Joining Data in SQL

Chapter 1 Introduction to joins

###Inner join

1

-- Select all columns from cities

SELECT \*

FROM cities

2

SELECT \*

FROM cities

-- Inner join to countries

INNER JOIN countries

-- Match on the country codes

ON cities.country\_code = countries.code;

3

SELECT cities.name AS city, countries.name AS country, region

FROM cities

INNER JOIN countries

ON cities.country\_code = countries.code;

### Inner join (2)

SELECT c.code AS country\_code, c.name, e.year, e.inflation\_rate

FROM countries AS c

INNER JOIN economies AS e

ON c.code = e.code;

### Inner join (3)

1

-- Select fields

SELECT c.code, name, region, year, fertility\_rate

-- From countries (alias as c)

FROM countries AS c

-- Join with populations (as p)

INNER JOIN populations AS p

-- Match on country code

ON c.code = p.country\_code;

2

-- Select fields

SELECT c.code, name, region, e.year, fertility\_rate, unemployment\_rate

-- From countries (alias as c)

FROM countries AS c

-- Join to populations (as p)

INNER JOIN populations AS p

-- Match on country code

ON c.code = p.country\_code

-- Join to economies (as e)

INNER JOIN economies AS e

-- Match on country code

ON c.code = e.code;

3

SELECT c.code, c.name, c.region, p.fertility\_rate, e.year, 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

AND p.year = e.year;

### Inner join with using

-- 4. Select fields

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

-- 1. From countries (alias as c)

FROM countries AS c

-- 2. Join to languages (as l)

INNER JOIN languages AS l

-- 3. Match using code

USING(code);

### Inner join with using

-- 4. Select fields

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

-- 1. From countries (alias as c)

FROM countries AS c

-- 2. Join to languages (as l)

INNER JOIN languages AS l

-- 3. Match using code

USING(code);

### Self-join

1

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

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

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

SELECT name, continent, code, surface\_area,

-- first case

CASE WHEN surface\_area > 2000000

-- first then

THEN 'large'

-- second case

WHEN surface\_area > 350000

-- second then

THEN 'medium'

-- else clause + end

ELSE 'small' END

-- alias resulting field of CASE WHEN

AS geosize\_group

-- from the countries table

FROM countries;

### Inner challenge

1

SELECT p.country\_code, p.size,

CASE WHEN size > 50000000

THEN 'large'

WHEN size > 1000000

THEN 'medium'

ELSE 'small' END

AS popsize\_group

FROM populations AS p

WHERE year = 2015;

2

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;

3

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 c.name, c.continent, c.geosize\_group, p.popsize\_group

FROM countries\_plus AS c

INNER JOIN pop\_plus AS p

ON c.code = p.country\_code

ORDER BY geosize\_group;

Chapter 2 Outer joins and cross joins

### Left Join

SELECT c1.name AS city, code, c2.name AS country,

region, city\_proper\_pop

-- specify left table

FROM cities AS c1

-- specify right table and type of join

INNER JOIN countries AS c2

-- how should the tables be matched?

ON c1.country\_code = c2.code

-- sort based on descending country code

ORDER BY code DESC;

### Left join (2)

SELECT c.name AS country, local\_name, l.name AS language, percent

-- countries on the left (alias as c)

FROM countries AS c

-- inner join with languages (as l) on the right

INNER JOIN languages AS l

-- give fields to match on

ON c.code = l.code

-- sort by descending country name

ORDER BY country DESC;

### Left join (3)

1

SELECT name, region, gdp\_percapita

FROM countries AS c

LEFT JOIN economies AS e

ON c.code = e.code

WHERE e.year = 2010;

2

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

FROM countries AS c

LEFT JOIN economies AS e

ON c.code = e.code

WHERE e.year = 2010

GROUP BY region;

3

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

FROM countries AS c

LEFT JOIN economies AS e

ON c.code = e.code

WHERE e.year = 2010

GROUP BY region

ORDER BY avg\_gdp DESC;

### Right join

-- convert this code to use RIGHT JOINs instead of LEFT JOINs

/\*

SELECT cities.name AS city, urbanarea\_pop, countries.name AS country,

indep\_year, languages.name AS language, percent

FROM cities

LEFT JOIN countries

ON cities.country\_code = countries.code

LEFT JOIN languages

ON countries.code = languages.code

ORDER BY city, language;

\*/

SELECT cities.name AS city, urbanarea\_pop, countries.name AS country,

indep\_year, languages.name AS language, percent

FROM languages

RIGHT JOIN countries

ON languages.code = countries.code

RIGHT JOIN cities

ON countries.code = cities.country\_code

ORDER BY city, language;

### Full join

SELECT name AS country, code, region, basic\_unit

FROM countries

FULL JOIN currencies

USING (code)

WHERE region = 'North America' OR region IS NULL

ORDER BY region;

### Full join (2)

SELECT countries.name, code, languages.name AS language

FROM languages

FULL JOIN countries

USING (CODE)

WHERE countries.name LIKE 'V%' OR countries.name IS NULL

ORDER BY countries.name;

### Full join (3)

SELECT country.name AS country, region, language.name AS language,

cur.basic\_unit, cur.frac\_unit

FROM countries AS country

FULL JOIN languages AS language

USING (code)

FULL JOIN currencies AS cur

USING (code)

WHERE region LIKE 'M%esia';

### A table of two cities

1

SELECT c.name AS city, l.name AS language

FROM cities AS c

CROSS JOIN languages AS l

WHERE c.name LIKE 'Hyder%';

2

SELECT c.name AS city, l.name AS language

FROM cities AS c

INNER JOIN languages AS l

ON c.country\_code = l.code

WHERE c.name LIKE 'Hyder%';

