How to Retrieve Data from Multiple Tables

The JOIN clause

- A JOIN clause allows you to access data from two or more tables in a query.
- A join links to tables on a common key between the two tables. Usually the primary key on one table is compared to the foreign key on another table using the equals (=) sign. This is an *equijoin* or an *inner-join.* However, other comparison operators are also valid.
- If column names from each table in the join have the same name, they must be *qualified* with the table name or a *table alias.*

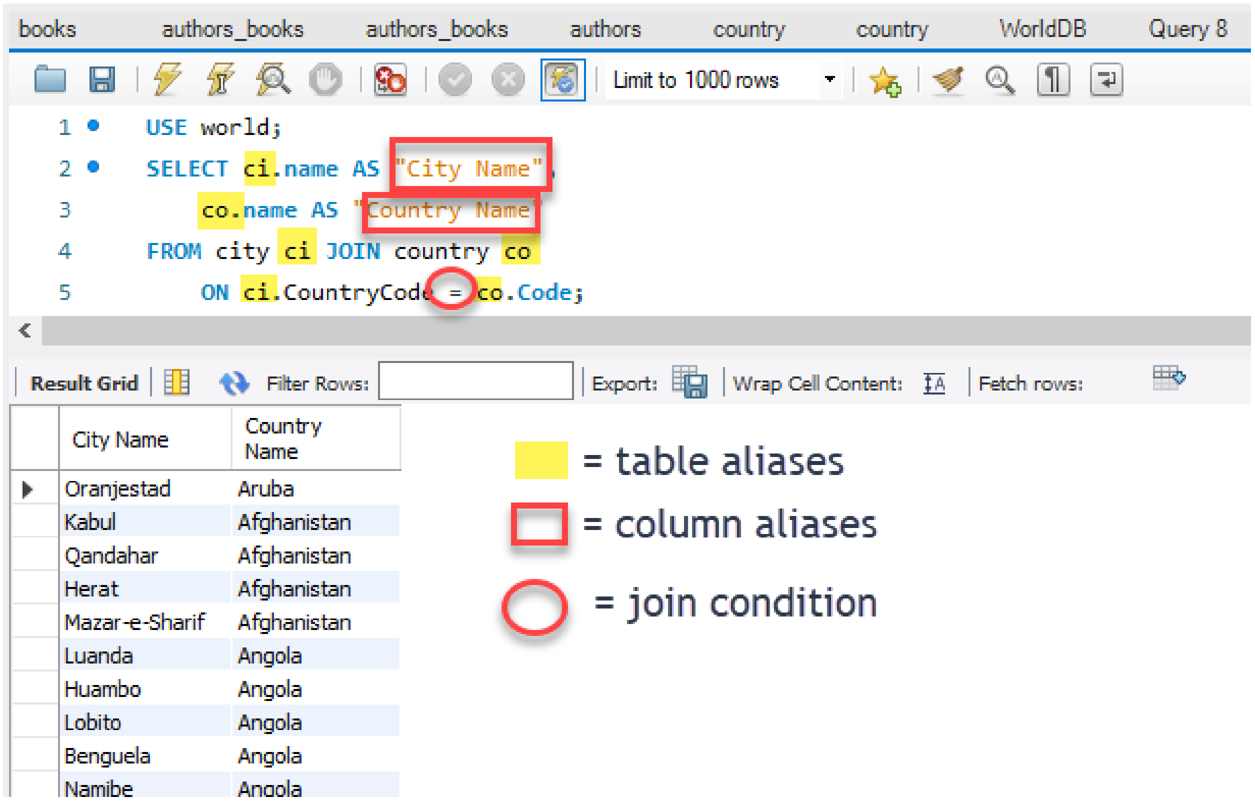
Below is a basic example of a SQL statement with an inner join clause using **explicit** syntax.

```
USE world;
SELECT city.name AS "City Name",
       country.name AS "Country Name"
FROM country
       JOIN city
              ON city.CountryCode = country.Code;
```

