uma prática padrão usar o alias dessa maneira e, se você optar pelo alias de tabelas ou for solicitado especificamente a um exercício neste curso, você deve seguir este protocolo.

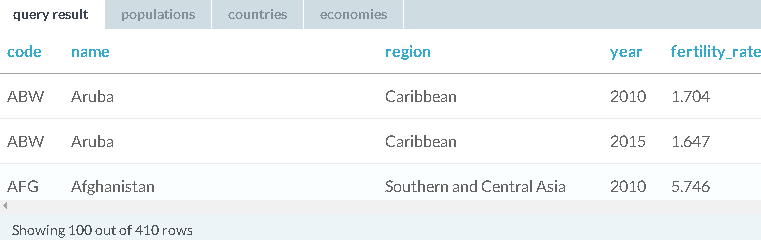
Agora, para cada país, você deseja obter o nome do país, sua região e a taxa de fertilidade e taxa de desemprego de 2010 e 2015.

Observe que os resultados devem funcionar ao longo deste curso com ou sem alias de tabela, a menos que especificado de maneira diferente.

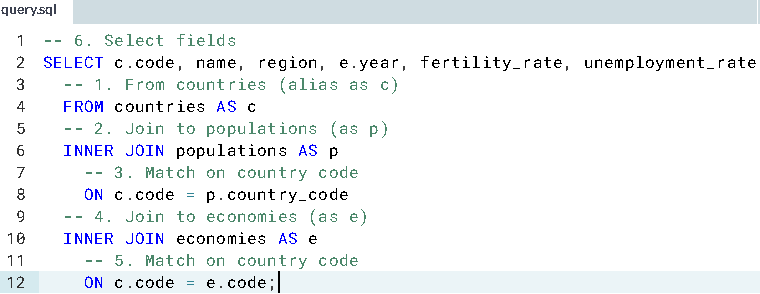
Ex.

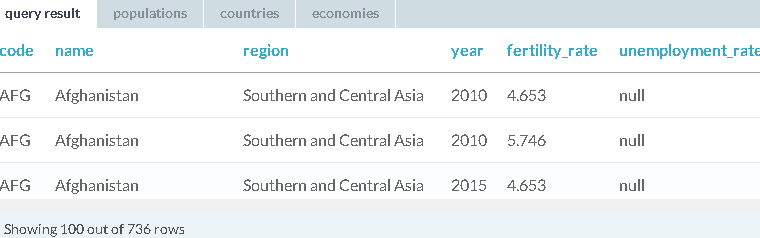
Passo 1: INNER JOIN countries (esquerda) e populations (direita) nos campos de code e country\_ code, respectivamente. - Alias countries AS c e population AS p. - Selecione o code, name e region de countries e também selecione o year e o fertility\_rate de populations (5 campos no total).





Passo 2: Adicione um INNER JOIN adicional com economies à sua consulta anterior juntando no código. - Inclua a coluna unemployment\_rate que ficou disponível por meio da junção com economies. - Observe que year aparece em populations e economies; portanto, você deve usar explicitamente “e.year” em vez de ano.

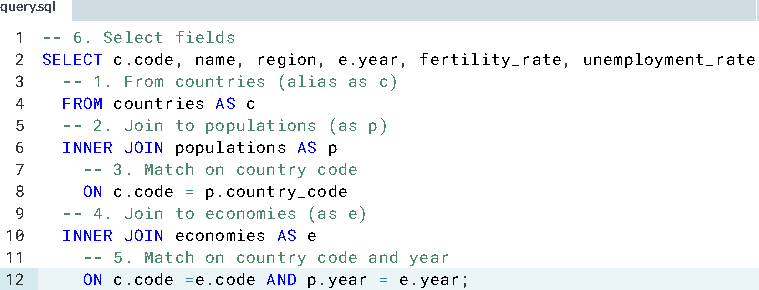


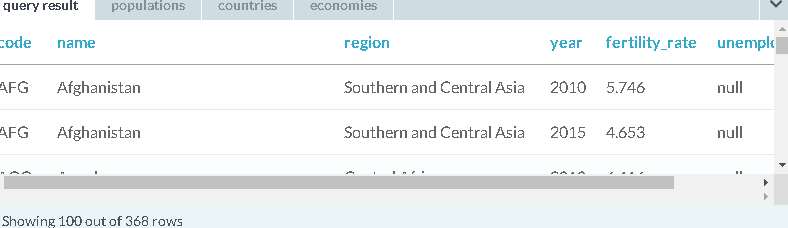


**Passo 3**: Role para baixo o resultado da consulta e dê uma olhada nos resultados da Albânia da sua consulta anterior. Algo parece estranho para você?

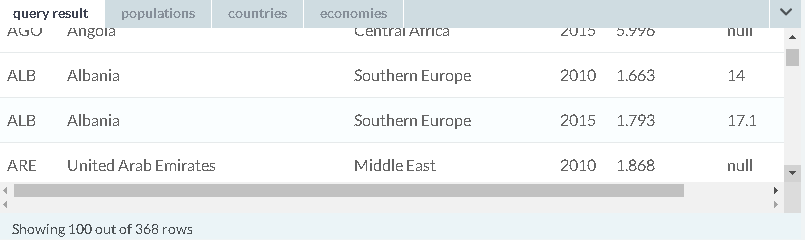
O problema de fazer sua última associação em c.code = e.code e não incluir também o ano é que, por exemplo, o valor de 2010 para fertility\_rate também é emparelhado com o valor de 2015 para unemployment\_rate.

Corrija sua consulta anterior: na sua última cláusula ON, use AND para adicionar uma condição de associação. Além de ingressar no código em c e e, também junte no year em e e p.



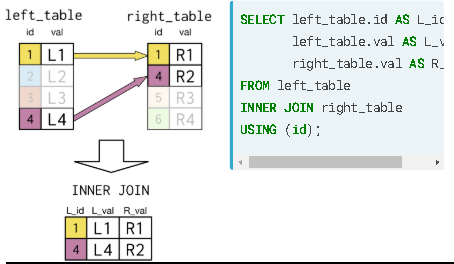


Veja que agora Albania tem duas linhas, antes tinhamos 4:

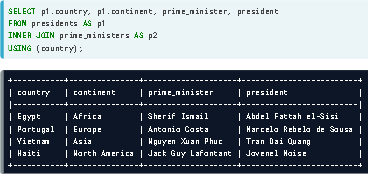


**INNER JOIN via USING**

Quando o campo-chave que você quer juntar tem o mesmo nome nas duas tabelas podemos usar a cláusula USING ao invés de ON. O campo “id” tem o mesmo nome em ambas as tabelas da esquerda e da direita, assim podemos especificar USING ao invés de ON. O parênteses é necessário ao redor do nome do campo

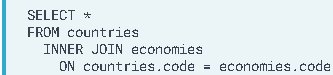


Vamos relembrar a tabela dos presidents e prime\_ministers para determinar quais países tem os dois líderes.

