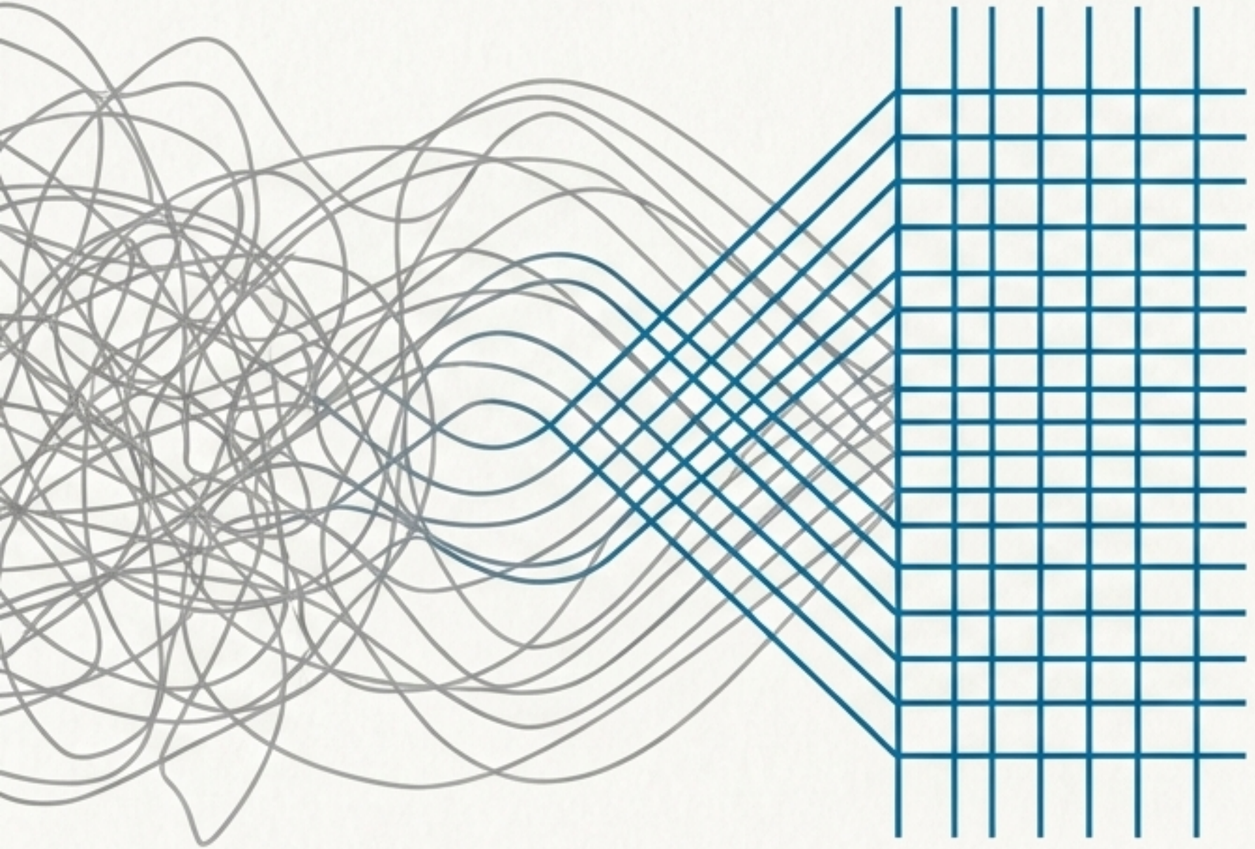
พีชคณิตเชิงสัมพันธ์: ภาษาแห่งการควบคุมข้อมูล

เปลี่ยนข้อมูลดิบให้กลายเป็นความเข้าใจเชิงลึก ด้วยเครื่องมือที่ทรงพลังที่สุดสำหรับฐานข้อมูล