You could right SQL statement more succinctly with an inner join clause using *table aliases*. Instead of writing out the whole table name to qualify a column, you can use a table alias.

```
USE world;
SELECT ci.name AS "City Name",
       co.name AS "Country Name"
FROM city ci
       JOIN country co
              ON ci.CountryCode = co.Code;
```

The results of the join query would yield the same results as shown below whether or not table names are explicitly written out or are represented with table aliases. The table aliases of **co** for country and **ci** for city are defined in the FROM clause and referenced in the SELECT and ON clause:



```
USE world;
SELECT ci.name AS "City Name",
    co.name AS "Country Name"
FROM city ci JOIN country co
    ON ci.CountryCode = co.Code;
```

City Name	Country Name
Oranjestad	Aruba
Kabul	Afghanistan
Qandahar	Afghanistan
Herat	Afghanistan
Mazar-e-Sharif	Afghanistan
Luanda	Angola
Huambo	Angola
Lobito	Angola
Benguela	Angola
Namibe	Angola

🟨 = table aliases

🟥 = column aliases

○ = join condition

Let us break the statement line by line:

`USE world;`

- The **USE** clause sets the database that we will be querying. You typically have more than one database on your database server. You have to specify which database you are working in.
- The semicolon "**;**" indicates the end of a statement. You can execute multiple statements in sequence by defining each statement with a semicolon

`SELECT ci.name AS “City Name”, co.name AS “Country Name”`

- The **SELECT** clause defines the columns and column order that you want to retrieve in your result set. In this example, we have columns from two separate tables. These columns have the same name, so they MUST be qualified with the full table name or a table alias. Otherwise, the column names are ambiguous.
- You separate each column name with a comma "," including the corresponding table alias if one is provided

- To create a friendlier column name in the output, we assign a *column alias* to each qualified column name. Instead of **ci.name** showing in the column header of the report, we assign a friendlier column alias of “City Name” and for **co.name** “Country Name.”

```
FROM city ci
```

- The **FROM** clause specifies the table(s) from which results will be returned.
- In a **JOIN** clause the first table to be joined is specified after the **FROM** clause.

```
JOIN country co
```

- Use a **JOIN** clause between the two tables.
- Include the alias if desired.