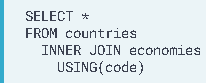


**REVIEW INNER JOIN USING ON**

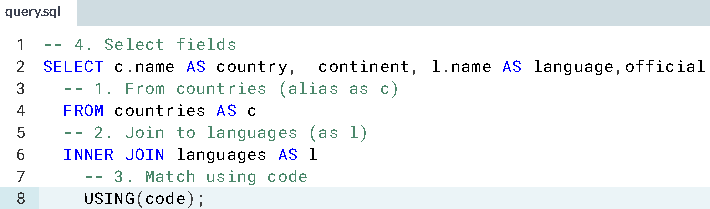
Ao unir tabelas com um nome de campo comum, por ex.:

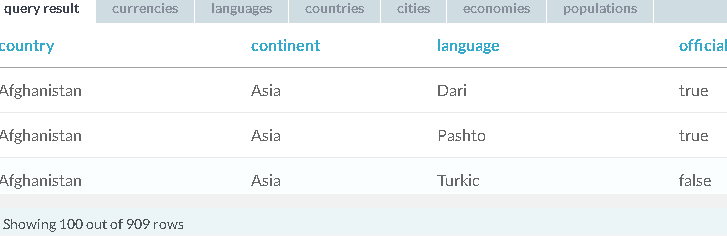


Podemos usar USING como atalho:



Ex. - INNER JOIN (associe) countries à esquerda e os languages à direita com USING (code). - Selecione os campos correspondentes a: nome do país AS country, nome do continente, nome do idioma AS language, e se o idioma é oficial ou não. Lembre-se de alias suas tabelas usando a primeira letra de seus nomes.

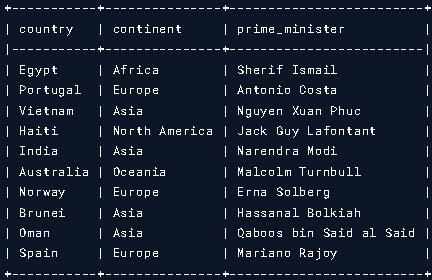




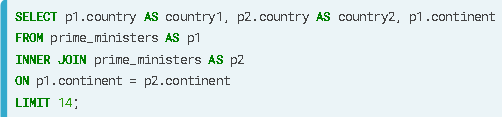
**Self Joins – (Self-ish joins) , just in CASE**

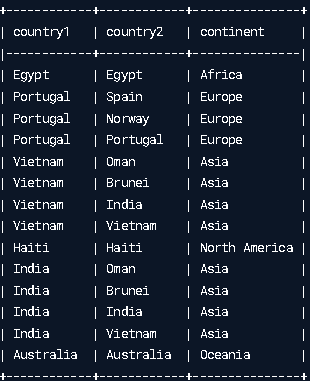
Vamos nos aprofundar em INNER JOINs em que a tabela é juntada com ela mesma (egoísta). Esse tipo de JOINs são chamos de Self Joins. Vamos aprender como fatiar um campo numérico em categorias usando o comando CASE. Juntar uma tabela nela mesma parece meio estranho.

As auto-junções são usadas para comparar valores em um campo com outros valores do mesmo campo de dentro da mesma tabela. Ex. da tabela prime\_minister:



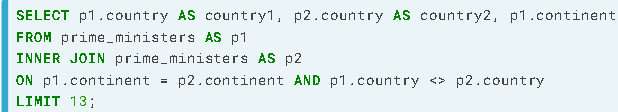
Se você quiser criar uma nova tabela mostrando countries que estão no mesmo continente combinados como pares? Veja que a coluna country e continent é selecionada duas vezes . A tabela prime\_minister está à esquerda e à direita. O passo vital aqui é estabelecer as colunas-chave pela qual nós combinaremos a tabela a ela msma. Para cada país teremos uma combinação se o país da tabela direita (q tbm é prime\_ministers) for no mesmo continente. Por fim, já que os resultados desta consulta são maiores do que pode caber no slide da aula, veremos apenas os primeiros 14 registros.





Veja que temos exatamente isso no resultado. É um emparelhamento de cada país com cada país no mesmo continente. Mas há um problema aqui. Nós não queremos listar os países com eles mesmos. Nós não queremos incluir linhas onde os países são os mesmos nos campos contry1 e country2.

A cláusula AND pode checar multiplas condições. Aqui uma combinação não será feita entre prime\_ministers e ela mesma se os países combinarem.

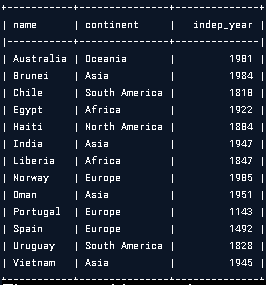


Agora teremos a tabela correta. Os resultados foram limitados para caber no slide da aula. Observe que o self-join não tem uma sintaxe tão simples quanto INNER JOIN

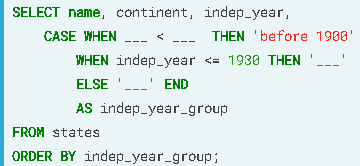


O próximo comando não é JOIN, mas é uma ferramenta útil no nosso repertório: CASE WHEN and THEN.

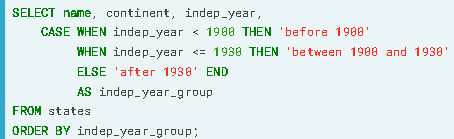
Vamos usar CASE com outra tabela da base de dados do líderes. A tabela States contém dados numéricos sobre diferentes países nos 6 continentes. Suponha que eu queira agrupar o ano da independencia em categorias de antes de 1900, entre 1900 e 1930 e depois de 1930. Vamos usar CASE, que é uma forma de fazer múltiplas declarações if-then-else (se-se não- então) de forma simplificada no SQL.

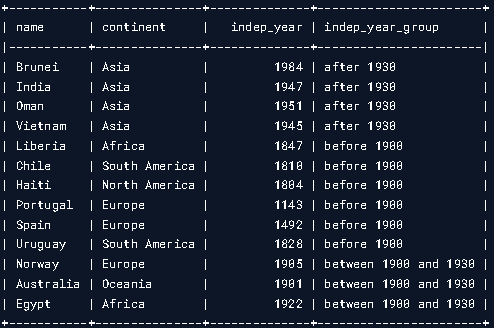


Veja o layout básico para criar um novo campo contendo os agrupamentos.