### Outer challenge

SELECT co.name AS country, region, p.life\_expectancy AS life\_exp

FROM countries AS co

LEFT JOIN populations AS p

ON co.code = p.country\_code

WHERE p.year = 2010

ORDER BY p.life\_expectancy

LIMIT 5;

Chapter 3 Set theory clauses

### Union

SELECT \*

FROM economies2010

UNION

SELECT \*

FROM economies2015

ORDER BY code, year;

### Union(2)

SELECT country\_code

FROM cities

UNION

SELECT code

FROM currencies

ORDER BY country\_code;

### Union All

SELECT code, year

FROM economies

UNION ALL

SELECT country\_code, year

FROM populations

ORDER BY code, year;

### Intersect

SELECT code, year

FROM economies

INTERSECT

SELECT country\_code, year

FROM populations

ORDER BY code, year;

### Intersect (2)

SELECT co.name

FROM countries AS co

INTERSECT

SELECT ci.name

FROM cities AS ci;

### Except

SELECT city.name

FROM cities AS city

EXCEPT

SELECT country.capital

FROM countries AS country

ORDER BY name;

### Except (2)

SELECT country.capital

FROM countries AS country

EXCEPT

SELECT city.name

FROM cities AS city

ORDER BY capital;

### Semi-join

1

SELECT country.code

FROM countries AS country

WHERE country.region = 'Middle East';

2

SELECT DISTINCT lang.name

FROM languages AS lang

ORDER BY lang.name;

3

SELECT DISTINCT name

FROM languages

WHERE code IN

(SELECT code

FROM countries

WHERE region = 'Middle East')

ORDER BY name;

### Diagnosing problems using anti-join

1

SELECT count(name)

-- From countries

FROM countries

-- Where continent is Oceania

WHERE continent = 'Oceania';

2

-- 5. Select fields (with aliases)

SELECT c1.code, c1.name, c2.basic\_unit as currency

-- 1. From countries (alias as c1)

FROM countries as c1

-- 2. Join with currencies (alias as c2)

INNER JOIN currencies as c2

-- 3. Match on code

ON c1.code = c2.code

-- 4. Where continent is Oceania

WHERE continent = 'Oceania';

3

SELECT code, name

FROM countries

WHERE continent = 'Oceania'

AND code NOT IN

(SELECT code

FROM currencies);

### Set theory challenge

-- 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

);

Chapter 4 Subqueries

### Subqueries inside where

1

SELECT avg(life\_expectancy)

FROM populations

WHERE year = 2015;

2

-- Select fields

SELECT \*

-- From populations

FROM populations

-- Where life\_expectancy is greater than

WHERE life\_expectancy >

-- 1.15 \* subquery

1.15 \* (SELECT AVG(life\_expectancy)

FROM populations

WHERE year = 2015) AND

year = 2015;

### Subquery inside where (2)

SELECT name, country\_code, urbanarea\_pop

FROM cities

WHERE name IN

(SELECT capital

FROM countries)

ORDER BY urbanarea\_pop DESC;

### Subquery inside select

1

SELECT countries.name AS country, COUNT(\*) AS cities\_num

FROM cities

INNER JOIN countries

ON countries.code = cities.country\_code

GROUP BY country

ORDER BY cities\_num DESC, country

LIMIT 9;

2

SELECT countries.name AS country, COUNT(\*) AS cities\_num

FROM cities

INNER JOIN countries

ON countries.code = cities.country\_code

GROUP BY country

ORDER BY cities\_num DESC, country

LIMIT 9;

### Subquery inside from

1

SELECT code, count(name) AS lang\_num

FROM languages

GROUP BY code;

2

SELECT local\_name, subquery.lang\_num

FROM countries,

(SELECT code, count(name) as lang\_num

FROM languages

GROUP BY code) AS subquery

WHERE countries.code = subquery.code

ORDER BY lang\_num DESC

### Advanced subquery

1

SELECT name, continent, inflation\_rate

FROM countries

INNER JOIN economies

USING (code)

WHERE year = 2015;

2

SELECT max(inflation\_rate) as max\_inf

FROM (

SELECT name, continent, inflation\_rate

FROM countries

INNER JOIN economies

USING (code)

WHERE year = 2015) AS subquery

GROUP BY continent;

3

-- Select fields

SELECT name, continent, inflation\_rate

-- From countries

FROM countries

-- Join to economies

INNER JOIN economies

-- Match on code

ON countries.code = economies.code

-- Where year is 2015

WHERE year = 2015

-- And inflation rate in subquery (alias as subquery)

AND inflation\_rate IN (

SELECT MAX(inflation\_rate) AS max\_inf

FROM (

SELECT name, continent, inflation\_rate

FROM countries

INNER JOIN economies

ON countries.code = economies.code

WHERE year = 2015) AS subquery

-- Group by continent

GROUP BY continent);

### Subquery challenge

SELECT code, inflation\_rate, unemployment\_rate

FROM economies

WHERE year = 2015 AND code NOT IN

(SELECT code

FROM countries

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

ORDER BY inflation\_rate;

### Final challenge

SELECT DISTINCT c.name, e.total\_investment, e.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 year = 2015 AND region = 'Central America'

-- Order by field

ORDER BY name;

### Final challenge(2)

SELECT c.region, c.continent, AVG(p.fertility\_rate) AS avg\_fert\_rate

-- From left table

FROM populations AS p

-- Join to right table

INNER JOIN countries AS c

-- Match on join condition

ON p.country\_code = c.code

-- Where specific records matching some condition

WHERE year = 2015

-- Group appropriately

GROUP BY c.continent, c.region

-- Order appropriately

ORDER BY avg\_fert\_rate;

### Final challenge(3)

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;