**INTRODUCTION TO SQL**

**Ch1: Relational databases**

Rows = ‘records’

Columns = ‘fields’

Table names and field names should be lowercase and not include spaces (use ‘\_’). If they do have a space, you need to enclose them in double quotes.

Field names should be singular

Field names should not share a name with the table name

There are different data types for strings, integers and floats, depending on the length.

DB schema: a db design with the tables included, their relationships, their fields and data types

**Ch2: Querying**

**Select fields from a table**

*SELECT name*

*FROM table;*

(It is best practice to end the query with a semi-colon to indicate that the query is complete)

*SELECT field1, field2*

*FROM table;*

*SELECT \**

*FROM table;*

**Select fields from a table and change their name**

*SELECT field1 AS my\_field, field2*

*FROM table;*

**Select unique values from a field**

*SELECT DISTINCT field1,*

*FROM table;*

**Select unique values from a combination of fields**

*SELECT DISTINCT field1, field*

*FROM table;*

**View**: a table that is the result of a saved SQL SELECT statement

When accessed, view automatically update in response to updates in the underlying data.

*CREATE VIEW view\_name AS*

*SELECT field2, field2*

*FROM table;*

Once a view is created, you can query it just as a normal table

*SELECT field2*

*FROM view\_name*

**SQL Flavors:** different SQL versions, dialects of the same language

* **PostgreSQL:** free, open-source, relational database system
  + Limit the number of records returned / return the first N records

*SELECT id, field1*

*FROM table*

*LIMIT 2;*

* **SQL Server:** has free and paid (T-SQL) versions, created by Microsoft
  + Limit the number of records returned / return the first N records

*SELECT id, field1*

*FROM table*

*TOP 2;*

**-- INTERMEDIATE SQL**

**-- Ch1: Selecting data**

**-- Return the number of records with a value in a field**

*SELECT COUNT(field1) AS count\_field1*

*FROM table;*

**Count multiple fields**

*SELECT COUNT(field1) AS count\_field1, COUNT(field2) AS count\_field2*

*FROM table;*

**-- Count the total number of records in a table (including missing values)**

*SELECT COUNT(\*) AS total\_records*

*FROM table;*

**-- Count the number of unique values of a field**

*SELECT COUNT(DISTINCT field2) AS count\_distinct\_field1*

*FROM table;*

**-- Order of execution:** SQL code is NOT processed in the order in which it is written.

-- First, the “FROM” statement, then “WHERE”, then the “SELECT” statement (before that the aliasing statements), then the “LIMIT” statement

**-- Debugging SQL code:** most common errors are misspelling, incorrect capitalization, and incorrect or missing punctuation, specially commas.

**-- SQL style:** formatting (new lines, capitalization, indentation) are NOT required, but there are style standards.

**-- Ch2: Filtering records**

**-- Filtering numbers**

*SELECT title*

*FROM films*

*WHERE release\_year > 2010;*

*SELECT title*

*FROM films*

*WHERE release\_year = 2010;*

*SELECT title*

*FROM films*

*WHERE release\_year <> 2010;*

**-- Filtering strings:** use single quotation marks

*SELECT title*

*FROM films*

*WHERE country = ‘Japan’;*

**-- Multiple criteria**

*SELECT \**

*FROM coats*

*WHERE color = ‘yellow’*

*OR color = ‘black;*

*SELECT \**

*FROM coats*

*WHERE color = ‘yellow’*

*AND length = ‘short’;*

*SELECT \**

*FROM coats*

*WHERE buttons*

*BETWEEN 1 AND 5;*

(between is INCLUSIVE)

*SELECT \**

*FROM coats*

*WHERE (color = ‘yellow’ OR color=’black’)*

*AND (length = ‘short’ OR length = ‘medium’);*

*Multiple OR conditions*

*SELECT \**

*FROM coats*

*WHERE color IN (‘red‘, ’black’, ‘blue’);*

**-- More string filtering techniques**

* **LIKE:** search for a pattern in a field
  + Wild card “%”: match zero, one or many characters

*SELECT name*

*FROM people*

*WHERE name LIKE ‘Jua%’*

Matches names like: Juan, Juan Bautista, Juan Alfredo. It is case sensitive.