Para preenchê-lo, depois do 1º WHEN devemos especificar q queremos checar indep\_year sendo menor que 1900. Em seguida (no 2º branco), queremos que o indep\_year contenha “entre 1900 e 1930”. Por fim registros que não combinem com essas condições serão atribuidos ao valor “after 1930” no indep\_year\_group. Veja aqui a query preenchida:





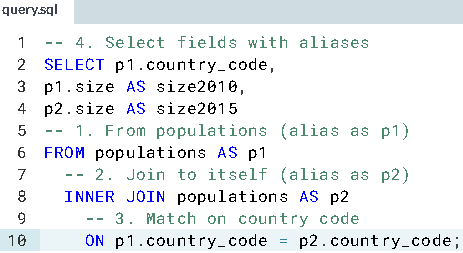
Note tbm como os continentes se relacioname com as datas de independencia.

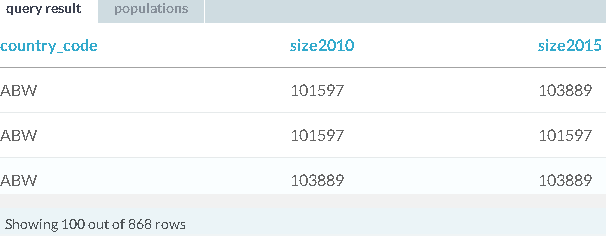
**Ex. Self-join: Self-join:**

In this exercise, you'll use the populations table to perform a self-join to calculate the percentage increase in population from 2010 to 2015 for each country code!

Since you'll be joining the populations table to itself, you can alias populations as p1 and also populations as p2. This is good practice whenever you are aliasing and your tables have the same first letter. Note that you are required to alias the tables with self-joins.

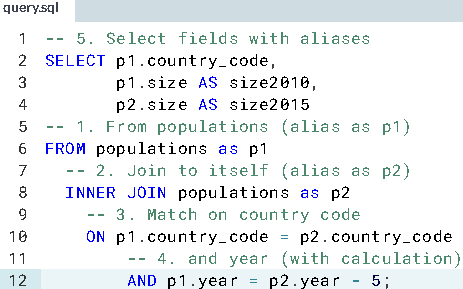
Passo 1 -Join populations with itself ON country\_code. - Select the country\_code from p1 and the size field from both p1 and p2. SQL won't allow same-named fields, so alias p1.size as size2010 and p2.size as size2015.





Passo 2: Notice from the result that for each country\_code you have four entries laying out all combinations of 2010 and 2015.

Extend the ON in your query to include only those records where the p1.year (2010) matches with p2.year - 5 (2015 - 5 = 2010). This will omit the three entries per country\_code that you aren't interested in.

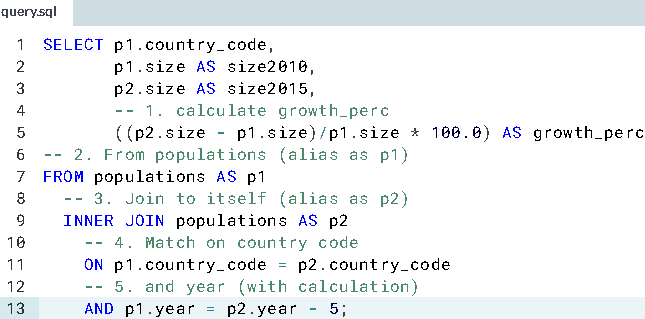


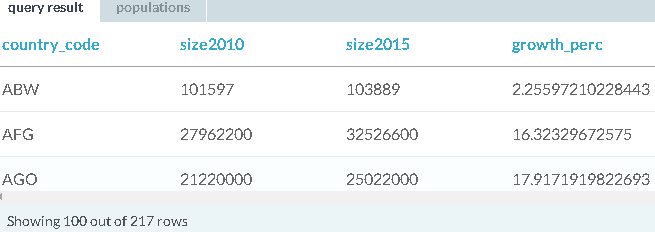


**Passo 3:** As you just saw, you can also use SQL to calculate values like p2.year - 5 for you. With two fields like size2010 and size2015, you may want to determine the percentage increase from one field to the next:

With two numeric fields A and B, the percentage growth from A to B can be calculated as (B−A)/A∗100.0.

Add a new field to SELECT, aliased as growth\_perc, that calculates the percentage population growth from 2010 to 2015 for each country, using p2.size and p1.size.





**Ex. Case when and then**

Freqüentemente, é útil olhar para um campo numérico não como dados brutos, mas como estando em diferentes categorias ou grupos.

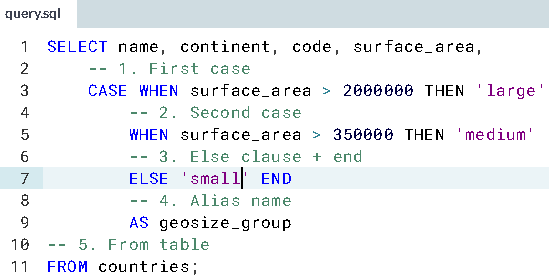
Você pode usar CASE com WHEN, THEN, ELSE e END para definir um novo campo de agrupamento.

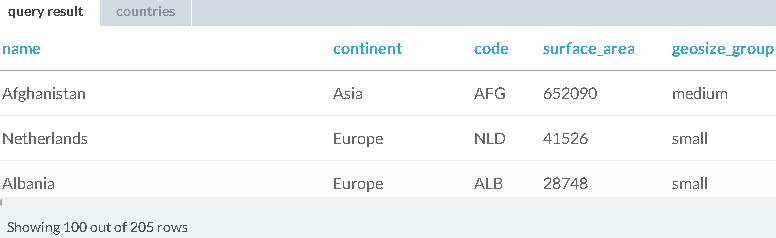
Ex. Using the countries table, create a new field AS geosize\_group that groups the countries into three groups:

- If surface\_area is greater than 2 million, geosize\_group is 'large'.

- If surface\_area is greater than 350 thousand but not larger than 2 million, geosize\_group is 'medium'.

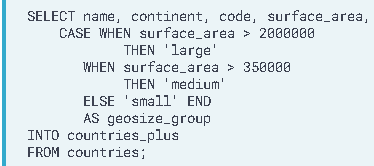
- Otherwise, geosize\_group is 'small'.

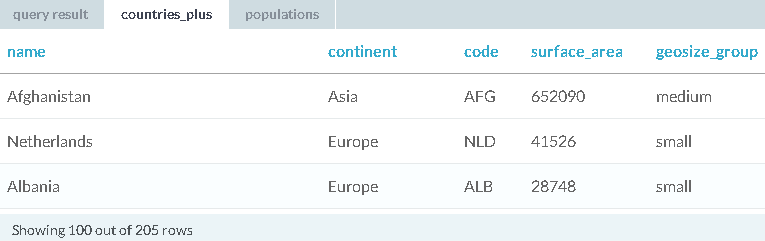




**Ex. Desafio:**

The table you created with the added geosize\_group field has been loaded for you here with the name countries\_plus. Observe the use of (and the placement of) the INTO command to create this countries\_plus table [a consulta que fiz no exercício anterior foi salva como tabela usando o INTO]:





Agora você explorará a relação entre o tamanho de um país em termos de área de superfície e em termos de população usando campos de agrupamento criados com o CASE.

