SQL notes

MySQL UI

- Start MySQL: MySQL Connections -> Create a new schema: apply, refresh -> select Table: import Wizard, refresh -> double-click the database -> start
- Save & Export MySQL: Save the script as a .sql file ; Export recordset as a CSV/Excel file
- Operations: SELECT … FROM … ; INSERT INTO … VALUES … ; DELETE FROM … WHERE ; UPDATE … SET … WHERE … ;

SQL queries

- SELECT/FROM: eg. SELECT p.product_id AS pid, p.aisle_id, p.department_id, aisle_id + department_id AS a_d_id FROM products p;
- WHERE: eg. SELECT … FROM … WHERE (d.department = 'snacks' AND a.aisle_id > 132) OR (dt != '2020-06-31' AND dt BETWEEN 1980 AND 1990);
- GROUP BY: eg. SELECT user_id, order_dow, COUNT(order_id) AS n_order, AVG(days) AS avg_days FROM orders o GROUP BY user_id, order_dow;
- ORDER BY: eg. SELECT order_id, user_id, order_number, days FROM orders o WHERE days < 30 ORDER BY days DESC, user_id ASC LIMIT 10;
- HAVING: when WHERE doesn't apply; eg. SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country HAVING COUNT(CustomerID) > 5;
- NULL: eg. SELECT address, year FROM records WHERE address IS NOT NULL AND year IS NULL;
- CAST: converts a value (of any type) into a specified datatype, eg. SELECT CAST(25.65 AS int)
- DISTINCT: eg. SELECT COUNT(DISTINCT city), MAX(income), MIN(children) FROM contact_details;
- IN: eg. SELECT * FROM airbnb_listings WHERE country IN ('USA', 'France');
- ALL: the condition will be true only if the operation is true for all values in the range, eg. SELECT ALL column_name(s) FROM table_name WHERE condition;
- IF: eg. SELECT OrderID, Quantity, IF(Quantity>10, "MORE", "LESS") FROM OrderDetails; => create categorical data
- CASE: eg. SELECT OrderID, Quantity, CASE WHEN Quantity > 30 THEN "1" WHEN Quantity = 30 THEN "2" ELSE "3" END FROM OrderDetails;
- LIKE: 'a%' starts with a; '%a' ends with a; '%a%' a anywhere; eg. SELECT * FROM airbnb_listings WHERE city LIKE 'j%' AND city NOT LIKE '%t';
- CONCAT: eg. SELECT CONCAT(first_name,' ', last_name) AS full_name FROM patients;
- LEN: SELECT first_name, LEN(first_name) AS length_of_name FROM patients;

- JOIN: INNER(default: only keep values contained in both), OUTER(keep all values), RIGHT(use values from the right), LEFT(use values from the left); eg.
    - SELECT p.product_name, a.aisle, d.department FROM products p
    - LEFT JOIN aisles a ON (p.aisle_id = a.aisle_id)
    - LEFT JOIN departments d ON (p.department_id = d.department_id)

Subqueries

- Def: A query nested inside another query or inside another subquery
- Single value: returns exactly one column and exactly one row. It can be used with comparison operators =, <, <=, >, or >=. => eg. SELECT name FROM city WHERE rating = (SELECT rating FROM city WHERE name = 'Paris');
- Multiple values: returns multiple columns or multiple rows. Such subqueries can be used with operators IN, EXISTS, ALL, or ANY
    - EXISTS: returns TRUE if the subquery returns one or more records, eg. WHERE EXISTS (…)
    - ANY: returns TRUE if any of the subquery values meet the condition, eg. WHERE column_name operator ANY (…)
    - ALL: returns TRUE if all of the subquery values meet the condition, eg. WHERE column_name operator ALL (…)
- Correlated subquery: refers to the tables introduced in the outer query which cannot be run independently from the outer query
- UNION: combines the results of two result sets and removes duplicates; UNION ALL doesn't remove duplicate rows
- INTERSECT: returns only rows that appear in both result sets.
- EXCEPT/MINUS: returns only the rows that appear in the first result set but do not appear in the second result set.
- Examples
    - SELECT name FROM city WHERE country_id IN (SELECT country_id FROM country WHERE population > 20000000);
    - SELECT Name FROM Suppliers WHERE EXISTS (SELECT Product FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);
    - SELECT ProductName FROM Products WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);
    - SELECT ProductName FROM Products WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);
    - SELECT name FROM cycling WHERE country = 'DE' UNION / UNION ALL SELECT name FROM skating WHERE country = 'DE';
    - SELECT name FROM cycling WHERE country = 'DE' INTERSECT SELECT name FROM skating WHERE country = 'DE';