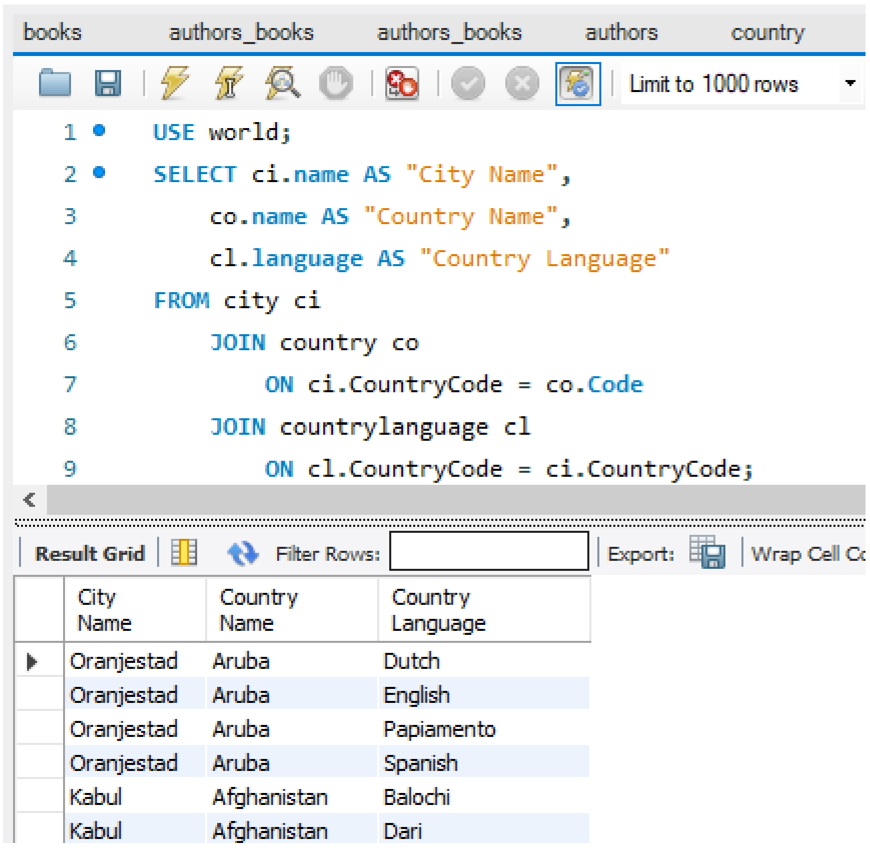
```
ON ci.CountryCode = co.Code;
```

- The **ON** clause specifies the common column from each table (usually a PK in one table and its corresponding foreign key in the other). Each column name is separated with an operator (join condition usually the equals (=) sign.

Joining more than two tables

- You can join more than two tables in a query by simply adding additional JOIN clauses.
- Once a table has been used in a JOIN clause, you don’t need to write it out again in subsequent Join statements.

```
USE world;
SELECT ci.name AS "City Name",
       co.name AS "Country Name",
       cl.language AS "Country Language"
FROM city ci
       JOIN country co
              ON ci.CountryCode = co.Code
       JOIN country language cl
              ON cl.CountryCode = ci.CountryCode;
```



```
USE world;
SELECT ci.name AS "City Name",
    co.name AS "Country Name",
    cl.language AS "Country Language"
FROM city ci
    JOIN country co
        ON ci.CountryCode = co.Code
    JOIN countrylanguage cl
        ON cl.CountryCode = ci.CountryCode;
```

City Name	Country Name	Country Language
Oranjestad	Aruba	Dutch
Oranjestad	Aruba	English
Oranjestad	Aruba	Papiamento
Oranjestad	Aruba	Spanish
Kabul	Afghanistan	Balochi
Kabul	Afghanistan	Dari

`JOIN countrylanguage cl`

- To include more tables in the query, you simply add more additional JOIN clauses.
- The “cl” is the alias for countrylanguage.
- You can refer to tables already specified in a previous join.

`ON cl.CountryCode = ci.CountryCode;`

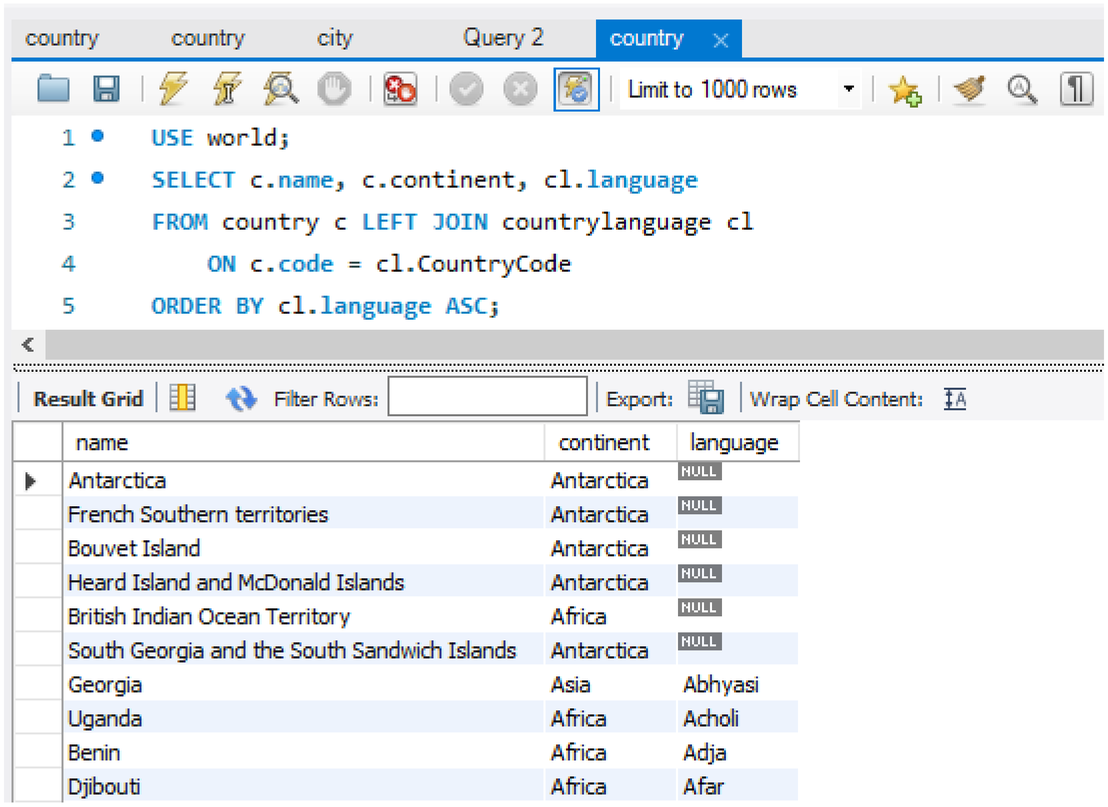
- The common column between the two tables being joined is the CountryCode column from the countrylanguage table and the CountryCode column from the city table.
- The “cl” alias previously defined for countrylanguage is used to specify the CountryCode column.