Ao final deste exercício, você escreverá duas consultas consecutivas em um único script.

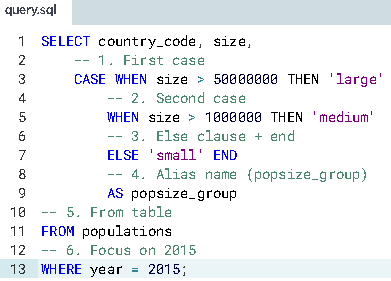
**Passo 1: Using the populations table focused only for the year 2015, create a new field aliased as popsize\_group to organize population size into**

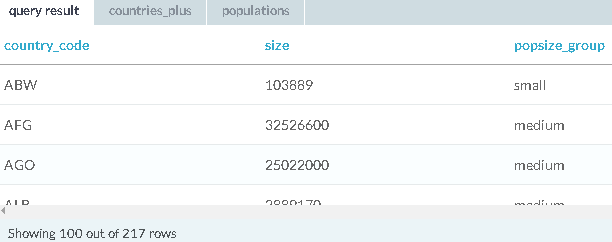
**- 'large' (> 50 million),**

**- 'medium' (> 1 million), and**

**- 'small' groups.**

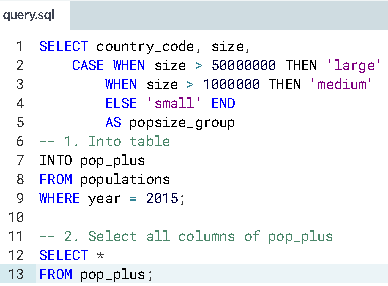
**Select only the country code, population size, and this new popsize\_group as fields.**

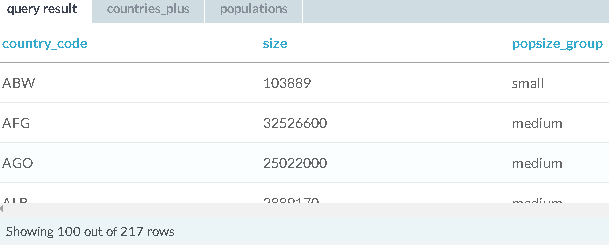




**Passo 2: Use INTO to save the result of the previous query as pop\_plus. You can see an example of this in the countries\_plus code in the assignment text. Make sure to include a ; at the end of your WHERE clause!**

**Then, include another query below your first query to display all the records in pop\_plus using SELECT \* FROM pop\_plus; so that you generate results and this will display pop\_plus in query result.**



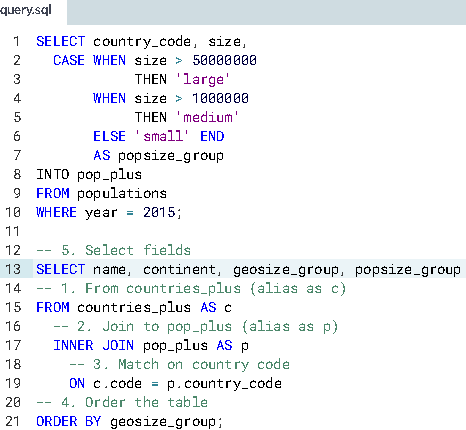


**Passo 3: - Keep the first query intact that creates pop\_plus using INTO.**

**- Write a query to join countries\_plus AS c on the left with pop\_plus AS p on the right matching on the country code fields.**

**- Sort the data based on geosize\_group, in ascending order so that large appears on top.**

**- Select the name, continent, geosize\_group, and popsize\_group fields.**



**LEFT and RIGHT JOINs (ou outer joins – full, right, left joins)**

As junções externas são como estender a mão para outra tabela enquanto mantém todos os registros da tabela original. Os INNER JOINs (junções internas) mantêm apenas os registros nas duas tabelas.

P.S.: **OUTER** vai te dar todas as linhas da tabela A e todas as linhas da tabela B, mesmo que não haja correspondência.

**INNER** vai te dar apenas as tuplas de ambas as tabelas que tem correspondência da chave que escolher para sua junção.

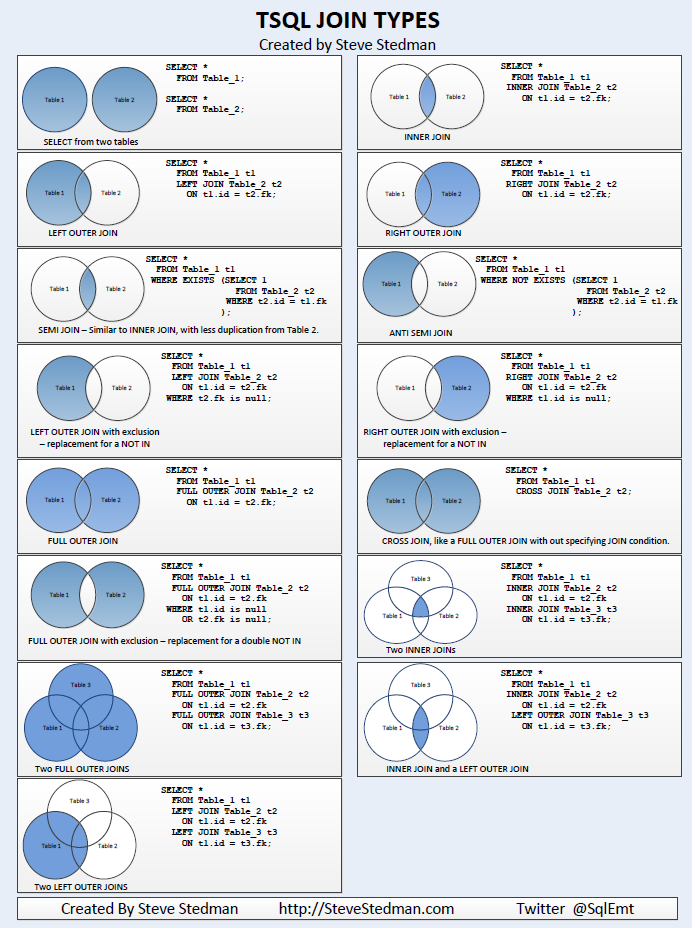
**Ex:**

a tabela A tem como chave da junção os valores: 1 | 2 | 3

E a tabela B: 2|3|4

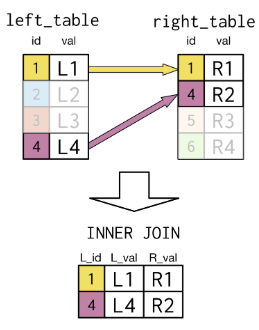
Um OUTER JOIN por essa chave entre essas tabelas resultaria em: 1|2|3|4

E um INNER JOIN resultaria em apenas: 2 | 3

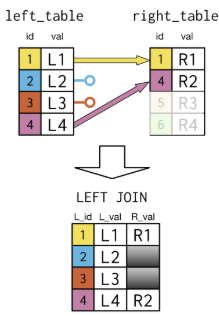


Há 3 tipos de Outer joins: right, left, full.