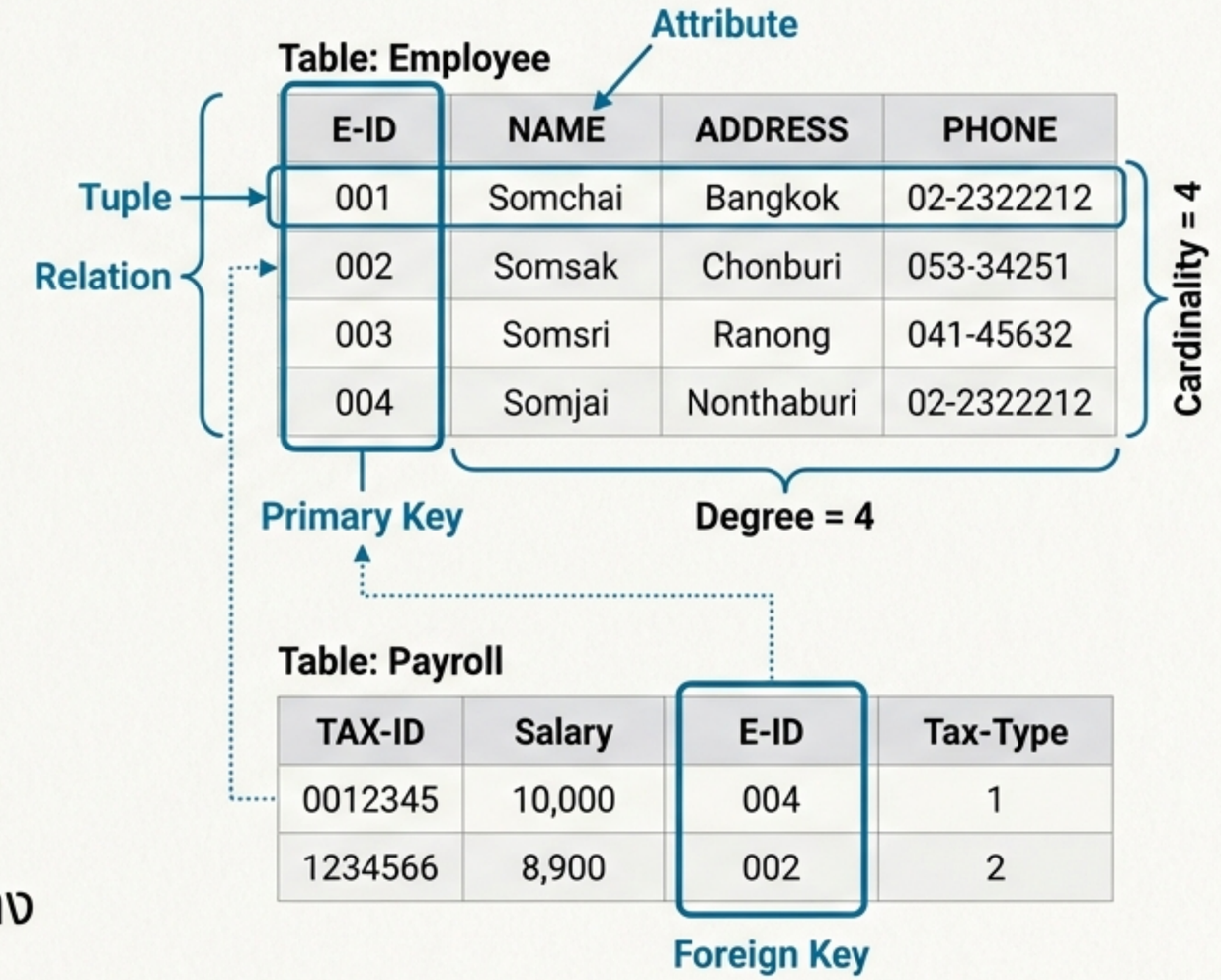
- ในโลกที่ข้อมูลมีอยู่มหาศาล การเข้าถึงและดึงข้อมูลที่ต้องการได้อย่างแม่นยำคือหัวใจสำคัญ
- ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) คือ ระบบจัดเก็บข้อมูลอย่างเป็นระเบียบในรูปแบบของตาราง (Relations)
- พีชคณิตเชิงสัมพันธ์ (Relational Algebra) คือ ภาษาเชิงกระบวนการ (Procedural Query Language) ที่เราใช้ในการสั่งการและจัดการข้อมูลเหล่านั้น เพื่อให้ได้ผลลัพธ์ตามที่ต้องการ
- ในบทเรียนนี้ เราจะมาทำความรู้จักกับ "ชุดเครื่องมือ" ทั้ง 8 ชนิดของ Relational Algebra ที่จะเปลี่ยนคุณให้เป็นผู้เชี่ยวชาญด้านข้อมูล

