Congratulations! You passed!
Grade received 100% To pass 100% or higher

Go to next item

# Hands-On Activity: Queries for JOINS

**Total points** 2

1. 1 / 1 point

## Activity overview

You've come a long way in working with relational databases and SQL. Now, you'll gain practice writing queries that join multiple tables together.

In this activity, you'll work with the World Bank's International Education Dataset. By mastering JOIN statements, you'll be able to fully harness the power of relational databases by combining data from tables linked by keys.

## Load and examine the dataset

1. Log in to BigQuery Sandbox. If you have a free trial version of BigQuery, you can use that instead. On the BigQuery page, click the **Go to BigQuery** button.

- **Note:** BigQuery Sandbox frequently updates its user interface. The latest changes may not be reflected in the screenshots presented in this activity, but the principles remain the same. Adapting to changes in software updates is an essential skill for data analysts, and it's helpful for you to practice troubleshooting. You can also reach out to your community of learners on the discussion forum for help.

2. If you have never created a BigQuery project before, click **CREATE PROJECT** on the right side of the screen. If you have created a project before, you can use an existing one or create a new one by clicking the project dropdown in the blue header bar and selecting **NEW PROJECT**.

3. Name your project something that will help you identify it later. You can give it a unique project ID or use an auto-generated one. Don't worry about selecting an organization if you don't know what to put.

4. Now, you'll see the **Editor** interface. In the middle of the screen is a window where you can type code, and to the left is the **Explorer** menu where you can search for datasets.

Before you begin joining two tables together, you'll first need to figure out which tables to join together. Recall that two tables must be connected in order to join them. Two tables can be joined if the **primary key** for one table is included in the other table as a **foreign key**.

To determine what information the tables contain and identify keys you can join them on, review the schema of the tables. To access the schema:

5. Click **+ ADD DATA** at the top of the Explorer menu, then **Explore public datasets** from the resulting dropdown.

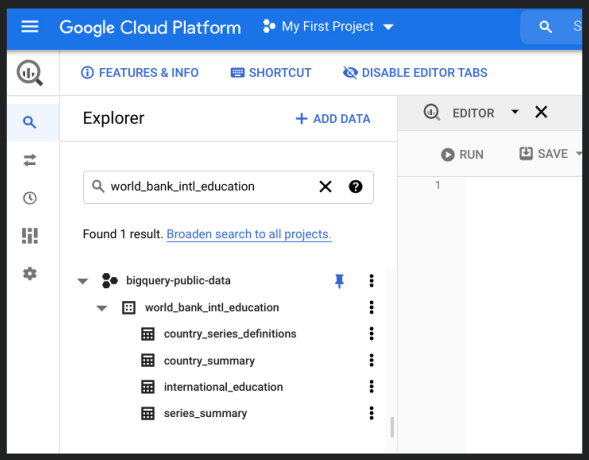
6. In the **Search Marketplace** bar, type **international education**.

7. Click the first result, The World Bank's **International Education** dataset.

8. Click **View Dataset**. This will bring you back to the BigQuery Sandbox interface in a new tab.

- **Note:** This may pin the **bigquery-public-data** dropdown to the **Explorer** menu. You can use this to browse datasets and tables.

9. In the Explorer menu, search for **world_bank_intl_education**. Click the dropdown arrow to expand the dataset.



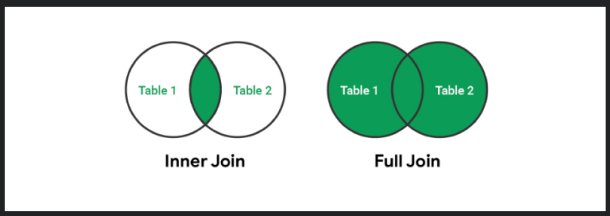

10. Click the **international_education** table. This will bring up the table's schema. If the schema doesn't appear, click on the **Schema** tab in the table viewer.

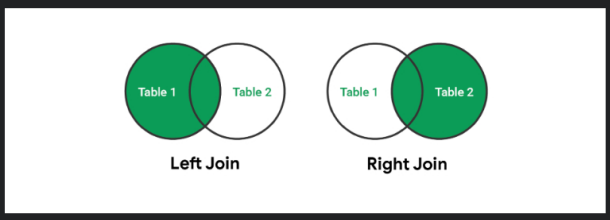
11. Next, select the **country_summary** table and examine its schema. You'll find that the country_code column appears in both the table schemas.

