SQL cheat sheet

Skeleton

```
SELECT columns
FROM table
WHERE condition
GROUP BY column
HAVING condition
ORDER BY column DESC
LIMIT 10;
```

Always written in this order.

It runs in a different order

```
FROM → WHERE → GROUP BY
→ HAVING → SELECT
→ ORDER BY → LIMIT
```

That's why WHERE can't use a COUNT or a SELECT alias.

Filter rows

```
age > 40 AND city = 'Salt Harbor'
job <> 'jeweler' OR age < 30
city IN ('Salt Harbor', 'San Ciro')
age BETWEEN 30 AND 40
name LIKE 'Ma%'
email IS NULL
```

Text in single quotes. % means "anything".

Summarize

```
COUNT(*)               -- rows
COUNT(email)           -- no NULLs
COUNT(DISTINCT city)   -- distinct
SUM(loot)  AVG(loot)
MIN(loot)  MAX(loot)
```

Group

```
SELECT city, SUM(loot)
FROM thefts
WHERE loot >= 500          -- rows
GROUP BY city
HAVING COUNT(*) >= 2;      -- groups
```

In SELECT, only GROUP BY columns and aggregates.

Subquery

```
SELECT name, total
FROM orders
WHERE total >
  (SELECT AVG(total) FROM orders);
```

The inner one runs first.

Dates

```
-- all of March, nights included
date >= '2024-03-01'
AND date < '2024-04-01'
```

Better than BETWEEN when the column stores the time.

Sort, no repeats

```
SELECT DISTINCT city FROM thefts;
ORDER BY loot DESC  -- highest first
LIMIT 3             -- goes last
```

Without ORDER BY, LIMIT returns arbitrary rows.

Join tables



LEFT JOIN
all of A, with B if any

`FROM a LEFT JOIN b ON a.id = b.a_id`



JOIN
only what matches

`FROM a JOIN b ON a.id = b.a_id`



LEFT JOIN + IS NULL
A with no match in B

`… WHERE b.a_id IS NULL`



FULL JOIN
everything, matched or not

`FROM a FULL JOIN b ON a.id = b.a_id`

Exam traps

= NULL finds nothing. Use IS NULL.

BETWEEN with dates and times. Loses the last day. Use >= start and < next day.

COUNT(*) with LEFT JOIN. Gives 1 instead of 0. Count a column from the other table.

NOT IN with a NULL in the list. Returns zero rows. Filter NULLs out of the subquery.

A WHERE on the right-hand table. Turns your LEFT JOIN into a plain JOIN.

WHERE with an aggregate. Error. It belongs in HAVING.