The Outer Join Clause

- An outer join will return all the rows from one table and only the rows from the other table that match the join condition.
- You can use LEFT JOIN or RIGHT JOIN. If you use LEFT JOIN, all the rows from the table on the left of the equals (=) sign will be included in the result set whether the join condition is satisfied or not.
- If you use RIGHT JOIN, all the rows from the table on the right of the equals (=) sign will be included in the result set whether the join condition is satisfied or not.

Below is a basic example of a SQL statement with an outer join clause.

```
USE world;
SELECT c.name, c.continent, cl.language
FROM country c LEFT JOIN countrylanguage cl
     ON c.code = cl.CountryCode
ORDER BY cl.language ASC;
```



```
USE world;
SELECT c.name, c.continent, cl.language
FROM country c LEFT JOIN countrylanguage cl
    ON c.code = cl.CountryCode
ORDER BY cl.language ASC;
```

name	continent	language
Antarctica	Antarctica	NULL
French Southern territories	Antarctica	NULL
Bouvet Island	Antarctica	NULL
Heard Island and McDonald Islands	Antarctica	NULL
British Indian Ocean Territory	Africa	NULL
South Georgia and the South Sandwich Islands	Antarctica	NULL
Georgia	Asia	Abhyasi
Uganda	Africa	Acholi
Benin	Africa	Adja
Djibouti	Africa	Afar

```
SELECT c.name, c.continent, cl.language
```

- The “**c.”** pre-pended to name and continent is a table alias to the country table. Therefore, return name and continent from the country table.
- The “**cl**” prepended to the language table is a table alias to the countrylanguage table. Therefore, return language from the countryLanguage table.

```
FROM country c LEFT JOIN countrylanguage cl
```

- “Country c” assigns “c” as an alias for “country”
- “countrylanguage cl” assigns “cl” as an alias for “countrylanguage”
- LEFT JOIN means that all rows on the left side of the JOIN operator (=) are included in the results whether they have a matching key from the table on the RIGHT side of the operator.