พื้นฐานสำคัญ: โครงสร้างของฐานข้อมูลเชิงสัมพันธ์

โครงสร้างข้อมูลหลักคือ **ตาราง (Relation)** ซึ่งเป็นตาราง 2 มิติ ประกอบด้วยแถวและคอลัมน์

คำศัพท์ที่ต้องรู้:

- **Relation (รีเลชั่น)**: คือ ตารางทั้งตาราง (เช่น ตาราง Employee)
- **Tuple (ทูเปิล)**: คือ แถวแต่ละแถวของข้อมูล (Row)
- **Attribute (แอตทริบิวต์)**: คือ คอลัมน์แต่ละคอลัมน์ (Column)
- **Cardinality**: จำนวนแถว (Tuples) ในตาราง
- **Degree**: จำนวนคอลัมน์ (Attributes) ในตาราง



เปิดกล่องเครื่องมือ: 8 คำสั่งพื้นฐานของ Relational Algebra

พีชคณิตเชิงสัมพันธ์ประกอบด้วย 8 โอเปอเรเตอร์พื้นฐาน ที่เปรียบเสมือนเครื่องมือสำหรับจัดการและสืบค้นข้อมูล เราจะแบ่งเครื่องมือเหล่านี้ออกเป็น 3 กลุ่มหลัก:

กลุ่มที่ 1: โอเปอเรชั่นแบบยูนารี (Unary Operations)

"เครื่องมือสำหรับจัดการข้อมูลภายในตารางเดียว"

- σ Select (σ): เลือกแถว
- ∏ Project (Π): เลือกคอลัมน์

กลุ่มที่ 2: โอเปอเรชั่นแบบเซต (Set Operations)

"เครื่องมือสำหรับจัดการข้อมูลระหว่างสองตาราง"

- ∪ Union (U): รวมข้อมูล (ไม่ซ้ำ)
- ∩ Intersection (∩): ข้อมูลที่ซ้ำกัน
- — Difference (-): ข้อมูลที่ต่างกัน
- × Product (×): การคูณข้อมูลทั้งหมด (จับคู่ทุกแถว)

กลุ่มที่ 3: โอเปอเรชั่นขั้นสูง (Advanced Operations)

"เครื่องมือสำหรับเชื่อมและหารข้อมูล"

- ⋈ Join (⋈): เชื่อมตารางตามเงื่อนไข
- ÷ Division (÷): การหารข้อมูล

เครื่องมือที่ 1: SELECT (σ) - เครื่องมือคัดกรองแถวข้อมูล

Function

ใช้สำหรับเลือก/แสดงข้อมูลเฉพาะ **แถว** (Tuples) ในตารางที่ตรงตามเงื่อนไข (Predicate) ที่กำหนด

Syntax

```
σ_เงื่อนไข(ชื่อตาราง)
```

- **σ** (Sigma) คือสัญลักษณ์ของ Select
- **เงื่อนไข** (Predicate) คือเงื่อนไขที่ใช้ในการเปรียบเทียบ เช่น `จังหวัด = ‘นครราชสีมา’` หรือ `ราคา < 1000`
- สามารถใช้ตัวดำเนินการเปรียบเทียบ (=, <, `>, <>) และตรรกะ (AND, `OR) ได้

Example

ต้องการข้อมูลของนักศึกษาที่อยู่จังหวัดนครราชสีมา

```
σ_จังหวัด='นครราชสีมา'(นักศึกษา)
```