- **Note:** Foreign keys don't always have the same names across tables. If you're ever unsure if the columns are the same, you can always double-check. You can query the column from each table and confirm that they contain the same kinds of information.

## Review JOINs

Before you join the two tables with queries, take a moment to review the various kinds of JOIN statements.




The two most common kinds of JOIN statements are INNER JOINs and OUTER LEFT JOINs (also known simply as LEFT JOINs). As a review:

- **INNER JOIN:** Returns only the rows where the target appears in both tables.
- **LEFT JOIN:** Returns every row from the left table, as well as any rows from the right table with matching keys found in the left table.

Note the difference between the INNER JOIN and LEFT JOIN and the implications for when each should be used. Consider the following queries:

```
-- Using INNER JOIN --> We get 10 rows in our results.
SELECT
    COUNT(*)
FROM
    table_1
INNER JOIN
    table_2
ON table_1.key = table_2.key;

-- Using LEFT JOIN --> We get 100 rows in our results.
SELECT
    COUNT(*)
FROM
    table_1
LEFT JOIN
    table_2
ON table_1.key = table_2.key;
```

The takeaway here is that the sort of JOIN you use matters. When writing a query, you can draw out a Venn diagram like the example graphic above to help you decide which sort of JOIN you need.

## Queries with JOINS and aliases

Now, it's time to actually query the dataset. As a starting point, try a query that pulls information from both the international_education and country_summary tables. Copy, paste, and run the following query:

```
SELECT
    `bigquery-public-data.world_bank_intl_education.international_education`.country_name,
    `bigquery-public-data.world_bank_intl_education.country_summary`.country_code,
    `bigquery-public-data.world_bank_intl_education.international_education`.value
FROM
    `bigquery-public-data.world_bank_intl_education.international_education`
INNER JOIN
    `bigquery-public-data.world_bank_intl_education.country_summary`
ON `bigquery-public-data.world_bank_intl_education.country_summary`.country_code = `bigqu
```

This basic query joins the tables on the country_code foreign key, and returns the country name, country code, and value column. This is quite a long, unwieldy query for such a basic result! The length of each table name (which must include the full address for each table for BigQuery to know where to pull the data from) makes this hard to read and work with.

However, you can solve this by setting an **alias** for each table.

### Use descriptive aliases

Try using descriptive aliases that tell you what they represent. This next query is the same query as the previous one, but with aliases to improve readability. Copy, paste, and run the following query:

```
    `bigquery-public-data.world_bank_intl_education.international_education` AS edu
INNER JOIN
    `bigquery-public-data.world_bank_intl_education.country_summary` AS summary
ON edu.country_code = summary.country_code
```

Your results should appear like this:

Query complete (8.7 sec elapsed, 119.9 MB processed)

Job information | Results | JSON | Execution details

| Row | country_name | country_code | value |
|---|---|---|---|
| 1 | Cabo Verde | CPV | 5238.0 |
| 2 | Chad | TCD | 1.4452543E7 |
| 3 | French Polynesia | PYF | 33810.0 |
| 4 | Georgia | GEO | 1.43780167291587E10 |
| 5 | Grenada | GRD | 9100.0 |
| 6 | Marshall Islands | MHL | 14.0 |

This query is much easier to read and understand. Recall that you can set aliases for tables by specifying the alias for the table after the table's name in FROM and/or JOIN statements.

For this example, the international_education table was renamed as **edu**, and the country_summary table as **summary**. Using descriptive aliases is a best practice and will help you keep your queries clean, readable, and easy to work with.

