### Understanding JOINS

1. INNER JOINS: (Intersection of two tables)
2. OUTER JOINS: (Union of two tables)
3. LEFT JOIN: A function that will return all the records from the left table and only the matching records from the right table.
4. RIGHT JOIN: The opposite of RIGHT JOIN

The table mentioned first is left, and the table mentioned second is right.

Examples:

| SELECT \* FROM  tableA LEFT JOIN  tableB ON  keyA=keyB; |
| --- |

Query Example:

| SELECT  Employees.name AS employee\_name,  Employees.role AS employee\_role,  Departments.name AS department\_name FROM  Employees INNER JOIN  Departments ON  Employees.department\_id = departments.department\_id; |
| --- |

| SELECT  Employees.name AS employee\_name,  Employees.role AS employee\_role,  Departments.name AS department\_name FROM  Employees LEFT JOIN  Departments ON  Employees.department\_id = departments.department\_id; |
| --- |

| SELECT  Employees.name AS employee\_name,  Employees.role AS employee\_role,  Departments.name AS department\_name FROM  Employees RIGHT JOIN  Departments ON  Employees.department\_id = departments.department\_id; |
| --- |

| SELECT  Employees.name AS employee\_name,  Employees.role AS employee\_role,  Departments.name AS department\_name FROM  Employees FULL OUTER JOIN  Departments ON  Employees.department\_id = departments.department\_id; |
| --- |

### Secret identities: The Importance of aliases

Aliases are used in SQL queries to create temporary names for a column or table. . Aliases make referencing tables and columns in your SQL queries much simpler when you have table or column names that are too long or complex to make use of in queries. Imagine a table name like special\_projects\_customer\_negotiation\_mileages. That would be difficult to retype every time you use that table. With an alias, you can create a meaningful nickname that you can use for your analysis. In this case “special\_projects\_customer\_negotiation\_mileages” can be aliased to simply “mileage.” Instead of having to write out the long table name, you can use a meaningful nickname that you decide.

#### Basic syntax for aliasing:

Aliasing is the process of using aliases.

| SELECT column\_names(s) FROM table\_name AS alias\_name; |
| --- |

Notice that AS is preceded by the table name and followed by the new nickname.

### Using JOINs effectively

In this reading, you will review how JOINs are used and will be introduced to some resources that you can use to learn more about them. A JOIN combines tables by using a primary or foreign keys to align the information coming from the tables in the combination process.

#### The general JOIN syntax

| SELECT  -- table columns from tables are inserted here  Table\_name1.column\_name  Table\_name2.column\_name FROM  Table\_name1 JOIN  Table\_name2 ON table\_name1.column\_name = table\_name2.column\_name |
| --- |

As you can see from the syntax, the JOIN statement is part of the FROM clause of the query. JOIN in SQL indicates that you are going to combine data from two tables. ON in SQL identifies how the tables are to be matched for the correct information to be combined from both.

| -- Let's say table\_1 has 100 rows and table\_2 has 10 rows. -- They share 10 keys in common.  -- 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; |
| --- |

## Queries in BigQuery JOINS and aliases

| 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 = `bigquery-public-data.world\_bank\_intl\_education.international\_education`.country\_code |
| --- |

### Use descriptive aliases

| SELECT  edu.country\_name,  summary.country\_code,  edu.value 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 |
| --- |

### Use a JOIN to answer a question

What is the average amount of money spent per region on education?

| 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 |
| --- |

### INNER JOINs versus OUTER JOINs

With LEFT JOIN

| SELECT seasons.market AS university, seasons.name AS team\_name, seasons.wins, seasons.losses, seasons.ties, mascots.mascot AS team\_mascot FROM `bigquery-public-data.ncaa\_basketball.mbb\_historical\_teams\_seasons` AS seasons LEFT JOIN `bigquery-public-data.ncaa\_basketball.mascots` AS mascots ON seasons.team\_id = mascots.id WHERE seasons.season = 1984 AND seasons.division = 1 ORDER BY seasons.market |
| --- |

With INNER JOIN

| SELECT seasons.market AS university, seasons.name AS team\_name, seasons.wins, seasons.losses, seasons.ties, mascots.mascot AS team\_mascot FROM `bigquery-public-data.ncaa\_basketball.mbb\_historical\_teams\_seasons` AS seasons INNER JOIN `bigquery-public-data.ncaa\_basketball.mascots` AS mascots ON seasons.team\_id = mascots.id WHERE seasons.season = 1984 AND seasons.division = 1 ORDER BY seasons.market |
| --- |

## Work with Subqueries

### Queries within Queries

Subqueries are a lot like nesting dolls. Your larger query can have a subquery in it.

You can get more done with a single query.

Outer query and Inner query

Inner query executes first.

To compare the number of bikes available at a station to the average number of bikes available.

| SELECT  station\_id,  num\_bikes\_available,  (SELECT  AVG(num\_bikes\_available)  FROM `bigquery-public-data.new\_york.citibike.stations`) AS avg\_num\_bikes\_available FROM  `bigquery-public-data.new\_york.citibike\_stations`; |
| --- |

### Using subqueries to aggregate data

**Having**: Allows you to add a filter to your query instead of the underlying table that can only be used with aggregate functions.