Before & After

ตาราง 'นักศึกษา' (ก่อน)

รหัส	ชื่อ	จังหวัด
B001	แดง	นครราชสีมา
B002	ดำ	กรุงเทพฯ
B003	เขียว	สระบุรี
B004	ขาว	นครราชสีมา

ผลลัพธ์ (หลัง)

รหัส	ชื่อ	จังหวัด
B001	แดง	นครราชสีมา
B004	ขาว	นครราชสีมา

เครื่องมือที่ 2: PROJECT (Π) - เครื่องมือเลือกคอลัมน์

Function

ใช้สำหรับเลือก/แสดงเฉพาะ **คอลัมน์ (Attributes)** ที่ต้องการจากตาราง

Syntax

```
Π_ชื่อคอลัมน์1, ชื่อคอลัมน์2, ...(ชื่อตาราง)
```

- Π (Pi) คือสัญลักษณ์ของ Project
- ระบุชื่อคอลัมน์ที่ต้องการคั่นด้วยจุลภาค (,)

Example

ต้องการแสดงข้อมูลชื่อ, จังหวัด, และสาขาวิชาของนักศึกษาทุกคน

```
`Π_ชื่อ, จังหวัด, สาขาวิชา(นักศึกษา)`
```

Before & After

ตาราง 'นักศึกษา' (ก่อน)

รหัส	ชื่อ	จังหวัด	สาขาวิชา
B001	แดง	นครราชสีมา	โยธา
B002	ดำ	กรุงเทพฯ	โทรคมนาคม
B003	เขียว	สระบุรี	โยธา
B004	ขาว	นครราชสีมา	คอมพิวเตอร์

ผลลัพธ์ (หลัง)

ชื่อ	จังหวัด	สาขาวิชา
แดง	นครราชสีมา	โยธา
ดำ	กรุงเทพฯ	โทรคมนาคม
เขียว	สระบุรี	โยธา
ขาว	นครราชสีมา	คอมพิวเตอร์

การทำงานร่วมกัน: ใช้ SELECT และ PROJECT เพื่อคำตอบที่เฉพาะเจาะจง

เราสามารถนำโอเปอเรเตอร์ **σ** และ **Π** มาใช้ร่วมกันเพื่อสร้างคำสั่งที่ซับซ้อนขึ้นได้ โดยการนำผลลัพธ์จากโอเปอเรเตอร์หนึ่งไปเป็นข้อมูลสำหรับอีกโอเปอเรเตอร์หนึ่ง

Π_ชื่อคอลัมน์(**σ**_เงื่อนไข(ชื่อตาราง))

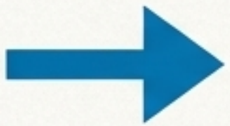
ต้องการแสดง **รหัส, ชื่อ, และสาขาวิชา** ของนักศึกษาที่มี **ชื่อว่า 'แดง'** เท่านั้น

Π_รหัส,ชื่อ,สาขาวิชา(σ_ชื่อ='แดง'(นักศึกษา))

ตารางเริ่มต้น

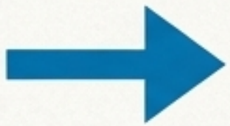
รหัส	ชื่อ	จังหวัด	สาขาวิชา
B001	แดง	นครราชสีมา	โยธา
B002	ดำ	กรุงเทพฯ	โทรคมนาคม
...			

ขั้นตอนที่ 1: ใช้ σ_ชื่อ='แดง'(...)



รหัส	ชื่อ	จังหวัด	สาขาวิชา
B001	แดง	นครราชสีมา	โยธา

ขั้นตอนที่ 2: ใช้ Π_รหัส,ชื่อ,สาขาวิชา(...)