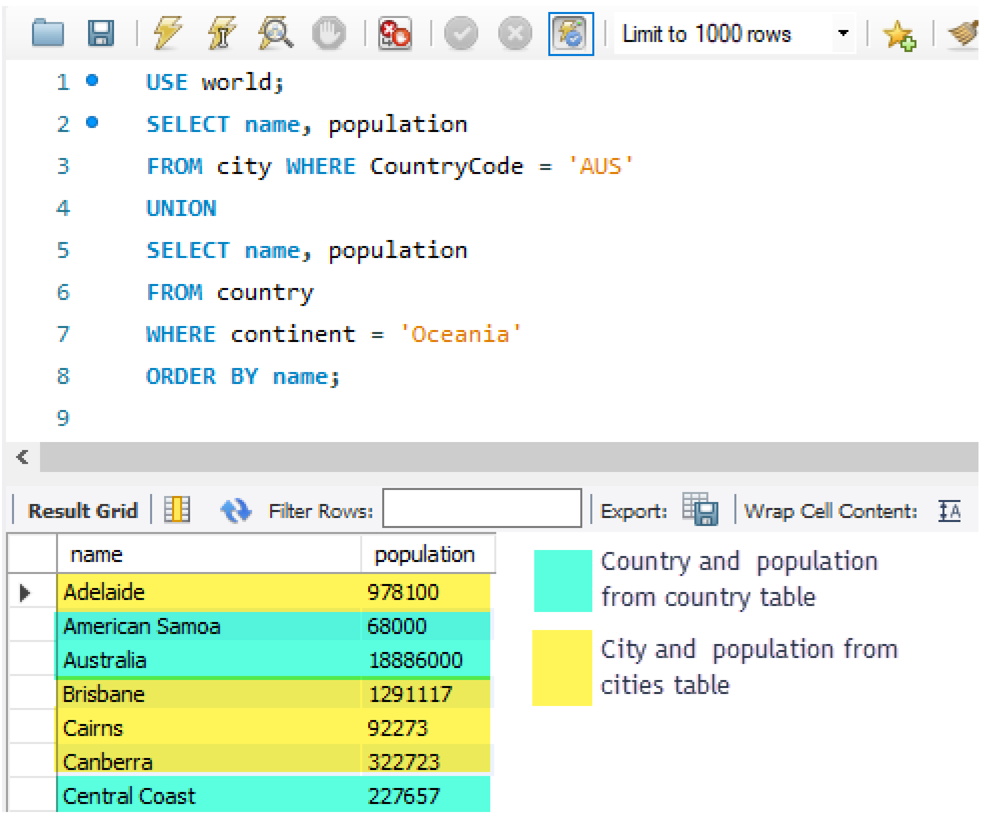
```
ON c.code = cl.CountryCode
```

- ON is the second part of the JOIN clause. It precedes the JOIN condition
- c.code refers to the code column from the country table and is a primary key. Since the key is on the LEFT side of the join condition, all rows from the country table will be included in the results whether they have a matching key in the countrylanguage table or not.
- Cl.CountryCode refers to the CountryCode on the countrylanguage table and is a foreign key to the country table. Only the rows that have a matching key in the country table will be included in the results.

How to Code a Union

- A UNION combines the results of two or more queries into a single result set
- Each result set must have the same number of columns
- The corresponding data types for each column must be compatible. However, the column names may be different from each result set.
- A UNION removes duplicate rows by default.
- You may interfile the results using an ORDERY BY clause if there is a column with a common name.

```
USE world;
SELECT name, population
FROM city WHERE CountryCode = 'AUS'
UNION
SELECT name, population
FROM country
WHERE continent = 'Oceania'
ORDER BY name;
```



```
USE world;
SELECT name, population
FROM city WHERE CountryCode = 'AUS'
UNION
SELECT name, population
FROM country
WHERE continent = 'Oceania'
ORDER BY name;
```

name	population
Adelaide	978100
American Samoa	68000
Australia	18886000
Brisbane	1291117
Cairns	92273
Canberra	322723
Central Coast	227657

🟦	Country and population from country table
🟨	City and population from cities table

```
SELECT name, population
FROM city
WHERE CountryCode = 'AUS'
```

- The first query returns the name and population from the city table.
- The filter (WHERE CLAUSE) of the query limits the country code to Australia.

```
UNION
```

- The ‘UNION’ clause will combine this query with the results of the subsequent query.

```
SELECT name, population
FROM country
WHERE continent = 'Oceania'
```

- The second query returns the name and population from the country table.
- The filter (WHERE CLAUSE) of the query limits the continent code to Oceania.

```
ORDER BY name;
```

- It is possible to sort (ORDER BY CLAUSE) and interfile the results of both queries because each query share a column with the same name. Otherwise, the ORDER BY clause would give an error.