* + Wild card “\_”: match a single character

*SELECT name*

*FROM people*

*WHERE name LIKE ‘Jua%’*

Matches names like: Juan, Juas

* **NOT LIKE:** match records that do not include a pattern

*SELECT name*

*FROM people*

*WHERE name LIKE ‘Juan%’*

**-- Find strings that end with ‘r’:**

*SELECT name*

*FROM people*

*WHERE name LIKE ‘%r’*

**-- Find strings that start with ‘B’:**

*SELECT name*

*FROM people*

*WHERE name LIKE 'B%'*

**-- Find strings that have ‘r’ as a second letter:**

*SELECT name*

*FROM people*

*WHERE name LIKE '\_r%'*

**-- Filter data that includes NULL values**

*SELECT name*

*FROM people*

*WHERE birthdate IS NULL;*

**-- Count missing values**

*SELECT COUNT(\*) AS no\_birthdates*

*FROM people*

*WHERE birthdate IS NULL;*

**-- Count non-missing values**

*SELECT COUNT(\*) AS count\_birthdates*

*FROM people*

*WHERE birthdate IS NOT NULL;*

**-- Ch3: Aggregate functions**

**Calculate an AVERAGE (only on numerical fields)**

*SELECT AVG(budget) AS average\_budget*

*FROM films;*

**Calculate a SUM (only on numerical fields)**

*SELECT SUM(budget) AS total\_budget*

*FROM films;*

**Calculate MIN (works with non-numerical data like strings or dates)**

*SELECT MIN(budget) AS min\_budget*

*FROM films;*

**Calculate MAX (works with non-numerical data like strings or dates)**

*SELECT MAX(budget) AS max\_budget*

*FROM films;*

**Round decimal numbers: ROUND(number to round, decimal places), default is 0**

*SELECT ROUND(AVG(budget), 2) AS max\_budget*

*FROM films;*

**Round numbers to the nearest unit/ten/hundred, etc.**

*SELECT ROUND(AVG(budget), -3) AS max\_budget*

*FROM films;*

**Using arithmetic**

*SELECT (1 + 2);*

*SELECT (1 - 2);*

*SELECT (1 \* 2);*

*SELECT (4 / 3);*

**This returns 1 (division without remainder). To get the precise result**

*SELECT (4.0 / 3.0);*

**Arithmetics add the records horizontally**

**Substract one field from another**

*SELECT (revenue – budget) AS profit*

*FROM films;*

**-- Ch3: Sorting and grouping**

**Sorting results according to a field (ascending order by default) (we don’t need to select the query we are sorting for)**

*SELECT title, budget*

*FROM films*

*ORDER BY budget;*

*SELECT title, budget*

*FROM films*

*ORDER BY budget ASC;*

**Sort in descending order**

*SELECT title, budget*

*FROM films*

*ORDER BY budget DESC;*

*SELECT title, budget*

*FROM films*

*WHERE budget IS NOT NULL*

*ORDER BY budget DESC;*

**Sort by multiple fields**

*SELECT title, budget*

*FROM films*

*ORDER BY budget DESC, title ASC;*

**Group by a single field (here, get amount of movies by certification type)**

*SELECT certification, COUNT(title) AS title\_count*

*FROM films*

*GROUP BY certification;*

**Group by returns an error if you try to select a field which is not present in the GROUP BY clause. This gives an error:**

*SELECT certification, title*

*FROM films*

*GROUP BY certification;*

**You would need to add an aggregate function around title**

**Group by multiple fields**

*SELECT certification, language, COUNT(title) AS title\_count*

*FROM films*

*GROUP BY certification, language;*

**Group by, make a calculation, and order the results**

*SELECT*

*certification,*

*COUNT(title) as title\_count*

*FROM films*

*GROUP BY certification*

*ORDER BY title\_count DESC;*

**Filtering grouped data.**

**We can’t filter aggregate functions with WHERE clauses. For example, this will NOT work:**

*SELECT release\_year, COUNT(title) AS title\_count*

*FROM films*

*GROUP BY release\_year*

*WHERE COUNT(title) > 10;*

**You need to use a HAVING clause**

*SELECT release\_year, COUNT(title) AS title\_count*

*FROM films*

*GROUP BY release\_year*

*HAVING COUNT(title) > 10;*

**A more complex example**

*SELECT*

*certification,*

*COUNT(title) AS title\_count*

*FROM films*

*WHERE certification*

*IN (‘G’, ‘PG’, ‘R’)*

*GROUP BY certification*

*HAVING COUNT(title) > 500*

*ORDER BY title\_count DESC*

*LIMIT 3;*

**-- JOINING DATA IN SQL**

**The join clause is usually written before the select clause, mainly because of aliasing.**

**INNER JOIN: returns records whose key is present in both tables**

*SELECT prime\_ministers.country, prime\_ministers.continent, prime\_minister, president*

*FROM prime\_ministers*

*INNER JOIN presidents*

*ON prime\_ministers.country = presidents.country;*

**When selecting fields that exist in both tables, you need to write table\_name.field\_name**

**You can make this easier by aliasing tables**

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

*FROM prime\_ministers AS p1*

*INNER JOIN presidents AS p2*

*ON p1.country = p2.country;*

**Also, when you join on identical column names, you can make it more succinct**

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

*FROM prime\_ministers AS p1*

*INNER JOIN presidents AS p2*

*USING(country);*

**When writing joins, many SQL users prefer to write the SELECT statement *after* writing the join code, in case the SELECT statement requires using table aliases.**

**Defining table relationships:**

**- One-to-may: author to books**

**- One-to-one: individual to fingerprints**

**- Many-to-many: countries to languages**

**MULTIPLE JOINS combined in a single query**

*SELECT \**

*FROM left\_table*

*INER JOIN right\_table*

*ON left\_table.id = right\_table.id*

*IINER JOIN another\_table*

*ON left\_table.id = another\_table.id*

**Joining on MULTIPLE FIELDS**

*SELECT \**

*FROM left\_table*

*INNER JOIN right\_table*

*ON left\_table.id = right\_table.id*

AND *left\_table.date = right\_table.date*