ผลลัพธ์สุดท้าย

รหัส	ชื่อ	สาขาวิชา
B001	แดง	โยธา

กลุ่มที่ 2: โอเปอเรชั่นแบบเซต (Set Operations)

เครื่องมือสำหรับการรวม, เปรียบเทียบ, และค้นหาความสัมพันธ์ระหว่างข้อมูลสองชุด

- โอเปอเรชั่นกลุ่มนี้ทำงานโดยใช้หลักการของทฤษฎีเซตทางคณิตศาสตร์
- **ข้อกำหนดสำคัญ:** ตาราง (Relation) ทั้งสองที่จะนำมาดำเนินการต้องมีโครงสร้างเหมือนกัน (มีจำนวนและชนิดข้อมูลของคอลัมน์ตรงกันในลำดับเดียวกัน)

เครื่องมือในกลุ่มนี้ประกอบด้วย:

- **Union (∪):** การรวมข้อมูล
- **Intersection (∩):** การหาข้อมูลร่วม
- **Difference (-):** การหาข้อมูลที่แตกต่าง
- **Product (×):** การจับคู่ข้อมูลทั้งหมด (Cartesian Product)

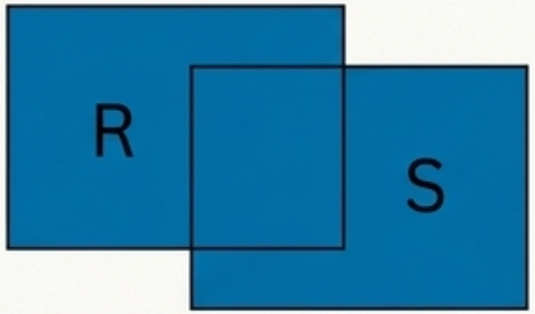
เครื่องมือเซตพื้นฐาน: Union (∪), Intersection (∩), และ Difference (-)

Union (∪)

รวมข้อมูลจาก 2 ตารางเข้าด้วยกัน
โดยข้อมูลที่ซ้ำกันจะแสดงเพียงแถว

A ∪ B

ตาราง A: {แดง, ดำ, เขียว}
ตาราง B: {แดง, ฝน}
A ∪ B: {แดง, ดำ, เขียว, ฝน}

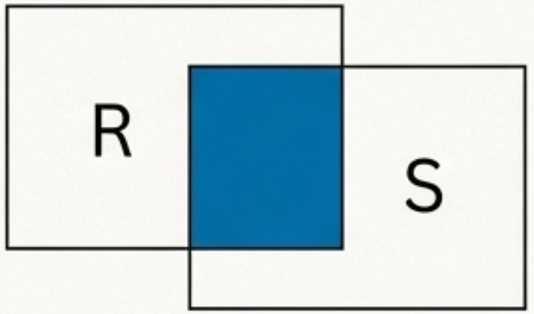


Intersection (∩)

เลือกเฉพาะข้อมูลที่ปรากฏ
เหมือนกัน ในทั้ง 2 ตาราง

A ∩ B

ตาราง A: {แดง, ดำ, เขียว}
ตาราง B: {แดง, ฝน}
A ∩ B: {แดง}

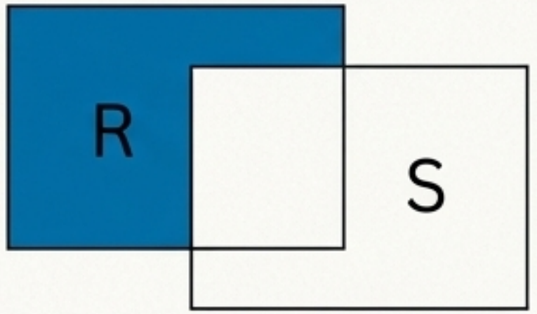


Difference (-)

แสดงข้อมูลที่อยู่ในตารางแรก
แต่ **ไม่อยู่** ในตารางที่สอง

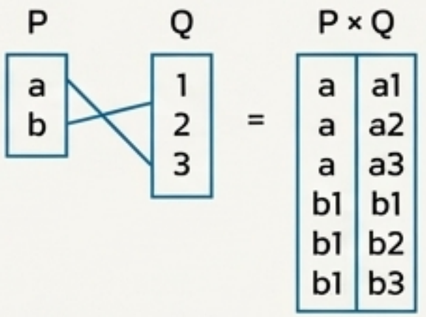
A – B

ตาราง A: {แดง, ดำ, เขียว}
ตาราง B: {แดง, ฝน}
A - B: {ดำ, เขียว}



เครื่องมือจับคู่: Product (×) หรือ Cartesian Product

Function สร้างตารางใหม่โดยการจับคู่ **ทุกแถว** จากตารางแรกกับ **ทุกแถว** ในตารางที่สอง
ผลลัพธ์ที่ได้จะมีจำนวนแถวเท่ากับ (จำนวนแถวตารางแรก × จำนวนแถวตารางที่สอง)



