Background

- Introduced by Codd (along with the Tuple Relational Calculus)

- Relational Algebra . . . :
  - Is procedural, like most programming languages
    — we need to supply an ordering of operations
  - Would not be a good replacement for SQL in a DBMS
  - Is a good introduction to the operators provided by SQL
Relations are *closed* under Relational Algebra operators

The eight basic Relational Algebra operators are:

The eight Relational Algebra operators can be grouped in two ways:
Select (σ, sigma) (1 / 2)

Example(s):

Select (σ, sigma) (2 / 2)

Notes:
Project ($\pi$, pi) (1 / 2)

Pronunciation: PRO-ject (not pro-JECT, not PRAH-ject)

---

Project ($\pi$, pi) (2 / 2)

Example(s):
Cartesian Product ($\times$) (1 / 2)

Example(s):

Cartesian Product ($\times$) (2 / 2)

Example(s):

What are the names of the active suppliers of nuts?
Union (∪) (1 / 2)

Example(s): 

Union (∪) (2 / 2)

Definition: Union Compatible

Example(s):
Difference (−) (1 / 2)

Example(s):

Difference (−) (2 / 2)
The Derived Operators

- Intersection (∩)
- Join (⋈)
- Division (÷)

Intersection (∩) (1 / 2)
Intersection (∩) (2 / 2)

Example(s):

Join (⊗) (1 / 3)

Background:
Join (⋈) (2 / 3)

Example(s):

Join (⋈) (3 / 3)

Three join variations:
Definition:

Contrasting Division and Cartesian Product:
Division (÷) (3 / 9)

What purpose does Division serve?

Example(s):

Recall:

<table>
<thead>
<tr>
<th>S</th>
<th>S#</th>
<th>Sname</th>
<th>Status</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P#</td>
<td>Pname</td>
<td>Color</td>
<td>Weight</td>
</tr>
<tr>
<td>SP</td>
<td>S#</td>
<td>P#</td>
<td>Qty</td>
<td></td>
</tr>
</tbody>
</table>

Division (÷) (4 / 9)

Example(s): (continued)
Example(s): (continued)

The content of $\alpha$ and $\beta$:

<table>
<thead>
<tr>
<th></th>
<th>S#</th>
<th>P#</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>S1</td>
<td>P1</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>P3</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>P5</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>P3</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>P6</td>
</tr>
<tr>
<td></td>
<td>S5</td>
<td>P1</td>
</tr>
<tr>
<td></td>
<td>S5</td>
<td>P2</td>
</tr>
<tr>
<td></td>
<td>S5</td>
<td>P3</td>
</tr>
<tr>
<td></td>
<td>S5</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>S5</td>
<td>P5</td>
</tr>
<tr>
<td></td>
<td>S5</td>
<td>P6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P#</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>P2</td>
</tr>
<tr>
<td></td>
<td>P3</td>
</tr>
</tbody>
</table>

Division ($\div$) (6 / 9)

Let’s examine the definition in detail:

$$\alpha \div \beta = \pi_{S#}(\alpha) - \pi_{S#}(\pi_{S#}(\alpha) \times \beta) - \alpha$$
### Division (\(\div\)) (7 / 9)

Looking at the data should help, too:

\[
\pi_{S#}(\alpha) \times \beta = \begin{array}{c|c} S# & P# \\ \hline S1 & P2 \\ S1 & P3 \\ S2 & P2 \\ S2 & P3 \\ S3 & P2 \\ S3 & P3 \\ S4 & P2 \\ S4 & P3 \\ S5 & P2 \\ S5 & P3 \\ \end{array} \quad \alpha = \begin{array}{c|c} S# & P# \\ \hline S1 & P1 \\ S2 & P3 \\ S2 & P5 \\ S3 & P3 \\ S3 & P4 \\ S4 & P6 \\ S5 & P1 \\ S5 & P2 \\ S5 & P3 \\ S5 & P4 \\ S5 & P5 \\ S5 & P6 \\ \end{array} \quad (\pi_{S#}(\alpha) \times \beta) - \alpha = \begin{array}{c|c} S# & P# \\ \hline S1 & P2 \\ S1 & P3 \\ S2 & P2 \\ S3 & P2 \\ S4 & P2 \\ S4 & P3 \\ S5 & P2 \\ \end{array}
\]

### Division (\(\div\)) (8 / 9)

And so, finally, we have our answer:

\[
\pi_{S#}(\alpha) = \begin{array}{c} S# \\ \hline S1 \\ S2 \\ S3 \\ S4 \\ S5 \\ \end{array} \quad \pi_{S#}((\pi_{S#}(\alpha) \times \beta) - \alpha) = \begin{array}{c} S# \\ \hline S1 \\ S2 \\ S3 \\ S4 \\ \end{array}
\]

\[
\alpha \div \beta = \pi_{S#}(\alpha) - \pi_{S#}((\pi_{S#}(\alpha) \times \beta) - \alpha) = \begin{array}{c} S# \\ \hline S5 \\ \end{array}
\]

Supplier S5 supplies all parts of weight 17 (P2 and P3).
Isn't there a short-cut to avoid that mess?