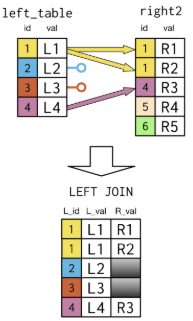
Diferença do INNER JOIN e LEFT JOIN: no INNER JOIN apenas os registros do campo id que se encontram em ambas as tabelas aparecem nos reusltados.



Já no LEFT JOIN, veja que há registros da tabela esquerda que não possuem valores correspondentes no campo da tabela direita (os campos de id 2 e 3 da tabela esquerda não tem correspondente na tabela direira). O resultado de um query usando LEFT JOIN é que ele mantém todos os registros originais da tabela esquerda, mas marca os valores que faltaram na tabela direita, pois não tem nenhum correspondente desse resgirto da tabela esquerda na tabela direita. Veja que os valores de id 5 e 6 da tabela direita não vão para os resultados do LEFT JOIN.

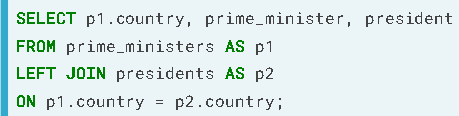


Não é sempre que cada valor-chave da tabela esquerda corresponde exatamente a um registro na coluna-chave da tabela direita.Veja esse ex.:



Há entradas faltando nos ids 2 e 3. Linhas duplicadas são mostradas no LEFT JOIN para o id 1, já q ele tem duas correspondências na tabela direita.

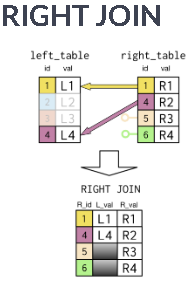
A sintaxe do LEFT JOIN é similar ao INNER JOIN. Vamos explorar o mesmo código que usamos antes para determinar os countries com prime\_minister e president. Mas desta vez vamos usar LEFT JOIN ao invés de INNER JOIN e remover continent para economizar espaço na tela.



Os primeiros 4 registros nesta tabela são os mesmos do INNER JOIN. Os últimos 6 correspondem aos países que não tem um presidente, por isso seus valores estão faltando.

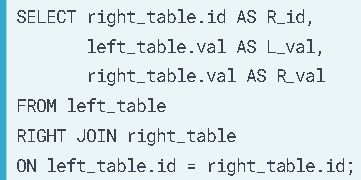


RIGHT JOIN é menos comum do que o LEFT JOIN, então vamos ver sobre ele brevemente.



Vamos ver como ele funciona. Ao invés de combinar entradas da coluna id da tabela esquerda para a tabela direita, o RIGHT JOIN faz o inverso. Vejo que na tabela direita os ids 5 e 6 não encontram correspondente na tabela esquerda. A tabela resultante do RIGHT JOIN mostra as entradas ausentes no campo L\_val.

Veja que a tabela direita aparece depois do RIGHT JOIN e tabela esquerda aparece depois do FROM.

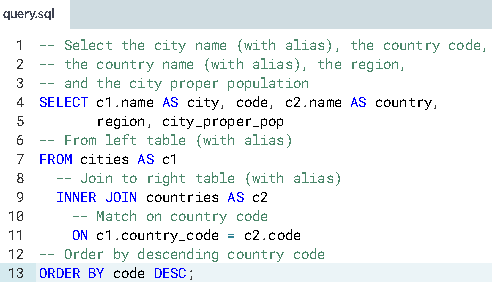


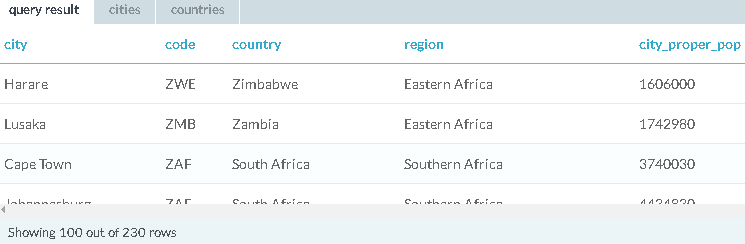
**EX LEFT JOIN:**

Vamos ver as diferenças entre usar INNER JOIN e LEFT JOIN usando as tabelas cities e countries.

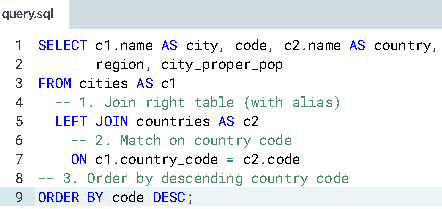
You'll begin by performing an inner join with the cities table on the left and the countries table on the right. Remember to alias the name of the city field as city and the name of the country field as country.

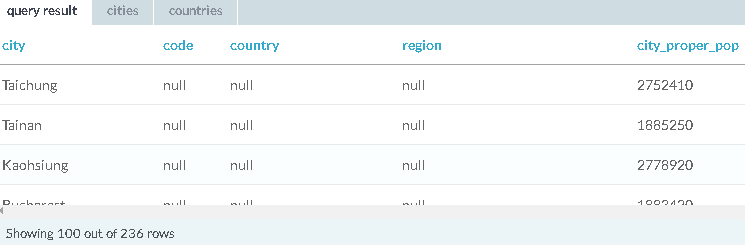
**1. Fill in the code based on the instructions in the code comments to complete the inner join. Note how many records are in the result of the join in the query result tab.**





**2.** **Change the code to perform a LEFT JOIN instead of an INNER JOIN. After executing this query, note how many records the query result contains.**





**EX. LEFT JOIN (2)**

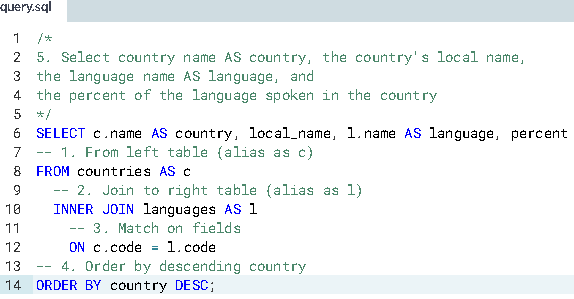
**Outro exemplo comparando INNER JOIN ao LEFT JOIN. Veja que a tabela languages possui 955 linhas e a tabela countries 205 linhas.**