Syntax

```
R × S
```

Example จับคู่ตาราง ‘นักศึกษา’ กับตาราง ‘วิชา’

ตาราง ‘นักศึกษา’ (R)

รหัสนักศึกษา
B001
B002

ตาราง ‘วิชา’ (S)

รหัสวิชา
C001
C002
C003

ผลลัพธ์ ‘นักศึกษา × วิชา’ (R × S)

รหัสนักศึกษา	รหัสวิชา
B001	C001
B001	C002
B001	C003
B002	C001
B002	C002
B002	C003

กลุ่มที่ 3: JOIN (⋈) - เครื่องมือเชื่อมตารางอย่างชาญฉลาด

Function เป็นการรวมข้อมูลจาก 2 ตาราง คล้ายกับ Product แต่จะแสดงเฉพาะแถวที่ **ตรงตามเงื่อนไขการเชื่อมที่กำหนดเท่านั้น** ทำให้ผลลัพธ์มีความหมายและเกี่ยวข้องกันมากขึ้น

Syntax $R \bowtie_{\text{เงื่อนไข}} S$

Theta Join

การ Join โดยใช้เงื่อนไขเปรียบเทียบใดๆ (<, >, =, ≠).

Equijoin

การ Join โดยใช้เงื่อนไข **เท่ากับ (=)** เท่านั้น และยังแสดงคอลัมน์ที่ใช้เชื่อมซ้ำกัน.

Natural Join

การ Join แบบเท่ากับที่นิยมที่สุด โดยระบบจะหาคอลัมน์ที่มีชื่อเหมือนกันเพื่อใช้เป็นเงื่อนไขโดยอัตโนมัติ และ **ตัดคอลัมน์ที่ซ้ำซ้อนออก** เหลือเพียงคอลัมน์เดียว

Outer Join (Left, Right, Full)

การ Join ที่เก็บแถวที่ไม่สามารถจับคู่ได้ไว้ โดยแสดงค่า `NULL` ในส่วนที่ไม่มีข้อมูล

การทำงานของ Natural Join: Product + Select + Project

ตาราง สินค้า

รหัสสินค้า	ชื่อสินค้า	ราคา
รหัสสินค้า	ชื่อสินค้า	ราคา
111110	สนุด	120
222220	คอมพิวเตอร์	30000

ตาราง การสั่งซื้อ

เลขใบสั่ง	รหัสลูกค้า	รหัสสินค้า
เลขใบสั่ง	รหัสลูกค้า	รหัสสินค้า
1	C001	111110
2	C002	222220

ขั้นตอนที่ 1: Product (×) - จับคู่ทุกแถว

Cartesian Product: สร้างตาราง 4 แถว (2x2)

รหัสสินค้า	ชื่อสินค้า	ราคา	เลขใบสั่ง	รหัสลูกค้า	รหัสสินค้า (การสั่งซื้อ)
111110	สนุด	120	1	C001	111110
222220	คอมพิวเตอร์	30000	2	C002	222220
222220	สนุด	120	2	C002	222220

ขั้นตอนที่ 3: Project (Π) - ตัดคอลัมน์ `รหัสสินค้า` ที่ซ้ำซ้อนออกไป 1 คอลัมน์

รหัสสินค้า	ชื่อสินค้า	ราคา	เลขใบสั่ง	รหัสลูกค้า
111110	สนุด	120	1	C001
222220	คอมพิวเตอร์	30000	2	C002