- SELECT name FROM cycling WHERE country = 'DE' EXCEPT / MINUS SELECT name FROM skating WHERE country = 'DE';

Reminders & Tips

- Orders: SELECT … FROM … JOIN … ON … (then) WHERE … ; instead of … WHERE … (then) JOIN …
- Double select using subqueries: eg. SELECT (SELECT count(…) FROM … WHERE …), (SELECT count(…) FROM … WHERE …)
- "For each …" => use GROUP BY
- "First/Last … date" => use MAX/MIN(date)
- "Same columns not not in one of the table" => use subqueries
- 156 / 100 = 1 ; 156 / 100.0 = 1.56
- Q: Show patient_id, diagnosis from admissions. Find patients admitted multiple times for the same diagnosis.
  => sol: SELECT patient_id, diagnosis FROM admissions GROUP BY patient_id, diagnosis HAVING COUNT(*) > 1;
- Q: Show first name, last name and role of every person that is either patient or doctor
  => sol: SELECT first_name, last_name, 'Patient' AS role FROM patients UNION ALL SELECT first_name, last_name, 'Doctor' AS role FROM doctors
- Q: display the first name, last name and number of duplicate patients based on their first name and last name.
  => sol: SELECT first_name, last_name, COUNT(*) FROM patients GROUP BY first_name, last_name HAVING COUNT(*) > 1
- Q: Show patient_id, first_name, last_name from patients whose does not have any records in the admissions table
  => sol: SELECT p.patient_id, first_name, last_name FROM patients p WHERE p.patient_id not in (SELECT a.patient_id FROM admissions a)
- Q: Show the provinces that has more patients identified as 'M' than 'F'. Must only show full province_name
  => sol: SELECT pn.province_name FROM patients p JOIN province_names pn ON (p.province_id = pn.province_id) GROUP BY pn.province_name => HAVING COUNT(CASE WHEN gender = 'M' THEN 1 END) > COUNT(CASE WHEN gender = 'F' THEN 1 END)
- Q: Show the percent of patients that have 'M' as their gender
  => sol: SELECT (SELECT COUNT(*) FROM patients WHERE gender = 'M') / CAST(COUNT(*) AS float) FROM patients

- Q: Sort the province names in ascending order in such a way that the province 'Ontario' is always on top.
  => sol: SELECT province_name FROM province_names ORDER BY (CASE WHEN province_name = 'Ontario' THEN 0 ELSE 1 END), province_name

Combined Example

- WITH agg_po AS (
    - SELECT po.product_id,
        - COUNT(po.add_to_cart_order) AS norder,
        - SUM(po.reordered) AS nreorder
    - FROM products_ordered po
    - LEFT JOIN orders o ON po.order_id=o.order_id
    - WHERE o.days_since_prior_order IS NOT NULL
    - GROUP BY po.product_id)
- SELECT apo.product_id, apo.norder, apo.nreorder, (apo.nreorder*1.0 / apo.norder) AS frac_reorder, p.product_name, p.aisle_id, p.department_id
- FROM agg_po as apo
- LEFT JOIN products p ON apo.product_id=p.product_id
- WHERE apo.nreorder > 10
- ORDER BY frac_reorder DESC;

Links

- W3School - SQL Tutoria : https://www.w3schools.com/sql/default.asp
- LearnSQL - SQL Basics Cheat Sheet : https://learnsql.com/blog/sql-basics-cheat-sheet/
- SQL Practice : https://www.sql-practice.com