# Module 2-2

**Aggregate Functions** 

# Additional SELECT options

#### **Data Concatenation**

Several columns can be concatenated into a single derive column using || .

Consider the following example:

SELECT name || ' is a country in ' || continent || ' with a population of ' || population AS sentence FROM country;

- The first three rows of output:
  - \* sentence
  - 1 Afghanistan is a country in Asia with a population of 22720000
  - 2 Netherlands is a country in Europe with a population of 15864000.
  - 3 Netherlands Antilles is a country in North America with a population of 217000

#### **Absolute Value**

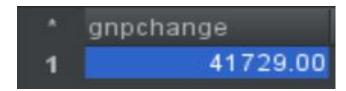
The absolute value can be calculated by using the ABS(...) function.

Consider the following example:

SELECT gnp - gnpold AS gnpchange FROM country;



SELECT ABS(gnp - gnpold) AS gnpchange FROM country;



Note that the results will never be negative now.

#### Limiting Results

You can limit the number of rows from your query with **LIMIT x**. For instance LIMIT 10 limits the results to 10 rows.

This tends to work best with ORDER BY as it allows you to construct lists like "top 10 of..."

#### Limiting Results Example

The following query gives you the "top 5" smallest countries by surface area:

SELECT name, surfacearea FROM country
ORDER BY surfacearea ASC LIMIT 5;

*	name	surfacearea
1	Holy See (Vatican City State)	0.4
2	Monaco	1.5
3	Gibraltar	6.0
4	Tokelau	12.0
5	Cocos (Keeling) Islands	14.0

# Sorting & Aggregating

# Sorting

In SQL, sorting is achieved through the ORDER BY statement:

#### ORDER BY [name of column] [direction]

- The ORDER BY section goes after the WHERE statement.
- You need to specify which column you want to sort by.
- You can optionally specify the direction of the sort:
  - ASC for ascending
  - DESC for descending.

#### Sorting Example

#### Consider the following example:

SELECT name, population FROM country ORDER BY population DESC;



Note that the records are now sorted in descending order with the largest population countries appearing first.

SELECT name, population FROM country ORDER BY population ASC;



Note that the records are now sorted in ascending order with the smallest population countries appearing first.

#### Sorting Example with Derived Fields

You can also sort by derived fields. Consider the following example:

SELECT name, population/surfacearea AS density
FROM country
ORDER BY density DESC

*	name	density
1	Macao	26277.77777777777
2	Monaco	22666.66666666668
3	Hong Kong	6308.837209302325
4	Singapore	5771.844660194175
5	Gibraltar	4166.66666666667

#### Sorting, default behavior

If no direction is provided (ASC or DESC), the sort order is **assumed to be ascending**.

SELECT name, population FROM country ORDER BY population **ASC**;

SELECT name, population FROM country ORDER BY population;



Let's write some SQL!

## Aggregate Functions

Aggregate data can be created by combining the value of one or more rows in a table. For example:

- The total population for North America.
- The total GNP for the whole world.
- The average surface area for all countries in Europe.
- The least populated country in Africa.

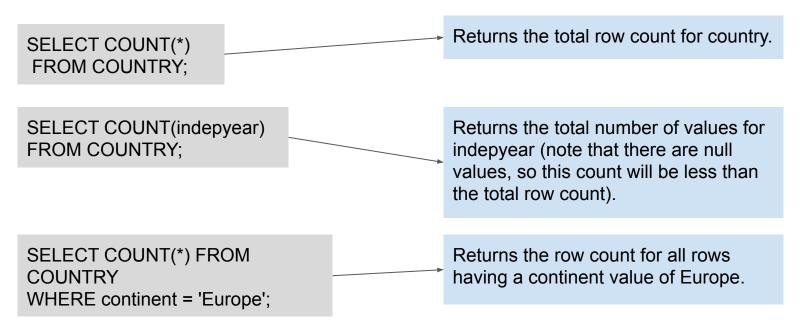
#### **Aggregate Functions**

Here are commonly used aggregate functions:

- **COUNT**: Provides the number of rows that meet a given criteria
- MAX / MIN: The maximum or minimum value of a column
- AVG: The average value of a column
- SUM: The sum of a column

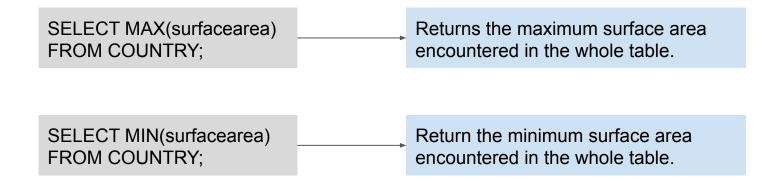
## Aggregate Functions: Count Example

The following are two examples for COUNT:



## Aggregate Functions: MAX/MIN example

Here we have examples of the MAX and MIN function:



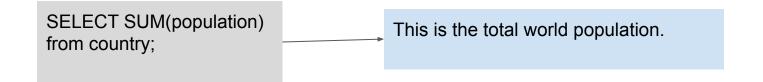
## Aggregate Functions: AVG example

The following is an example of AVG:



## Aggregate Functions: SUM example

The following is an example of SUM:



Let's write some SQL!

# Aggregate Functions: Group By

The previous examples illustrated how to apply the aggregate functions to the entire table, but what if we wanted to apply the aggregate functions only to subsets of the data?

In order to do this, we introduce the concept of aggregating (or grouping)
 which is achieved through the SQL command GROUP BY.

#### **GROUP BY [name of column]**

The GROUP BY section goes <u>before</u> the ORDER BY section.

# Aggregate Functions: Group By Example

Suppose you wanted to find out the sum of the population for each continent. Logically, if you did this manually you might have broken this process up into two steps:

- 1. Group all the rows into 5 groups, one for each continent.
- 2. For each group, sum up the population

You end up with 5 numbers, the population count for each of the five continents.

# Aggregate Functions: Group By Example

Just like how you would break up this process in two steps if done manually, SQL requires two elements to successfully aggregate this data:



 \* continent
 sum

 1 Asia
 3705025700

 2 South America
 345780000

 3 North America
 482993000

 4 Oceania
 30401150

 5 Antarctica
 0

 6 Africa
 784475000

 7 Europe
 730074600

This is equivalent to part 1, treat all rows with the same continent value as part of the same "bucket" of data or subset.

This is equivalent to part 2, adding up all the population values **only for a given subset** 

#### Aggregate Functions: A more complex example

You can combine multiple derived derived fields using different aggregate functions. Consider this example, where I want the <u>maximum GNP</u>, the <u>average</u> <u>population size</u>, and the <u>minimum surface area</u> of each continent:

SELECT continent,
MAX(gnp) AS "Max GNP",
AVG(population) AS "Average
Population",
MIN(surfacearea) AS "Smallest
Surface Area"
FROM country
GROUP BY continent

*	continent	Max GNP	Average Population	Minimum Surface Area
1	Asia	3787042.00	72647562.745098039216	18.0
2	South America	776739.00	24698571.428571428571	12173.0
3	North America	8510700.00	13053864.864864864865	53.0
4	Oceania	351182.00	1085755.357142857143	12.0
5	Antarctica	0.00	0E-20	59.0
6	Africa	116729.00	13525431.034482758621	78.0
7	Europe	2133367.00	15871186.956521739130	0.4

Let's write some SQL!