Natural Join คือการรวม 3 ขั้นตอนที่เราเรียนมาแล้วเข้าด้วยกัน:

เชื่อมตาราง ‘สินค้า’ และ ‘การสั่งซื้อ’ ด้วย `รหัสสินค้า’

ขั้นตอนที่ 2: Select (σ) - เลือกเฉพาะแถว

เลือกเฉพาะแถวที่ `สินค้า.รหัสสินค้า = การสั่งซื้อ.รหัสสินค้า`

รหัสสินค้า	ชื่อสินค้า	ราคา	เลขใบสั่ง	รหัสลูกค้า	รหัสสินค้า (การสั่งซื้อ)
111110	สนุด	120	1	C001	111110
222220	คอมพิวเตอร์	30000	2	C002	222220

ขั้นตอนที่ 3: Project (Π) - ตัดคอลัมน์ `รหัสสินค้า`

รหัสสินค้า	ชื่อสินค้า	ราคา	เลขใบสั่ง	รหัสลูกค้า	รหัสสินค้า
111110	สนุด	120	1	C001	111110
222220	คอมพิวเตอร์	30000	2	C002	222220

ผลลัพธ์สุดท้าย (Final Result)

รหัสสินค้า	ชื่อสินค้า	ราคา	เลขใบสั่ง	รหัสลูกค้า
111110	สนุด	120	1	C001
222220	คอมพิวเตอร์	30000	2	C002

เครื่องมือขั้นสูง: Division (÷) - การค้นหาความสัมพันธ์ที่ครบถ้วน

- **Function:** ใช้สำหรับค้นหาค่าในคอลัมน์หนึ่ง ที่มีความสัมพันธ์ **ครบทุกค่า** กับอีกตารางหนึ่ง (ตัวหาร) มักใช้ตอบคำถามเช่น "ซัพพลายเออร์คนไหนที่ส่งสินค้าครบทุกชิ้น?"
- **Syntax**

```
R ÷ S
```

  - **R** (ตัวตั้ง) คือตารางที่มีข้อมูลความสัมพันธ์ (เช่น ใครซื้ออะไร)
  - **S** (ตัวหาร) คือตารางที่มีเซตของเงื่อนไขที่ต้องครบถ้วน (เช่น รายการสินค้าทั้งหมด)
- **Example:** ค้นหารหัสสินค้า (รหัสสินค้า) ที่ถูกขายโดยผู้ขาย (รหัสผู้ขาย) ทุกราย

ตาราง 'การขาย' (ตัวตั้ง - R)

รหัสสินค้า	รหัสผู้ขาย
P1	S1
P1	S2
P2	S1
P3	S2

÷

ตาราง 'ผู้ขาย' (ตัวหาร - S)

รหัสผู้ขาย
S1
S2

>

ผลลัพธ์ (R ÷ S)

รหัสสินค้า
P1

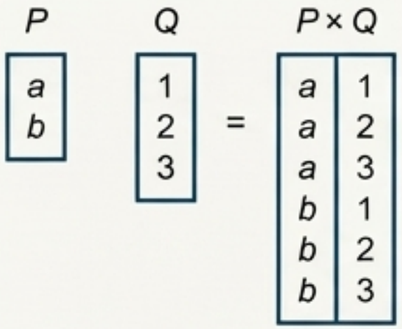
(เพราะ P1 เป็นสินค้าเดียวที่ถูกขายโดยทั้ง S1 และ S2)

สรุปชุดเครื่องมือ Relational Algebra ทั้งหมด

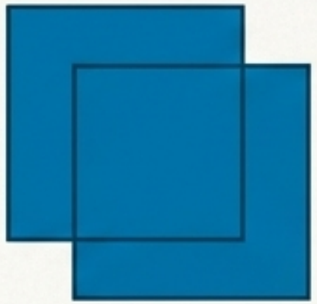
ภาพรวมของโอเปอเรเตอร์พื้นฐานสำหรับจัดการและสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์

Selection (σ)

