Union

- Combine these two tables into one table containing all of the fields in `economies2010`. The `economies` table is also included for reference.
- Sort this resulting single table by country code and then by year, both in ascending order.

```
-- pick specified columns from 2010 table

select *

-- 2010 table will be on top

from  economies2010

-- which set theory clause?

union

-- pick specified columns from 2015 table

select *

-- 2015 table on the bottom

from  economies2015

-- order accordingly

order by code, year asc;
```

Union (2)

- Determine all (non-duplicated) country codes in either the `cities` or the `currencies` table. The result should be a table with only one field called `country_code`.
- Sort by `country_code` in alphabetical order.

```
-- pick specified columns from 2010 table

select country_code

-- 2010 table will be on top

from  cities

-- which set theory clause?

union

-- pick specified columns from 2015 table
```

```
select code

-- 2015 table on the bottom

from  currencies

-- order accordingly

order by country_code asc;
```

Union all

- Determine all combinations (include duplicates) of country code and year that exist in either the `economies` or the `populations` tables. Order by code then year.
- The result of the query should only have two columns/fields. Think about how many records this query should result in.
- You'll use code very similar to this in your next exercise after the video. Make note of this code after completing it.

```
SELECT code, year

FROM economies

union all

SELECT country_code, year

FROM populations

ORDER BY code, year;
```

Intersect

- Again, order by `code` and then by `year`, both in ascending order.
- Note the number of records here (given at the bottom of **query result**) compared to the similar `UNION ALL` query result (814 records).

```
SELECT code, year

FROM economies

intersect

SELECT country_code, year

FROM populations

ORDER BY code, year asc;
```

Intersect (2)

Use `INTERSECT` to answer this question with `countries` and `cities`!

SELECT name

FROM countries

intersect

SELECT name

FROM cities;

Review union and intersect

Ans ==> `INTERSECT`: returns only records appearing in both tables

Except

- Order the resulting field in ascending order.
- Can you spot the city/cities that are actually capital cities which this query misses?

SELECT name

FROM cities

except

SELECT capital

FROM countries

ORDER BY name asc;

Except (2)

- Order by `capital` in ascending order.
- The `cities` table contains information about 236 of the world's most populous cities. The result of your query may surprise you in terms of the number of capital cities that DO NOT appear in this list!

SELECT capital

FROM countries

except

SELECT name

FROM cities

ORDER BY capital asc;

Semi-join

1. Flash back to our **Intro to SQL for Data Science course and begin by selecting all country codes in the Middle East as a single field result using** `SELECT`, `FROM`, and `WHERE`.

select code

from countries

where region='Middle East';

- Comment out the answer to the previous tab by surrounding it in `/*` and `*/`. You'll come back to it!
- Below the commented code, select only unique languages by name appearing in the `languages` table.
- Order the resulting single field table by `name` in ascending order.

select distinct(name)

from languages

order by name;

Now combine the previous two queries into one query:

- Add a `WHERE IN` statement to the `SELECT DISTINCT` query, and use the commented out query from the first instruction in there. That way, you can determine the unique languages spoken in the Middle East.

Carefully review this result and its code after completing it. It serves as a great example of subqueries, which are the focus of Chapter 4.

select distinct(name)

from languages

where code in

(select code

from countries

where region = 'Middle East')

order by name;

Relating semi-join to a tweaked inner join

Ans ==> Distinct

Diagnosing problems using anti-join

1. Begin by determining the number of countries in `countries` that are listed in Oceania using `SELECT`, `FROM`, and `WHERE`.

```
select count(*)
from countries
 where continent = 'Oceania';
```

- Complete an inner join with `countries AS c1` on the left and `currencies AS c2` on the right to get the different currencies used in the countries of Oceania.
- Match `ON` the `code` field in the two tables.
- Include the country `code`, country `name`, and `basic_unit AS currency`.

Observe **query result** and make note of how many *different* countries are listed here

```
select c1.code, c1.name, c2.basic_unit as currency
from countries as c1
inner join currencies as c2
on c1.code = c2.code
 where continent = 'Oceania';
```

2. Note that not all countries in Oceania were listed in the resulting inner join with `currencies`. Use an anti-join to determine which countries were not included!

Use `NOT IN` and `(SELECT code FROM currencies)` as a subquery to get the country code and country name for the Oceanian countries that are not included in the `currencies` table.

```
select c1.code, c1.name
from countries as c1
where continent = 'Oceania'
and code not in
(select code from currencies);
```

Set theory challenge

- Identify the country codes that are included in either `economies` or `currencies` but not in `populations`.
- Use that result to determine the names of cities in the countries that match the specification in the previous instruction.

```
-- select the city name

select name

-- alias the table where city name resides

from cities AS c1

-- choose only records matching the result of multiple set theory clauses

WHERE country_code IN

(

   -- select appropriate field from economies AS e

   SELECT e.code

   FROM economies AS e

   -- get all additional (unique) values of the field from currencies AS c2

   union

   SELECT c2.code

   FROM currencies AS c2

   -- exclude those appearing in populations AS p

   except

   SELECT p.country_code

   FROM populations AS p

 );
```