Get the names of people, sorted by birth date:

select name from people order by birthdate;

Get the title of films released in 2000 or 2012, in the order they were released.

select title from films where release\_year IN(2000,2012) order by release\_year;

Get all details for all films except those released in 2015 and order them by duration.

select \* from films where release\_year not in (2015) order by duration;

Get the title and gross earnings for movies which begin with the letter 'M' and order the results alphabetically.

select title,gross from films where title like 'M%' order by title;

Get the IMDB score and film ID for every film from the reviews table, sorted from highest to lowest score

select imdb\_score from reviews order by imdb\_score desc;

Get the title for every film, in reverse order.

select title from films order by title desc;

Get the title and duration for every film, in order of longest duration to shortest.

select title, duration from films order by duration desc

Get the birth date and name of people in the people table, in order of when they were born and alphabetically by name.

select birthdate, name from people order by birthdate,name

Get the release year, duration, and title of films ordered by their release year and duration.

select release\_year, duration ,title from films order by release\_year, duration

Get certifications, release years, and titles of films ordered by certification (alphabetically) and release year.

select certification, release\_year, title from films order by certification ,release\_year;

Get the release year and count of films released in each year.

select release\_year, count(\*) from films group by release\_year

Get the release year and average duration of all films, grouped by release year.

select release\_year, avg(duration) from films group by release\_year;

Get the release year and largest budget for all films, grouped by release year.

select release\_year , max(budget) from films group by release\_year

Get the IMDB score and count of film reviews grouped by IMDB score in the reviews table.

select imdb\_score , count(\*) from reviews group by imdb\_score ;

Get the release year and lowest gross earnings per release year.

select release\_year , min(gross) from films group by release\_year

Get the release year, country, and highest budget spent making a film for each year, for each country. Sort your results by release year and country.

select release\_year,country, max(budget) from films group by release\_year, country order by release\_year,country

In how many different years were more than 200 movies released?

select release\_year from films group by release\_year having count(\*) >200;

select count(release\_year) from films group by release\_year having count(\*) >200;

Get the release year, budget and gross earnings for each film in the films table.

select release\_year, budget,gross from films

Modify your query so that only records with a release\_year after 1990 are included.

select release\_year, budget,gross from films where release\_year>1990

Remove the budget and gross columns, and group your results by release year.

select release\_year from films where release\_year>1990 group by release\_year

Modify your query to include the average budget and average gross earnings for the results you have so far. Alias the average budget as avg\_budget; alias the average gross earnings as avg\_gross.

select release\_year, avg(budget) as avg\_budget, avg(gross) as avg\_gross from films where release\_year>1990 group by release\_year

Modify your query so that only years with an average budget of greater than $60 million are included.

select release\_year, avg(budget) as avg\_budget, avg(gross) as avg\_gross from films where release\_year>1990 group by release\_year having avg(budget) > 60000000

Finally, modify your query to order the results from highest average gross earnings to lowest.

select release\_year, avg(budget) as avg\_budget, avg(gross) as avg\_gross from films where release\_year>1990 group by release\_year having avg(budget) > 60000000 order by avg\_gross desc

Get the country, average budget, and average gross take of countries that have made more than 10 films. Order the result by country name, and limit the number of results displayed to 5. You should alias the averages as avg\_budget and avg\_gross respectively.

select country, avg(budget) as avg\_budget , avg(gross) as avg\_gross from films group by country having count(title)>10 order by country limit 5

Select the country\_code from p1 and 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 p1 inner join populations p2 using(pop\_id)

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 JOIN populations as p2 ON p1.country\_code = p2.country\_code and p1.year = (p2.year-5)

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.Add a new field to SELECT, 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;

When and then:

SELECT name, continent, code, surface\_area,CASE WHEN surface\_area > 2000000 THEN 'large'

WHEN surface\_area >350000 THEN 'medium'ELSE 'small' END AS geosize\_group

FROM countries;

LEFT JOIN

SELECT c1.name AS city, code, c2.name AS country,region, city\_proper\_pop FROM cities AS c1

left JOIN countries AS c2 ON c1.country\_code = c2.code ORDER BY c2.code DESC;

SELECT region, avg(gdp\_percapita) AS avg\_gdp

-- From countries (alias as c)

FROM countries AS c

-- 2. Left join with economies (alias as e)

LEFT JOIN economies AS e

-- 3. Match on code fields

ON e.code = c.code

-- 4. Focus on 2010

WHERE e.year = 2010

-- Group by region

GROUP BY region order by avg\_gdp desc

-- Select fields

select country.name as country, region, life\_expectancy as life\_exp

-- From countries (alias as c)

from countries country

-- Join to populations (alias as p)

left join populations as p

-- Match on country code

on p.country\_code=country.code

-- Focus on 2010

where p.year=2010

-- Order by life\_exp

order by life\_exp

-- Limit to 5 records

limit 5

UNION:

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

select country\_code from cities union select code from currencies order by country\_code;

INTERSECT:

-- Select fields

select code,year

-- From economies

from economies

-- Set theory clause

INTERSECT

-- Select fields

SELECT COUNTRY\_CODE, YEAR

-- From populations

FROM POPULATIONS

-- Order by code and year

ORDER BY CODE, YEAR;

EXCEPT:

SELECT name

-- From cities

FROM cities

-- Set theory clause

except

-- Select field

SELECT capital

-- From countries

FROM countries

-- Order by result

ORDER BY name;

SEMI JOIN:

You are now going to use the concept of a semi-join to identify languages spoken in the Middle East.

select distinct name

-- From languages

from languages

-- Where in statement

WHERE code IN

-- Subquery

(select

code from countries where region='Middle East')

-- Order by name

order by name;

* 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 name from cities where

country\_code in(

select c2.code from currencies as c2 union select e.code from economies as e

except

select country\_code from populations)

SUBQUERY

Select all fields from populations with records corresponding to larger than 1.15 times the average you calculated in the first task for 2015. In other words, change the 100 in the example above with a subquery.

SELECT \* FROM populations WHERE life\_expectancy > 1.15 \* (SELECT AVG(life\_expectancy) FROM populations WHERE year = 2015) AND year = 2015;

SELECT local\_name, subquery.lang\_num

FROM countries,

(SELECT code, COUNT(\*) AS lang\_num

FROM languages

GROUP BY code) AS subquery

WHERE countries.code = subquery.code

ORDER BY lang\_num DESC;

Let's test your understanding of the subqueries with a challenge problem! Use a subquery to get 2015 economic data for countries that do **not** have

* gov\_form of 'Constitutional Monarchy' or
* 'Republic' in their gov\_form.

Here, gov\_form stands for the form of the government for each country. Review the different entries for gov\_form in the countries table.

* Select the country code, inflation rate, and unemployment rate.
* Order by inflation rate ascending.
* Do not use table aliasing in this exercise.

-- Select fields

SELECT code, inflation\_rate, unemployment\_rate

-- From economies

FROM economies

-- Where year is 2015 and code is not in

WHERE year = 2015 AND code not in

-- Subquery

(SELECT code

FROM countries

WHERE (gov\_form = 'Constitutional Monarchy' OR gov\_form LIKE '%Republic%'))

-- Order by inflation rate

ORDER BY inflation\_rate;

In this exercise, you'll need to get the country names and other 2015 data in the economies table and the countries table for **Central American countries with an official language**.

-- Select fields

SELECT DISTINCT name, total\_investment, imports

-- From table (with alias)

FROM countries AS c

-- Join with table (with alias)

LEFT JOIN economies AS e

-- Match on code

ON (c.code = e.code

-- and code in Subquery

AND c.code IN (

SELECT l.code

FROM languages AS l

WHERE official = 'true'

) )

-- Where region and year are correct

WHERE region = 'Central America' AND year = 2015

-- Order by field

ORDER BY name;

Let's ease up a bit and calculate the average fertility rate for each region in 2015.

* Include the name of region, its continent, and average fertility rate aliased as avg\_fert\_rate.
* Sort based on avg\_fert\_rate ascending.
* Remember that you'll need to GROUP BY all fields that aren't included in the aggregate function of SELECT.

-- Select fields

SELECT c1.region, continent , avg(fertility\_rate) AS avg\_fert\_rate

-- From left table

FROM countries AS c1

-- Join to right table

INNER JOIN populations AS p1

-- Match on join condition

ON p1.country\_code = c1.code

-- Where specific records matching some condition

WHERE p1.year = 2015

-- Group appropriately

GROUP BY region, continent

-- Order appropriately

ORDER BY avg\_fert\_rate;

You are now tasked with determining the top 10 capital cities in Europe and the Americas in terms of a calculated percentage using city\_proper\_pop and metroarea\_pop in cities.

-- Select fields

SELECT name, country\_code, city\_proper\_pop, metroarea\_pop,

-- Calculate city\_perc

city\_proper\_pop / metroarea\_pop \* 100 AS city\_perc

-- From appropriate table

FROM cities

-- Where

WHERE name IN

-- Subquery

(SELECT capital

FROM countries

WHERE (continent = 'Europe'

OR continent LIKE '%America%'))

AND metroarea\_pop IS not null

-- Order appropriately

ORDER BY city\_perc desc

-- Limit amount

LIMIT 10;