## Use a JOIN to answer a question

Now that you've confirmed that the JOIN statement works, try to answer an actual data question using this dataset. What is the average amount of money spent per region on education? Copy, paste, and run the following query:

```
SELECT
    AVG(edu.value) average_value, summary.region
FROM
    `bigquery-public-data.world_bank_intl_education.international_education` AS edu
INNER JOIN
    `bigquery-public-data.world_bank_intl_education.country_summary` AS summary
ON edu.country_code = summary.country_code
WHERE summary.region IS NOT null
GROUP BY summary.region
ORDER BY average_value DESC
```

Your results should appear like this:

Query complete (1.3 sec elapsed, 63 MB processed)

Job information | Results | JSON | Execution details

| Row | average_value | region |
|---|---|---|
| 1 | 4.165343760778633E10 | North America |
| 2 | 3.882406275688643E9 | East Asia & Pacific |
| 3 | 2.6965348138836946E9 | South Asia |
| 4 | 2.3741490290454454E9 | Europe & Central Asia |
| 5 | 9.734776653314434E8 | Middle East & North Africa |
| 6 | 9.6599906854756E8 | Latin America & Caribbean |
| 7 | 2.1169202870654103E8 | Sub-Saharan Africa |

Notice how in this query, an alias is also set to give the **AVG(edu.value)** a more descriptive name for the temporary table the query returns.

Also note that the WHERE statement excludes rows with any null information. This is necessary to present the data succinctly and display only seven rows for the seven regions represented in the data. However, this WHERE statement means that the results will return the same regardless of which JOIN you use. In the next section, you'll explore a situation where you need to use a specific kind of join in your query...

## INNER JOINs versus OUTER JOINs

In the last query, you used an INNER JOIN to find the average amount of money spent per region on education. Because of the WHERE statement in this query, using any kind of JOIN produces the same result.

Now, you will write a LEFT JOIN, a type of OUTER JOIN, for a situation where the type of query you use will change the result you return.

Consider this scenario:

You have been tasked to provide data for a feature sports article on Michael Jordan's basketball career. The writer wants to include a funny twist and asks you to find out if Michael Jordan played better at schools with animal mascots.

To analyze his early career, you start with the years he played basketball in college. You need to examine National Collegiate Athletic Association (NCAA) college basketball stats from 1984.

You'll need a list of all NCAA Division I colleges and universities; their mascots, if applicable; and their number of wins and losses. You can find this information in the public dataset **ncaa_basketball** on BigQuery.

Next, you will write a query. Your query should join the season statistics from one table with the mascot

information from another. You need to use a LEFT JOIN instead of an INNER JOIN because not all teams have mascots. If you use an INNER JOIN, you would exclude teams with no mascot.

To demonstrate this, copy, paste, and run the following query:

```
18    seasons.market
```

This is an example of when a LEFT JOIN is more helpful than an INNER JOIN. With this query, you can look at college basketball statistics to get a better sense of Michael Jordan's early career, find out more information about which teams had mascots, and answer your business question.

## Confirmation and reflection

In the last query, you use a LEFT JOIN instead of an INNER JOIN to find the correct information. Beneath the query results, you'll find that the number of rows in your joined table is 281. If you rerun the query with an INNER JOIN instead of a LEFT JOIN, how many rows would it return?

- (•) 274
- ( ) 281
- ( ) 301
- ( ) 324

**Correct**
The number of rows returned by an INNER JOIN is 274. When you run the query with an INNER JOIN instead of a LEFT JOIN, you exclude universities without mascots and return fewer rows of data. Knowing which JOIN to use is very important for analyzing data. Going forward, you can use your knowledge of JOINs to properly combine data from multiple tables.

**2.** In this activity, you used JOIN statements to combine data from multiple tables. In the text box below, write 2-3 sentences (40-60 words) in response to each of the following questions. **1 / 1 point**

- Why do you think JOIN statements are important for working with databases?
- How do you distinguish INNER JOINS from OUTER JOINS?

Why do you think JOIN statements are important for working with databases?
Because one can Join different tables.
How do you distinguish INNER JOINS from OUTER JOINS?
Inner Join will return only the rows where the target appears in both tables.
Outer joins OUTER JOIN returns all records from both tables even if data isn't populated in one of the tables.

**Correct**
Congratulations on completing this hands-on activity! A good response would include that JOINs allow you to combine data from linked tables, which helps you make comparisons and answer business questions.

Mastering JOIN statements is one of the most important parts of SQL, as combining data from multiple database tables is a core skill for data analysts. And when you apply for jobs, remember that JOIN statements are a common topic in data analyst interviews! The more JOIN statements you write, the more prepared you will be for a data analyst role.