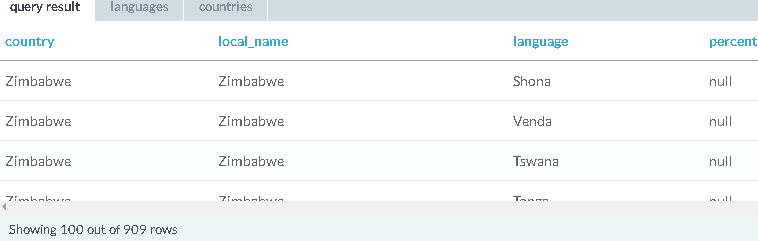
**You will begin with an inner join on the countries table on the left with the languages table on the right. Then you'll change the code to a left join in the next bullet.**

**Note the use of multi-line comments here using /\* and \*/.**

**1.** **Perform an inner join. Alias the name of the country field as country and the name of the language field as language.**

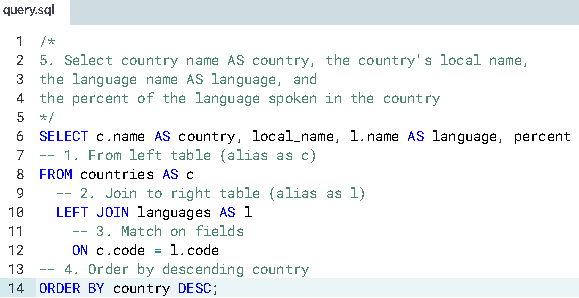
**Sort based on descending country name.**

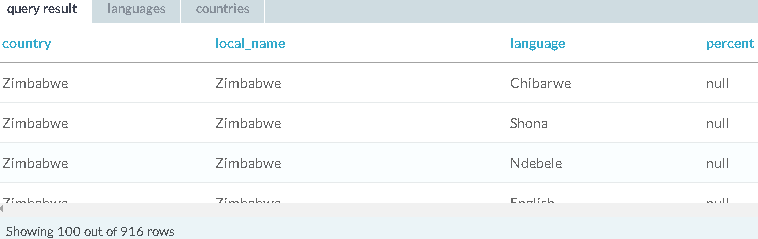




**2.** **Perform a left join instead of an inner join. Observe the result, and also note the change in the number of records in the result.**

**Carefully review which records appear in the left join result, but not in the inner join result.**

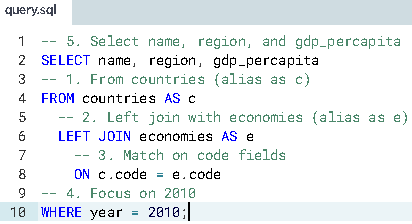


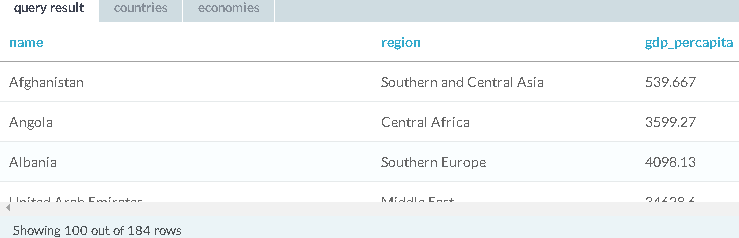


**Ex. LEFT JOIN (3)**

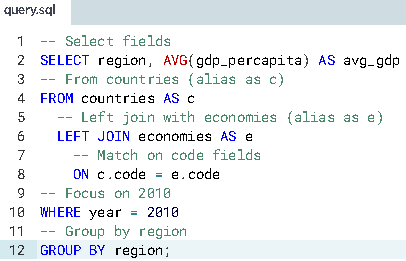
**You'll now revisit the use of the AVG() function introduced in our Intro to SQL for Data Science course. You will use it in combination with left join to determine the average gross domestic product (GDP) per capita by region in 2010.**

**Passo 1: Begin with a left join with the countries table on the left and the economies table on the right. Focus only on records with 2010 as the year.**



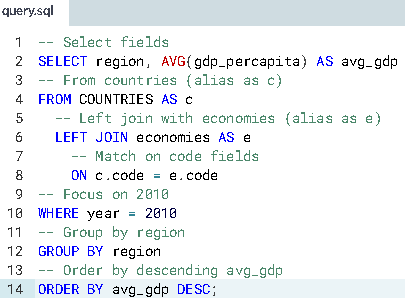


**Passo 2: Modify your code to calculate the average GDP per capita AS avg\_gdp for each region in 2010. Select the region and avg\_gdp fields.**





**Passo 3: Arrange this data on average GDP per capita for each region in 2010 from highest to lowest average GDP per capita.**

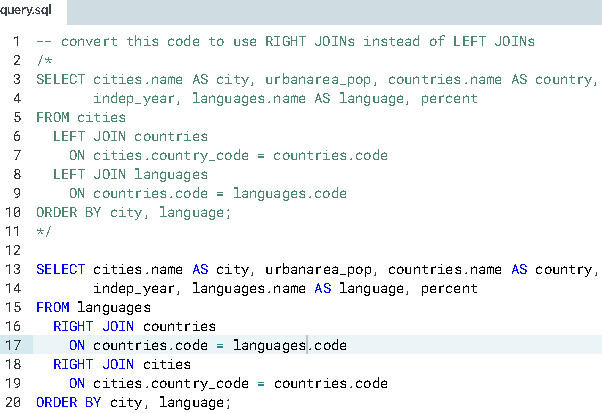


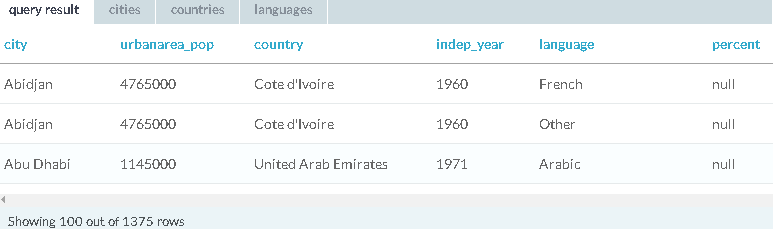


**Ex. RIGHT JOIN**

Right joins aren't as common as left joins. One reason why is that you can always write a right join as a left join.

**Ex. The left join code is commented out here. Your task is to write a new query using rights joins that produces the same result as what the query using left joins produces. Keep this left joins code commented as you write your own query just below it using right joins to solve the problem. Note the order of the joins matters in your conversion to using right joins!**



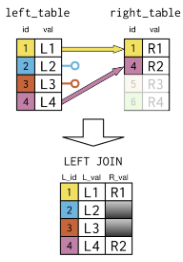
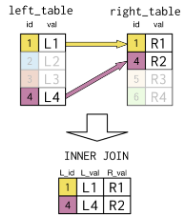


**FULL JOIN**

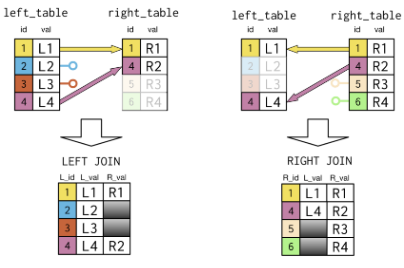
O último dos 3 tipos de OUTER JOINs é o FULL JOIN.