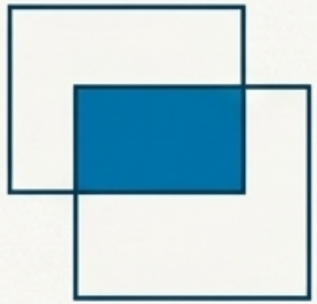
Projection (Π)



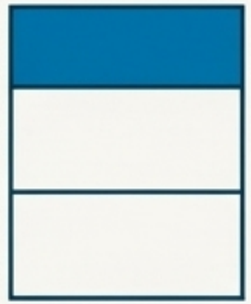
Cartesian product (×)



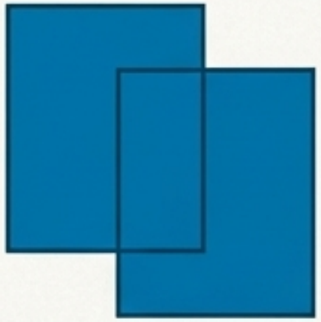
Union (∪)



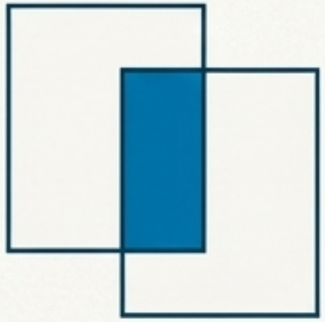
Intersection (∩)



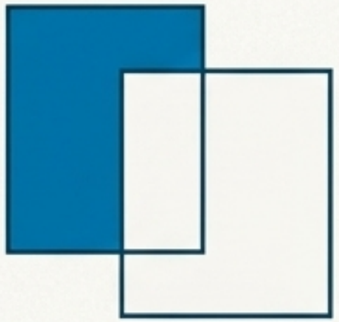
Set difference (-)



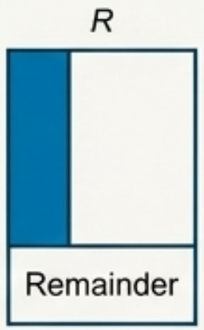
Natural join (⋈)

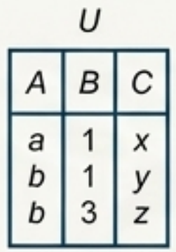


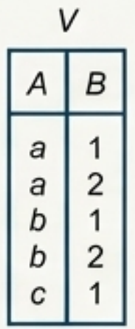
Intersection (∩)

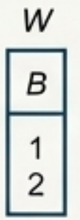


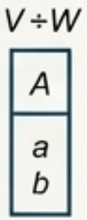
Set difference (-)











Division (÷)

Example of division

ถึงเวลาทดสอบเครื่องมือของคุณ: แบบฝึกหัด

จากตารางข้อมูลผู้เล่น (Player) ด้านล่าง จงใช้ความรู้เกี่ยวกับ Relational Algebra ที่ได้เรียนมา เขียนคำสั่งและหาผลลัพธ์สำหรับคำถามต่อไปนี้:

Table: Player

Name	position	age	height	weight
สุเมธ	กองหน้า	26	183	82
ผลดี	ปีกซ้าย	21	175	78
ชาติชาย	กองหลัง	30	169	71
ก้องเกียรติ	กองกลาง	24	180	78
เอกพจน์	ปีกขวา	20	180	78
สุชาติ	กองหน้า	19	179	80
พิเชษฐ์	ปีกซ้าย	22	177	76

โจทย์ (Challenge):

1. แสดงข้อมูลผู้เล่นที่มีอายุมากกว่า 23 ปี

```
σ_age > 23 (Player)
```

2. แสดงข้อมูลผู้เล่นที่มีความสูงตั้งแต่ 180 ขึ้นไป **หรือ** มีน้ำหนักน้อยกว่า 80

```
σ_height >= 180 OR weight < 80 (Player)
```

3. แสดง **ชื่อ** ของผู้เล่นที่เป็นกองหน้า **และ** มีอายุน้อยกว่า 25 ปี

```
Π_Name (σ_position = 'กองหน้า' AND age < 25 (Player))
```