Joins:

- INNER: same as set intersection
- LEFT: all from left + intersection from right
- RIGHT: same as LEFT but the other way
- FULL OUTER: all

SELECT ¡cols¿
FROM ¡t1¿
INNER JOIN ¡t2¿
ON ¡¡condition¿¿;
SELECT ¡cols¿
FROM ¡t1¿
LEFT JOIN ¡t2¿
ON ¡¡condition¿¿;
SELECT ¡cols¿
FROM ¡t1¿
RIGHT JOIN ¡t2¿
ON ¡¡condition¿¿;
Left, right and full joins add null for cases without match.
SELECT ¡cols¿
FROM ¡t1¿
FULL OUTER JOIN ¡t2¿
ON ¡¡condition¿¿;
OUTER is redundant
FULL JOIN by UNION:
SELECT ¡cols¿
FROM ¡t1¿
LEFT JOIN ¡t2¿
ON ¡¡condition¿¿
UNION
SELECT ¡cols¿
FROM ¡t1¿
RIGHT JOIN ¡t2¿
ON ¡¡condition¿¿
SELF JOIN:
SELECT ¡cols¿
FROM ¡t1¿ ¡alias1¿, ¡t1¿ ¡alias2¿
WHERE ¡¡condition¿¿;
CROSS JOIN matches every left row with every right row, so if we had 20 left and 10 right, result will have 200 rows.
SELECT ¡cols¿ FROM ¡t1¿ CROSS JOIN ¡t2¿;
CROSS can be ommited;
same as:
SELECT ¡cols¿ FROM ¡t1¿, ¡t2¿;

EQUI joins use = comparison operator in where/or clause, NON EQUI joins use other comparison operators.

NATURAL JOIN:
SELECT * FROM ¡t1¿ NATURAL JOIN ¡t2¿;

columns with same name will appear only once.

Semi join - only match from l and r.
Anti join - only non match from l.

UNION, INTERSECT, MINUS(EXCEPT) work same way as in sets.

Joins append new columns, whereas unions append new rows.

Aggregate funcs(work on columns):

- MIN(), MAX()
- COUNT()
- AVG()
- SUM()

GROUP BY groups rows with same values in given column into "summary" row.

HAVING is kinda like where but can be used with Aggregate functions, eg. we can filter out summary row if it has ¡n rows.

SELECT ¡cols¿
FROM ¡t¿
WHERE ¡condition¿
GROUP BY ¡cols¿
HAVING ¡condition¿
ORDER BY ¡cols¿;

CASE statement is same as switch etc:
CASE
WHEN ¡cond1¿ THEN ¡res1¿
WHEN ¡cond2¿ THEN ¡res2¿
WHEN ¡cond3¿ THEN ¡res3¿
ELSE ¡default¿ END;

ELSE can be ommited and it will return NULL by default.

SUBQUERY can be used in SELECT, FROM, WHERE, etc; and can't be used in ORDER BY or GROUP BY.

Types of SUBQUERY:

- single row
- multi row
- multi col
- correlated - references >= 1 column(s) in outer statement
- selfcontained(non correlated) - independent of the outer query.

- nested

EXIST returns true if subquery returns at least one row.
SELECT ¡cols¿
FROM ¡t¿
WHERE EXISTS
(¡subquery¿);

ANY returns true if at least one row meets condition. ALL returns true if all rows meets condition.

SELECT ¡cols¿
FROM ¡t¿
WHERE ¡col¿ ¡comparison¿ ¡ANY — ALL¿
(¡subquery¿);

CTE - common table expression.

WITH
¡expr name¿ (¡col names¿)
AS
(definition),
¡expr2 name¿ (¡col names2¿)
AS
(def2)

CTE is temporary result that can be used in another statement(kinda like subquery)

Recursive cte:
WITH ¡cte name¿ (¡cols¿)
AS
(
¡anchor member¿
UNION ALL
¡recursive member¿
)

eg. WITH cte AS (SELECT 1 AS n UNION ALL SELECT n + 1 FROM cte WHERE n ¡ 50)