**SQL Fundamentals**

Table of Contents

[Course 1: Introduction to SQL 2](#_Toc100612337)

[Chapter 1: Selecting Columns 2](#_Toc100612338)

[Chapter 2: Filtering Rows 2](#_Toc100612339)

[Chapter 3: Aggregate Functions 3](#_Toc100612340)

[Chapter 4: Sorting and Grouping 3](#_Toc100612341)

[Course 2: Joining Data in SQL 4](#_Toc100612342)

[Chapter 1: Introduction to Joins 4](#_Toc100612343)

[Chapter 2: Outer Joins and Cross Joins 6](#_Toc100612344)

[Chapter 3: Set Theory Clauses 6](#_Toc100612345)

[Chapter 4: Subqueries 6](#_Toc100612346)

# Course 1: Introduction to SQL

## Chapter 1: Selecting Columns

* Row=record; column=field
* Query = request data from a database table or multiple tables
* Select data using SELECT
* SELECT field

FROM table; (select single column)

* SELECT and FROM are SQL keywords which are not case sensitive
* Semicolon is optional
* To select multiple columns, separate them by commas
* SELECT field1,field2

FROM table; (select multiple columns)

* SELECT \*

FROM table; (select all columns)

* SELECT \*

FROM table

LIMIT x; (select all columns but limit to x rows/observations)

* SELECT DISTINCT field

FROM table; (selects unique values in field)

* SELECT COUNT (\*)

FROM table; (gives count of all numbers of rows ; how many observations?)

* COUNT returns the number of non-missing values!
* COUNT should have a () next to it
* You can use COUNT on a particular column :

SELECT COUNT (field)

FROM table;

* To find the number of distinct values in a particular column, combine COUNT and DISTINCT:

SELECT COUNT (DISTINCT field)

FROM table;

## Chapter 2: Filtering Rows

* SELECT field

FROM table

WHERE field = ‘text’;

Operations: =,>,<,<=,>=, <> (not equal) [also !=]

* Single quotes for text (using PostgreSQL)
* SELECT field

FROM table

WHERE condition 1

AND condition 2; (multiple conditions)

* SELECT field

FROM table

WHERE condition 1

OR condition 2; (at least one condition met)

* SELECT field

FROM table

WHERE (condition 1 AND condition 2)

AND (condition 3 OR condition 4); (make sure when combining to use parentheses)

* SELECT field

FROM table

WHERE field

BETWEEN x and y (filter range ; inclusive of both ends)

* SELECT field

FROM table

WHERE field IN (a,b,c,d); (filter based on many conditions OR)

* IS NULL is useful when combined with WHERE to figure out what data you're missing
* IS NOT NULL
* WHERE field IS NOT NULL / IS NULL not IS NULL/IS NOT NULL field
* LIKE and NOT LIKE to match based on Patterns of string
* % is a wildcard that matches zero one or many characters in text
* \_ is a wildcard that matches any single character

## Chapter 3: Aggregate Functions

* Aggregate functions to summarise data: MIN(), MAX(), AVG(), SUM()
* SELECT aggfunction(field)

FROM table;

* SELECT (arithmetic here) : could be \*,+,-,/
* NOTE: integer/integer gives back integer so be careful when you want exact : SELECT(4/3) returns 1 instead of 1.3. To fix this, use : SELECT(x/y) AS z; (now answer is returned as 1.3 and is named z)
* AS is aliasing (to avoid getting same name of column for different results that use same aggregate operation: example SELECT MAX(budget) , MAX(duration)
* SOLUTION:
* SELECT MAX(budget) AS max\_budget,

MAX(duration) AS max\_duration

FROM films;

* Dividing by 60 is different than dividing by 60.0

## Chapter 4: Sorting and Grouping

* To sort: use SORT BY

SELECT field

FROM table

ORDER BY field DESC; (desc for descending)

* Sorting by multiple columns:

SELECT field1,field2

FROM table

ORDER BY field2,field1

(Order of fields is important so this will sort in ascending order by field 2 and then ascending order by field1)

* To group a result by one or more column, we use GROUP BY ; often used with aggregate functions

SELECT field, COUNT(\*)

FROM table

GROUP BY field;

* SQL will return an error if you try to SELECT a field that is not in your GROUP BY clause without using it to calculate some kind of value about the entire group.
* ORDER BY always goes after GROUP BY
* aggregate functions can't be used in WHERE clauses ; we use HAVING instead

SELECT field1

FROM table

GROUP BY field1

HAVING COUNT(field2) > 10;

# Course 2: Joining Data in SQL

## Chapter 1: Introduction to Joins

* KEY fields are used to reference one table to another
* INNER JOIN : key is in both tables
* Example of INNER JOIN:

SELECT p1.country,p1.continent,prime\_minister,president

FROM prime\_ministers AS p1

INNER JOIN presidents AS p2

ON p1.country=p2.country;

* Multiple INNER JOINs:

-- 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 and year

ON c.code=e.code AND e.year=p.year;

* The USING keyword in SQL and how it can be used in joins: If the key field is the same in both tables, you can use USING instead of ON. We place that key field between parantheses, eg:

Diagram

Description automatically generated

* In an INNER JOIN, order of tables doesn’t matter (Maybe!)
* SELF JOINS: an INNER JOIN of a table with itself
* Self-joins are used to compare values in a field to other values of the same field from within the same table.
* != can also be presented as <>
* With two numeric fields A and B, the percentage growth from A to B can be calculated as (B−A)/A∗100.0.
* CASE WHEN can be used to slice a numerical var into categories:Text

  Description automatically generated with medium confidence
* Use INTO to save the result of the previous query into a table of its own

## Chapter 2: Outer Joins and Cross Joins

## Chapter 3: Set Theory Clauses

## Chapter 4: Subqueries