EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced SQL
Winter Term 2025/2026
Prof. Torsten Grust, Tim Fischer, Björn Bamberg
WSI — Database Systems Research Group

Assignment 4

Hand in this assignment until **Friday, November 21st 2025, 12:00 pm** at the latest.

> 🤔 Running out of ideas?
> Are you hitting a roadblock? Are some of the exercises unclear? Do you just need that one hint to get the ball rolling? Refer to the `#forum` 🌐 channel on our Discord server—maybe you'll find just the help you need.

Task 1: `WITH` or without

CTEs · Correlation · Unnesting

You can leverage CTEs to assemble complex queries from simple building blocks. But properly identifying these simple buildings blocks becomes complicated in the presence of correlation. Consider the two following queries `Q1` and `Q2` over the two tables `t(x int NOT NULL, y int NOT NULL)` and `p(val int NOT NULL)`:

Query `Q1`: uncorrelated subquery

```sql
SELECT t.x AS x
FROM   t AS t
WHERE  t.x IN
  (SELECT p.val
   FROM   p
   WHERE  p.val > 5);
```

Query `Q2`: correlated subquery

```sql
SELECT t.x AS x
FROM   t AS t
WHERE  t.x IN
  (SELECT p.val
   FROM   p
   WHERE  p.val > t.y);
```

A **Rewrite query `Q1` without subqueries** by introducing a CTE `subquery` that represents the *uncorrelated* subquery in the original query.

B **Rewrite query `Q2` without subqueries** by introducing a CTE `subquery` that represents the *correlated* subquery in the original query.

C Is rewriting one of the two queries *easier* than the other? **Explain briefly.**

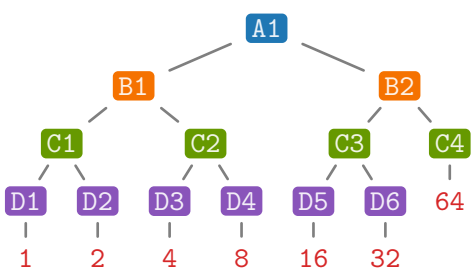
Task 2: Limited-depth trees and `GROUPING SETS`

Grouping Sets · Rollup

> ☝ What are "`GROUPING SETS`"?! And what is a "`ROLLUP`"!?
> Prof. Grust only briefly touched on this topic during the lecture, but it's too cool of a tool to pass up! In short, `GROUPING SETS` and `ROLLUP` allow you to perform multiple `GROUP BY`s in the same query. You can find more infos on these two features in chapter 2 of the lecture materials, that is on the slides 41 and 42, respectively, and in the code examples 🌐.

In this task we will look at **arbitrary** binary trees that are up to four layers deep. For clarity, we assign each level unique letters—in order of increasing depth, these are `A*`, `B*`, `C*` and `D*`. The leaf nodes of these trees hold integer labels. In the example below, the nodes `D1`, `D2`, `D3`, `D4`, `D5`, `D6` and `C4` are leaves, each holding of a power of two as a label.

In order to encode these trees in SQL, we use table `tree`—as seen below. Each row in table `tree` describes a leaf by its *root-to-leaf-path*, *i.e.*, it describes position of the leaves in the tree by listing out all of their individual ancestors.



`tree`

a	b	c	d	label
A1	B1	C1	D1	1
A1	B1	C1	D2	2
A1	B1	C2	D3	4
A1	B1	C2	D4	8
A1	B2	C3	D5	16
A1	B2	C3	D6	32
A1	B2	C4	NULL	64

A Write a SQL query using `GROUP BY ROLLUP` which produces a tree in which each node holds the *sum of all labels in its subtree*. Leaves keep their original values.

> 💡 Are you on the right track?
> For the example tree above, your query should produce the result to the right.

> ⚠️ Keep it simple!
> Do not use recursive CTEs, *i.e.*, `WITH RECURSIVE`, to solve this subtask! The way we have structured our table `tree`, makes it straightforward to write your query using `GROUP BY ROLLUP`.

a	b	c	d	sum
A1	B1	C1	D1	1
A1	B1	C1	D2	2
A1	B1	C1	NULL	3
A1	B1	C2	D3	4
A1	B1	C2	D4	8
A1	B1	C2	NULL	12
A1	B1	NULL	NULL	15
A1	B2	C3	D5	16
A1	B2	C3	D6	32
A1	B2	C3	NULL	48
A1	B2	C4	NULL	64
A1	B2	NULL	NULL	112
A1	NULL	NULL	NULL	127

B Write a SQL query similar to Subtask **A** to count the leaves below each node (the count of leaves below a leaf is 0).

> 💡 Want to verify your solution?
> For the example tree above, your query's output should contain the results show to the right.

a	b	c	d	cnt
…	…	…	…	…
A1	B2	C4	NULL	0
A1	B1	NULL	NULL	4
A1	NULL	NULL	NULL	7
…	…	…	…	…

C Now consider that the tree encoded by `tree` describes a tournament tree —the leaves represent contestants and the labels their skill levels. Matchups in the tournament are always won by the contestant with the higher skill level, which then proceeds onwards. On a tie, either contestant may proceed.

Write a SQL query similar to Subtask **A** to determine the winning skill levels for each competition. The tree root will hold the overall winning skill level.

> 💡 Is your tournament above board?
> For the tournament described by the example tree above, your query's output should contain the competition results show to the right.

a	b	c	d	lvl
…	…	…	…	…
A1	B1	C2	D3	4
A1	B1	C2	D4	8
A1	B1	C2	NULL	8
…	…	…	…	…