Jenny starts with the departmental listing. This has employees listed within department.

She drills down through the employee name. The inferred ER structure relates this to the payroll records in the database and the project members in the spreadsheet.

She selects the payroll option.

The additional fields from the payroll record are added as columns. Note the name field from the payroll does not appear at this point as she has 'come from' the name already. Note also that the 'payroll' title and 'salary' column name are slightly lower to show they are drilled down from the department level.
Jenny drills down a second time through the name field. This time the payroll option is greyed out as this is already expanded.

She selects projects and against each person is listed the projects they are involved in. Note that there may be several project lines listed per person or none. Note also that the ‘projects’ title is at the same level as ‘payroll’ as they are both drilled down from department.
Instead, she decides to create a new column. She is selecting from the name in the department listing so gets a column that has one entry per person. Selecting ‘insert column’ on the project column would give one field per project.

It would be possible at this point to drill down through projects, getting a list of people for each project. This would make it possible to find all the people who are in the same project as someone. This would not be part of a normal universal relation.
The column is added at the end, but note that its column header is at the same level as 'department' and 'name'. Jenny is able to name the column and enter a formula for it. The formula ‘count(project)’ is interpreted like an SQL calculation if there were a ‘group by name’ clause.

The column is added and the formulae evaluated. Note that the second employee has two projects, so 'nos proj' is 2, but the last employee is in no project, so 'nos proj' is zero.
Jenny inserts a second column, also at the one per name level, and calls it ‘proj cost’. The formula uses the previous inserted column ‘nos proj’ and the ‘salary’ field from the payroll.

This field is the employees salary divided by the number of projects to give an approximate cost per project.

Note that one of the employees is in no projects, so has an undefined value for the formula, but this is to be expected.
Jenny pivots the table so that it is sorted by project. Notice now that there are three levels of headings, because payroll is drilled from department listing and department listing (one per person) is now drilled from projects.

Notice the 2nd and 4th row are the same employee listed under two different projects. See how the salary has been split between the projects.
### Projects

<table>
<thead>
<tr>
<th>name</th>
<th>total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>51464</td>
</tr>
</tbody>
</table>

Jenny inserts another column, this time at the project level (one field per project). She enters a formula summing the person by person costs over each project.

See how this gives a single cost for each project.

Note also the level of the ‘total cost’ column header and the dashed line to help see what columns are at what level. If the table were going to be left like this, Jenny might have moved this column over to the right of the project column to make it easier to read.
Jenny now hides all the columns apart from ‘project’ and ‘total cost’. The ‘null’ column in the full table no longer corresponds to any selected entity so is suppressed by default.

This is precisely the information Jenny requires for her boss!