Inner join

1. Begin by selecting all columns from the `cities` table.

```
select *

from cities;
```

2. Inner join the `cities` table on the left to the `countries` table on the right, keeping all of the fields in both tables. You should join on the `country_code` field in `cities` and the `code` field in `countries`. **Do not** alias your tables here or in the next task though. Using `cities` and `countries` is fine for now.

```
select *

from cities

inner join countries

on cities.country_code = countries.code;
```

3. Modify the `SELECT` statement to keep only the name of the city, the name of the country, and the name of the region the country resides in.

   Recall from our **Intro to SQL for Data Science course that you can alias fields using** `AS`. Alias the name of the city `AS city` and the name of the country `AS country`.

```
select *

from cities

inner join countries

on cities.country_code = countries.code;
```

Inner join (2)

- Join the tables `countries` (left) and `economies` (right). What field do you need to use in `ON` to match the two tables?
- Alias `countries AS c` and `economies AS e`.
- From this join, `SELECT`:
  - `c.code`, aliased as `country_code`.
  - `name`, `year`, and `inflation_rate`, not aliased.

```
SELECT c.code AS country_code, name, year, inflation_rate
```

FROM countries AS c

inner JOIN economies AS e

ON c.code = e.code;

Inner join (3)

- Inner join `countries` (left) and `populations` (right) on the `code` and `country_code` fields respectively.
- Alias `countries AS c` and `populations AS p`.
- Select `code`, `name`, and `region` from `countries` and also select `year` and `fertility_rate` from `populations` (5 fields in total).

SELECT c.code, c.name, c.region, p.year, p.fertility_rate

FROM countries as c

INNER JOIN populations as p

ON c.code = p.country_code;

- Add an additional inner join with `economies` to your previous query by joining on `code`.
- Include the `unemployment_rate` column that became available through joining with `economies`.
- Note that `year` appears in both `populations` and `economies`, so you have to explicitly use `e.year` instead of `year` as you did before.

SELECT c.code, name, region, e.year, fertility_rate, e.unemployment_rate

FROM countries AS c

INNER JOIN populations AS p

ON c.code = p.country_code

INNER JOIN economies as e

ON c.code = e.code;

- Scroll down the query result and take a look at the results for Albania from your previous query. Does something seem off to you?
- The trouble with doing your last join on `c.code = e.code` and not also including `year` is that e.g. the 2010 value for `fertility_rate` is also paired with the 2015 value for `unemployment_rate`.

- Fix your previous query: in your last ON clause, use AND to add an additional joining condition. In addition to joining on `code` in `c` and `e`, also join on `year` in `e` and `p`.
- SELECT c.code, name, region, e.year, fertility_rate, unemployment_rate

FROM countries AS c

INNER JOIN populations AS p

ON c.code = p.country_code

INNER JOIN economies AS e

on c.code=e.code and p.year=e.year;

Review inner join using on

Ans ==> `INNER JOIN` requires a specification of the key field (or fields) in each table.

Inner join with using

1. Inner join `countries` on the left and `languages` on the right with `USING(code)`. Select the fields corresponding to:
- country name `AS country`,
- continent name,
- language name `AS language`, and
- whether or not the language is official.

Remember to alias your tables using the first letter of their names.

SELECT c.name AS country, c.continent, l.name AS language, l.official

FROM countries AS c

inner JOIN languages AS l

using(code);

Self-join

- Join `populations` with itself ON `country_code`.
- Select the `country_code` from `p1`.
- Select 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`.

```
SELECT p1.country_code,
    p1.size AS size2010,
    p2.size AS size2015
FROM populations AS p1
inner JOIN populations AS p2
ON  p1.country_code = p2.country_code;
```

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.

```
SELECT p1.country_code,
    p1.size AS size2010,
    p2.size AS size2015
FROM populations AS p1
inner JOIN populations AS p2
ON  p1.country_code = p2.country_code
AND p1.year = p2.year - 5;
```

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 AA and BB, the percentage growth from AA to BB can be calculated as (B−A)/A∗100.0$(B-A)/A*100.0$.

   To `SELECT` add a new field aliased as `growth_perc` that calculates the percentage population growth from 2010 to 2015 for each country, using `p2.size` and `p1.size`.

```
SELECT p1.country_code,
    p1.size AS size2010,
    p2.size AS size2015,
    ((p2.size - p1.size)/p1.size * 100.0) AS growth_perc
```

```
FROM populations AS p1
inner JOIN populations AS p2
ON  p1.country_code = p2.country_code
AND p1.year = p2.year - 5;
```

Case when and then

1. 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'`.

```
SELECT name, continent, code, surface_area,
    -- first case
  CASE WHEN surface_area > 2000000 THEN 'large'
    -- second case
    WHEN surface_area > 350000 THEN 'medium'
    -- else clause + end
    ELSE 'small' END
    AS geosize_group
FROM countries;
```

Inner challenge

1. Using the `populations` table focused only for the `year` 2015, create a new field `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.

```
SELECT country_code, size,
```

```
    CASE WHEN size > 50000000 THEN 'large'
        WHEN size > 1000000 THEN 'medium'
        ELSE 'small' END
        AS popsize_group
FROM populations
WHERE year = 2015;
```

- 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**.

```
SELECT country_code, size,
    CASE WHEN size > 50000000 THEN 'large'
        WHEN size > 1000000 THEN 'medium'
        ELSE 'small' END
        AS popsize_group
into pop_plus
FROM populations
WHERE year = 2015 ;

select * from pop_plus;
```

- Keep the first query intact that creates `pop_plus` using `INTO`.
- Remove the `SELECT * FROM pop_plus;` code and instead write a second query to join `countries_plus AS c` on the left with `pop_plus AS p` on the right matching on the country code fields.
- Select the `name`, `continent`, `geosize_group`, and `popsize_group` fields.
- Sort the data based on `geosize_group`, in ascending order so that `large` appears on top.