SQL

JOINS :

(INNER) JOIN: Returns records that have matching values in both tables
LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

Join three tables:

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);

SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;

SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SELF JOIN:

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID AND A.City = B.City
ORDER BY A.City;
```

```
CREATE TABLE exercise_logs (id INTEGER PRIMARY KEY AUTOINCREMENT, type TEXT, minutes
INTEGER, calories INTEGER, heart_rate INTEGER);

INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("biking", 30, 115, 110);

SELECT type, SUM(calories) AS total_calories FROM exercise_logs GROUP BY type HAVING total_calories
> 150;

select author, sum(words) as total_words from books group by author HAVING sum(words) >
1000000;

/* CASE */
SELECT type, heart_rate,
    CASE
        WHEN heart_rate > 220-30 THEN "above max"
        WHEN heart_rate > ROUND(0.90 * (220-30)) THEN "above target"
        WHEN heart_rate > ROUND(0.50 * (220-30)) THEN "within target"
        ELSE "below target"
    END as "hr_zone" FROM exercise_logs;

SELECT COUNT(*),
    CASE
        WHEN heart_rate > 220-30 THEN "above max"
        WHEN heart_rate > ROUND(0.90 * (220-30)) THEN "above target"
        WHEN heart_rate > ROUND(0.50 * (220-30)) THEN "within target"
        ELSE "below target"
    END as "hr_zone"
FROM exercise_logs
GROUP BY hr_zone;
```

SQL UNION:
UNION Vs UNION ALL . (duplicates)
Equal columns, same datatypes, same order in both select statements.

```
SELECT columnname(s) from table1
union (ALL)
SELECT columnname(s) from table2
```

SELECT column_name(s) FROM table_name
WHERE condition GROUP BY column_name(s)
HAVING condition ORDER BY column_name(s);

```
UPDATE diary_logs SET content = "I had a horrible fight with OhNoesGuy" WHERE id = 1;

DELETE FROM diary_logs WHERE id = 1;
```

SQL QUERY PLANNING & OPTIMIZATION :

Declarative language -> query declares WHAT we want the sql engine to do.
But HOW to run this query is what makes the query efficient. This is called PLAN.