Looks the same, is not

`owners_min` and the duplicate hunt on `games`

Names _____

Superset → **SQL → SQL Lab**, database `Dades classe`, schema `public`, table `games`, and your saved query `games_step1`. Two items that look identical and will not stack: something hidden makes them different. Pairs, one keyboard; swap who types when Part A is done. Run **one statement at a time**: select it, then **Run**.

Reference

Tool	What it does
`ASCII(c)`	The number of a character: its fingerprint.
`SPLIT_PART(x, sep, n)`	Cuts `x` at every `sep` and returns piece `n`.
`REGEXP_REPLACE(x, p, new, 'g')`	Replaces **every** match of the pattern `p`. `[^0-9]` means “anything that is not a digit”.
`SELECT DISTINCT *`	Keeps one copy of each fully identical row.
`ROW_NUMBER() OVER (PARTITION BY k ...)`	Numbers the rows 1, 2, 3… inside each group. The group `k` is the **key**: the columns that say “this is the same thing”.

Character numbers: 32 the space your keyboard types · 44 the comma · 46 the dot · 160 the **non-breaking space**, which looks exactly like a space. In a web page it is written ` ` and means “do not break the line here”.

Part A · From a text range to a number (16 min)

`owners` reads `10,000,000 .. 20,000,000`. You want the first number, as a real number. Cutting at `' .. '` did not work.

A1 · The fingerprint. Take the character just before the dots and ask for its number.

```
SELECT ASCII(SUBSTR(owners, POSITION('..' IN owners) - 1, 1)) AS mystery,
       ASCII(' ') AS normal_space
FROM games WHERE game = 'Half-Life 2';
```

`mystery` _____ `normal_space` _____ So why did `' .. '` never match? _____

A2 · Cut at what you can trust. The two dots are ordinary characters. Cut there and measure.

```
SELECT SPLIT_PART(owners, '..', 1) AS part,
       LENGTH(SPLIT_PART(owners, '..', 1)) AS len, LENGTH('10,000,000') AS expected
FROM games WHERE game = 'Half-Life 2';
```

`len` _____ `expected` _____ Where does the difference come from? _____

A3 · This one fails on purpose. Last week’s tools: `TRIM` the space, `REPLACE` the commas, cast.

```
SELECT REPLACE(TRIM(SPLIT_PART(owners, '..', 1)), ',', '')::bigint FROM games;
```

The error says: _____

Look inside the quotes, after the digits. What is still there, and why did `TRIM` not remove it? _____

A4 · Keep only the digits. Commas, spaces of any kind, whatever comes next year: all out.

```
SELECT REGEXP_REPLACE(SPLIT_PART(owners, '..', 1), '[^0-9]', '', 'g')::bigint AS owners_min,
       COUNT(*) AS n
FROM games GROUP BY owners_min ORDER BY owners_min;
```

Rows _____ Smallest _____ Biggest _____ Games with the smallest value _____
Does the smallest value mean that nobody owns those games? What does `owners_min` really measure?

_____

A5 · Add it to the clean query. Open `games_step1` (**SQL → Saved Queries**, click its name). In the `WITH` block, replace the plain `owners` by the A4 expression `AS owners_min`, and rename the block `step2`. Check with:

```
SELECT game, owners_min FROM step2 ORDER BY owners_min DESC LIMIT 4;
```

Save it as `games_step2`: **Save**, change the name, **Save as new** (**Update** would overwrite `games_step1`).

The game on top: _____ The next three: _____

Part B · The duplicate hunt (18 min)

74 names appear more than once in `games`. Before you delete anything, three cases. **True duplicate**: every column identical, the same row twice (a file loaded twice); remove the copies. **Apparent duplicate**: same name, another developer and date, like the three `Dark Matter`; keep them all. **Grey zone**: same name **and** developer, two dates, like `Penumbra Overture`; a decision, written down.

B1 · Does `games` have true duplicates?

```
SELECT COUNT(*) FROM games;
SELECT COUNT(*) FROM (SELECT DISTINCT * FROM games) AS t;
```

First count _____ Second count _____ True duplicates, yes or no? _____

B2 · The double upload. `UNION ALL` glues two results one under the other: pressing Upload twice, without touching the table.

```
WITH twice AS (SELECT * FROM games UNION ALL SELECT * FROM games)
SELECT COUNT(*) FROM twice;
```

Rows in `twice` _____. Keep the `WITH` line and count `SELECT DISTINCT * FROM twice`, wrapped as in B1. Rows now _____

B3 · The same with a key. Keeping only `WHERE rn = 1` would remove the copies.

```
WITH twice AS (SELECT * FROM games UNION ALL SELECT * FROM games),
numbered AS (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY game, release_date, developer ORDER BY number) AS rn
  FROM twice
)
SELECT rn, COUNT(*) FROM numbered GROUP BY rn ORDER BY rn;
```

Rows with `rn = 1` _____ Rows with `rn = 2` _____

B4 · The careless key. "A game is its name": use `PARTITION BY game` alone, on the **original** table (`FROM games`, no `twice`).

Rows with `rn = 1` _____ Rows a `WHERE rn = 1` would delete _____ One game you would lose: _____

B1 said there are no true duplicates. So what did this clean-up delete? _____

Done when Every blank is filled · `games_step2` is in your saved queries. **Fast pair?** Build `owners_max` from piece 2 of the split. In your own open dataset, is any column a true key?

Stuck?

- **`invalid input syntax for type bigint`** in A4, with a comma inside the quotes: the `'g'` is missing in `REGEXP_REPLACE`.
- **The result says `1k rows`**: that is the display limit. Wrap the query in `SELECT COUNT(*) FROM (...) AS t`.
- **`games_step1` is not in Saved Queries**: ask the teacher for its SQL.

At home In your own open dataset, which columns together identify one row? Prove it with a `GROUP BY ... HAVING COUNT(*) > 1` that returns nothing.