Aggregates & grouping

SQL Reference

Aggregate functions

An aggregate function turns many rows into one summary value. Wrap an average in `ROUND(x, 2)` to tidy it.

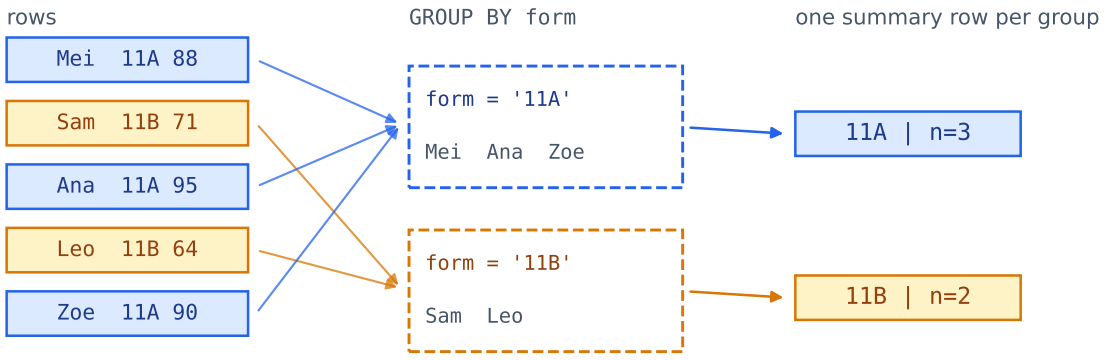
Function	Gives
`COUNT(*)`	how many rows
`SUM(col)`	the total
`AVG(col)`	the mean average
`MIN(col)` / `MAX(col)`	the smallest / largest value

```
CREATE TABLE student (id INTEGER, name TEXT, form TEXT, score INTEGER);
INSERT INTO student VALUES
↪  (1,'Mei','11A',88),(2,'Sam','11B',71),(3,'Ana','11A',95);
SELECT COUNT(*), ROUND(AVG(score), 2), MAX(score) FROM student;
```

GROUP BY and HAVING

`GROUP BY` makes one summary row per group. `HAVING` filters those groups – it is like `WHERE`, but it runs after grouping.

```
CREATE TABLE student (id INTEGER, name TEXT, form TEXT, score INTEGER);
INSERT INTO student VALUES
↪  (1,'Mei','11A',88),(2,'Sam','11B',71),(3,'Ana','11A',95);
SELECT form, COUNT(*) AS n, ROUND(AVG(score), 2) AS avg_score
FROM student
GROUP BY form
HAVING COUNT(*) > 1;
```



GROUP BY collects rows into one bucket per value; each bucket becomes one summary row

Write SQL clauses in their required syntax order. The table below is a logical model for reasoning about a query; the engine may optimise its physical execution order:

Step	Clause
1	`FROM` (and any `JOIN`)
2	`WHERE` – filter rows
3	`GROUP BY` – form groups
4	`HAVING` – filter groups
5	`SELECT` – compute the output columns
6	`ORDER BY`, then `LIMIT`

Common mistakes

- For predictable grouped results, group each selected non-aggregate column (or use a known functional dependency supported by the database). SQLite permits a bare column outside `GROUP BY`, but it may come from an arbitrary row in the group; a successful query does not guarantee the value you intended.
- Filter rows with `WHERE` (before grouping) and filter groups with `HAVING` (after).
- `COUNT(*)` counts rows; `COUNT(col)` skips NULLs in that column.