Vamos ver a diferença entre o FULL JOIN e os outros JOINs que aprendemos até agora (INNER JOIN, LEFT JOIN, RIGHT JOIN).

Lembrando que o INNER JOIN só mantém os registros que possuem um campo-chave com valores correspondentes nas duas tabelas. O LEFT JOIN mantém todos os regsistros da tabela esquerda e traz os valores ausentes daqueles cujos valores dos campos-chaves não aparecem da tabela direita.

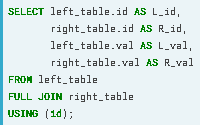
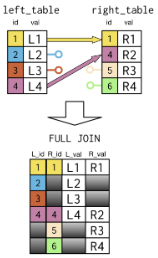


Lembrando as diferenças do RIGHT JOIN e do LEFT JOIN. Os valores id de 2 e 3 da tabela esquerda não correspondem aos valores id da tabela direita. Assim, valores ausentes são trazidos no LEFT JOIN. Do mesmo modo para RIGHT JOIN, valores ausentes são trazidos para os valores de id 5 e 6.



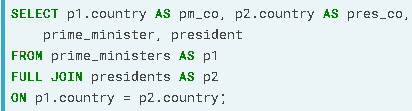
O FULL JOIN combina o LEFT JOIN e o RIGHT JOIN, veja, ele vai trazer todos os registros de ambas as tabelas esquerda e direita e rastrear os valores ausentes. Note os valores ausentes e veja que todos os 6 valores id foram incluídos na tabela.

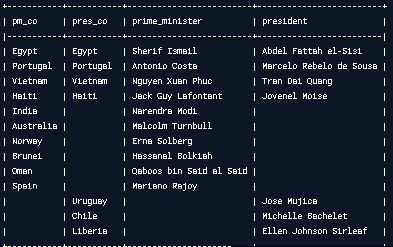
Veja também o código SQL para produzir o resultado deste FULL JOIN



Ex. da base de dados dos líderes. Os países com prime ministers e/ou presidents.

O SELECT inclui o campo COUNTRY de ambas as tabelas e os campos prime\_minister e president. O FROM especifica a tabela esquerda como prime\_ministers. A ordem importa aqui, se você mudar as duas tabelas o resultado será um pouco diferente. O FULL JOIN especifica a tabela direita. Por fim o ON diz que o JOIN é baseado no campo-chave country nas duas tabelas.





**FULL JOIN EX.**

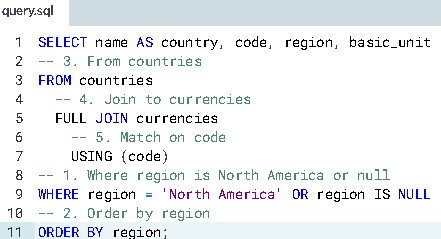
In this exercise, you'll examine how your results differ when using a full join versus using a left join versus using an inner join with the countries and currencies tables.

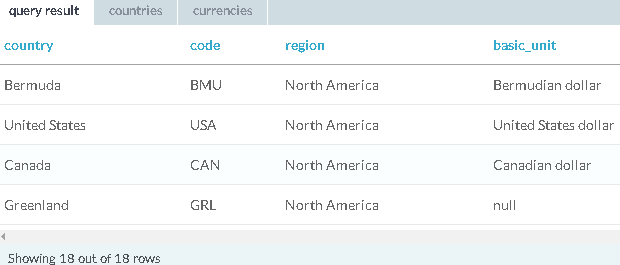
You will focus on the North American region and also where the name of the country is missing. Dig in to see what we mean!

Begin with a full join with countries on the left and currencies on the right. The fields of interest have been SELECTed for you throughout this exercise.

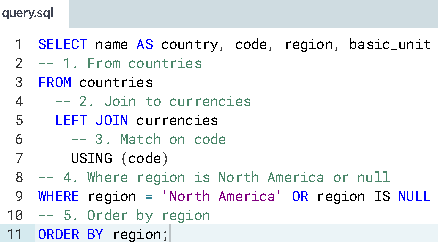
Then complete a similar left join and conclude with an inner join.

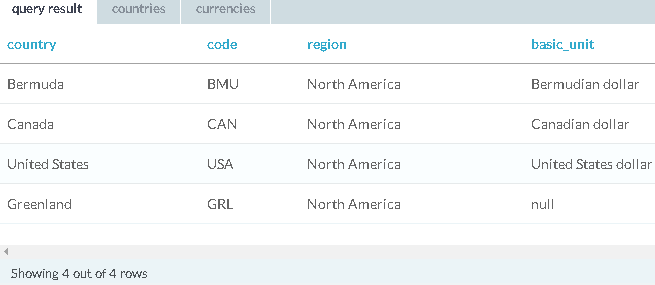
**1. Choose records in which region corresponds to North America or is NULL.**



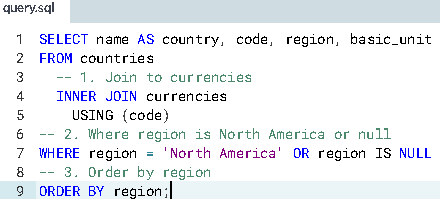


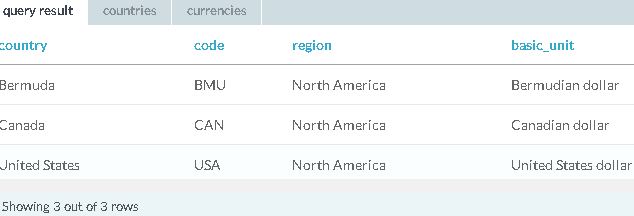
**2.** **Repeat the same query as above but use a LEFT JOIN instead of a FULL JOIN. Note what has changed compared to the FULL JOIN result!**





**3.** **Repeat the same query as above but use an INNER JOIN instead of a FULL JOIN. Note what has changed compared to the FULL JOIN and LEFT JOIN results!**





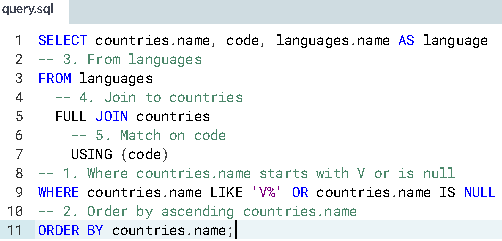
**FULL JOIN (2) EX.**

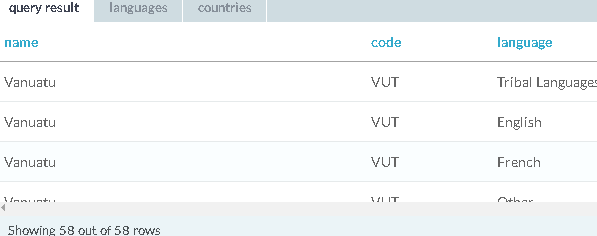
**You'll now investigate a similar exercise to the last one, but this time focused on using a table with more records on the left than the right. You'll work with the languages and countries tables.**

