|  |  |
| --- | --- |
| |  | | --- | | **Interview Query Solution #33 | Over-Budget Projects** | |

|  |  |
| --- | --- |
| |  | | --- | | Here's the question and answer for yesterdays sql question.  This question was asked by: **Boston Consulting Group** | |

|  |  |
| --- | --- |
| |  | | --- | | employees projects  +---------------+---------+ +---------------+---------+  | id | **int** |<----+ +->| id | **int** |  | first\_name | varchar | | | | title | varchar |  | last\_name | varchar | | | | start\_date | date |  | salary | **int** | | | | end\_date | date |  | department\_id | **int** |--+ | | | budget | **int** |  +---------------+---------+ | | | +---------------+---------+  | | |  departments | | | employees\_projects  +---------------+---------+ | | | +---------------+---------+  | id | **int** |<-+ | +--| project\_id | **int** |  | name | varchar | +-----| employee\_id | **int** |  +---------------+---------+ +---------------+---------+    Over budget on a project is defined when the salaries, prorated to the day, exceed the budget of the project.  For example, if Alice and Bob both combined income make 200K and work on a project of a budget of 50K that takes half a year, then the project is over budget given 0.5 \* 200K = 100K > 50K.  Write a query to select all projects that are over budget. Assume that employees only work on one project at a time. | |

|  |  |
| --- | --- |
| |  | | --- | | **Solution:** | |

|  |  |
| --- | --- |
| |  | | --- | | Let's map out what the formula for constructing overbudget. In the question there was a simple example. If we define E as number of employees, PL as project length in days, S as in total salaries, and B as budget. Then our formula can be:  **PL / 365 \* (S\*E) > B**  To calculate this with SQL, we need this formula to apply on each row. This means we need each project to have it's own row, with each variable having it's own column like shown.  title | project\_days | budget | total\_salary  --------+---------------+-----------+-------------  tps | 100 | 50,000 | 100,000  reports | 125 | 200,00 | 250,000  ..  To get there we need to do join each table together with the corresponding information and then aggregate by title to get each project within it's own row.  SELECT  , title  -- days of project  , DATEDIFF(end\_date, start\_date, 'day') AS project\_days  , budget  -- coalesce turns null values to 0 in case of left joins  , SUM(COALESCE(salary,0)) AS total\_salary  FROM projects AS p  LEFT JOIN employees\_projects AS ep  ON p.id = ep.project\_id  LEFT JOIN employees AS e  ON e.id = ep.employee\_id  GROUP BY 1,2,3  Notice how we're left joining `projects` to both `employee\_projects` and `employees`. This is due to the effect that if there exists no employees on a project, we still need to define it as overbudget and setting the salaries as 0.  We're also grouping by title, project\_days, and budget, so that we can get the total sum. Given that each of title, project\_days, and budget are distinct for each project, we can do the group by without a fear of duplication in our SUM.  Finally, we can wrap the query into a sub-query and apply our equation to each project title with a CASE statement to tell if the project is over budget or not.  SELECT  title  , CASE WHEN  CAST(project\_days AS DECIMAL)/365 \* total\_salary > budget  THEN 'overbudget' ELSE 'within budget'  END AS project\_forecast  FROM (  SELECT  , title  , DATEDIFF(end\_date, start\_date) AS project\_days  , budget  , SUM(COALESCE(salary,0)) AS total\_salary  FROM projects AS p  LEFT JOIN employees\_projects AS ep  ON p.id = ep.project\_id  LEFT JOIN employees AS e  ON e.id = ep.employee\_id  GROUP BY 1,2,3  ) AS temp | |