The imputation log

What do we write in the empty cells of `games`? Names _____

Superset → **SQL → SQL Lab**, database `Dades classe`, schema `public`, table `games`. You cannot clean an empty cell: you can only decide what to do with it. That decision is called **imputation**. Pairs, one types and one writes; swap at Task 3. Nothing is modified today: every query is a `SELECT`. Run **one statement at a time**: select it, then **Run**.

Reference · the markbook

Option	In the markbook	Fair when...
A constant	Write a 0.	You **know** what the blank means: the work was never handed in.
A central value	Write the class average (or the median).	Few blanks, and nothing special about who has them.
Leave it empty + flag	Write nothing, add a note “not assessed”.	You do not know, or most of the column is blank.

`COUNT(*)` counts rows · `COUNT(col)` counts the rows where `col` is **not** empty · `COALESCE(a, b)` returns the first argument that is not `NULL`: “`a`, and if it is empty, `b`”.

Task 1 · Measure the hole

```sql
SELECT COUNT(*)                 AS total_rows,
       COUNT(price)             AS with_value,
       COUNT(*) - COUNT(price)  AS missing,
       ROUND(100.0 * (COUNT(*) - COUNT(price)) / COUNT(*), 1) AS pct_missing
FROM games;
```

Repeat it for the other three columns. This is your **log**: two columns now, the last two as you finish each task.

Column	Missing	% of rows	Option (markbook row)	Why, in one sentence
`price`				
`metascore`				
`developer`				
`publisher`				

Task 2 · `price`: is empty really free? Evidence first

```sql
SELECT COUNT(*) FROM games WHERE price = 0;        -- how many games cost exactly 0?
SELECT MIN(price) FROM games;                      -- the cheapest price on record
SELECT game, owners FROM games
WHERE price IS NULL ORDER BY number LIMIT 10;      -- who are they?
```

Games that cost exactly 0 _____ Cheapest price _____ Two games you know in the list: _____

So what does an empty price mean in this table? _____

Now fill it, and keep a note in the margin: the **flag** remembers which values were empty.

```sql
SELECT game, COALESCE(price, 0) AS price, (price IS NULL) AS is_free
FROM games WHERE game IN ('Half-Life 2', 'Dota 2', 'Portal') ORDER BY game;
```

`Dota 2`: price _____ `is_free` _____ `Half-Life 2`: price _____ `is_free` _____

Without `is_free`, how would you tell a real 0 from a filled one? _____

Two averages of the same column:

```sql
SELECT ROUND(AVG(price)::numeric, 2)              AS avg_ignoring_null,
       ROUND(AVG(COALESCE(price, 0))::numeric, 2) AS avg_free_as_zero
FROM games;
```

`avg_ignoring_null` _____ `avg_free_as_zero` _____ By how many rows does each one divide? _____

Which one would you publish, and for which question? _____

Task 3 · `metascore`: try to fill it, then look at the damage

The mean score is 71.9. Fill the blanks with 72 and count.

```sql
SELECT COUNT(*) FROM games WHERE metascore = 72;                -- real 72s
SELECT COUNT(*) FROM games WHERE COALESCE(metascore, 72) = 72;  -- after filling
```

Real 72s _____ 72s after filling _____ What would a chart of the filled column look like? _____

Now check **who** is missing. Small games first:

```sql
SELECT (owners LIKE '0%')  AS under_20k_owners,
       COUNT(*)            AS games,
       COUNT(metascore)    AS with_score,
       ROUND(100.0 * COUNT(metascore) / COUNT(*), 1) AS pct_with_score
FROM games GROUP BY 1;
```

Small games with a score _____ % The rest _____ % Are the blanks spread at random? _____

Why is 72 a lie for a game nobody reviewed? _____

The column you keep instead: `(metascore IS NOT NULL) AS has_metascore`.

Task 4 · `developer` and `publisher`: a label for the blank

A text column cannot take a mean. Give the blank its own category, so that `GROUP BY` shows it instead of hiding it.

```sql
SELECT COALESCE(developer, 'Unknown') AS developer, COUNT(*) AS games
FROM games GROUP BY 1 ORDER BY games DESC LIMIT 3;
```

Top row _____ with _____ games. Is that the biggest developer in the table?

Why is it a trap? _____

Done when Every blank is filled · the log of Task 1 has its four lines. **Fast pair?** The opposite problem: `average_playtime` is 0 for most games, because nobody measured it. `NULLIF(x, 0)` turns a 0 back into `NULL`: compare `AVG(average_playtime)` with `AVG(NULLIF(average_playtime, 0))`.

Stuck?

- **`pct_missing` shows 0**: integer division. Write `100.0 *`, with the decimal, before dividing.
- **`COALESCE types text and integer cannot be matched`**: the replacement must have the column's type.
- **`function round(double precision, integer) does not exist`**: add `::numeric` before rounding, as in Task 2.

At home In your own open dataset, pick the column with the most empty cells. Measure it (rows and %), choose one of the three options and write the `COALESCE`, or explain why you leave it empty.