**Begin with a full join with languages on the left and countries on the right. Appropriate fields have been selected for you again here.**

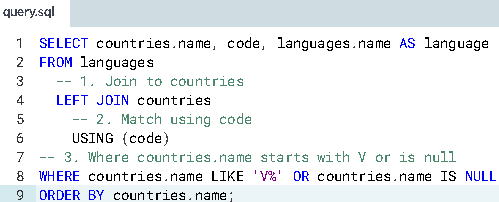
**1.** **Choose records in which countries.name starts with the capital letter 'V' or is NULL.**

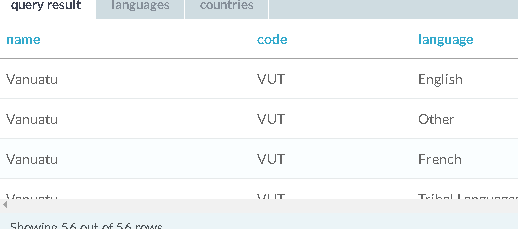
**Arrange by countries.name in ascending order to more clearly see the results.**



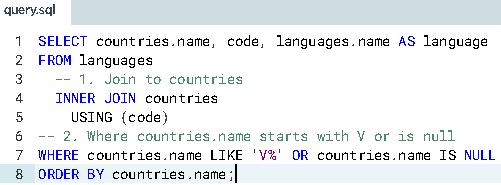


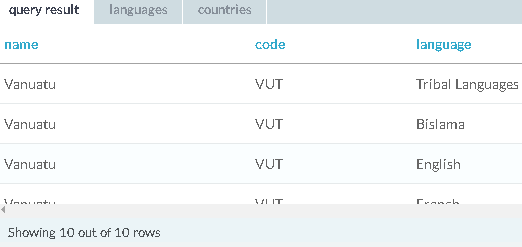
**2.** **Repeat the same query as above but use a left join instead of a full join. Note what has changed compared to the full join result!**





**3.** **Repeat once more, but use an inner join instead of a left join. Note what has changed compared to the full join and left join results.**





**FULL JOIN (3) EX**

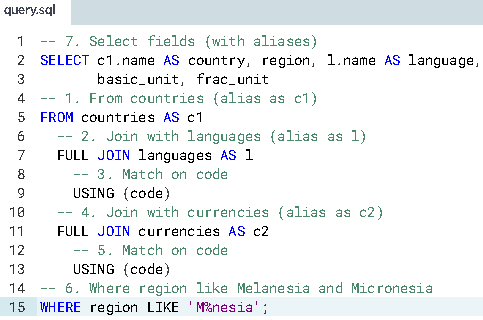
**You'll now explore using two consecutive full joins on the three tables you worked with in the previous two exercises.**

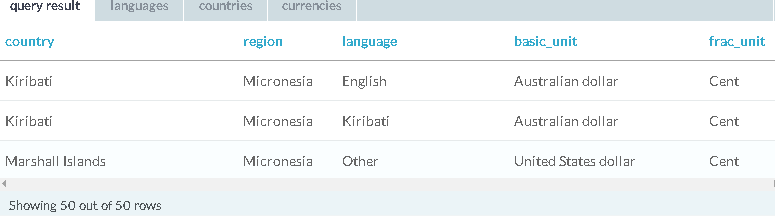
**- Complete a full join with countries on the left and languages on the right.**

**- Next, full join this result with currencies on the right.**

**- Use LIKE to choose the Melanesia and Micronesia regions (Hint: 'M%esia').**

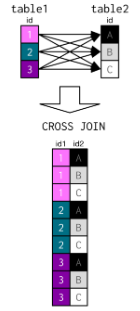
**- Select the fields corresponding to the country name AS country, region, language name AS language, and basic and fractional units of currency.**



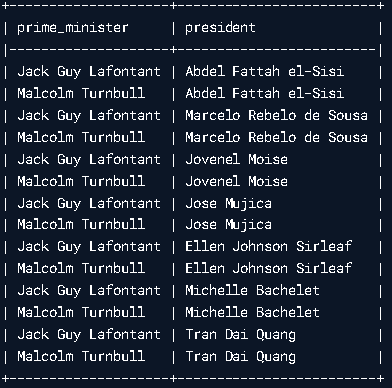
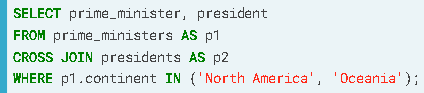


**CROSSing the Rubicon**

Agora que já vimos INNER JOINs e OUTER JOINs é hora de aprender CROSS JOIN. CROSS JOINs criam todas as combinasções possíveis de duas tabelas. Veja no diagrama temos as tabelas 1 e 2. Cada tabela tem apenas um campo de nome id. O resultado do CROSS JOIN são 9 combinações dos valores dos ids 1, 2 e 3 na tabela 1 com os valores de id A, B e C da tabela 2.



Vamos ver um ex da base de dados dos líderes. O WHERE foca só nos prime\_ministers de North America e Oceania. O resultado da query nos dá o emparelhamento dos dois prime\_ministers de North America e Oceania da tabela prime\_ministers com os 7 presidentes da tabela presidents.



**Ex. A table of two cities**

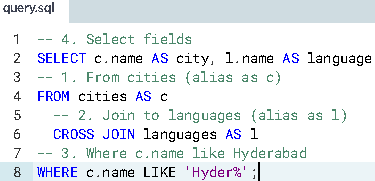
**Passo 1: This exercise looks to explore languages potentially and most frequently spoken in the cities of Hyderabad, India and Hyderabad, Pakistan.**

**You will begin with a cross join with cities AS c on the left and languages AS l on the right. Then you will modify the query using an inner join in the next tab.**

**- Create the cross join as described above. (Recall that cross joins do not use ON or USING.)**

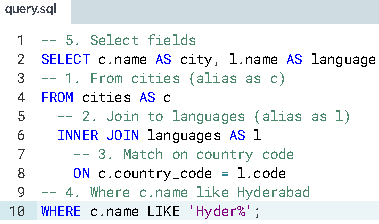
**- Make use of LIKE and Hyder% to choose Hyderabad in both countries.**

**- Select only the city name AS city and language name AS language.**





**Passo 2: Use an inner join instead of a cross join. Think about what the difference will be in the results for this inner join result and the one for the cross join.**





**EX. OUTER CHALLENGE**

**In terms of life expectancy for 2010, determine the names of the lowest five countries and their regions.**

**- Select country name AS country, region, and life expectancy AS life\_exp.**

**- Make sure to use LEFT JOIN, WHERE, ORDER BY, and LIMIT.**