IMEN891K / IDSC501 / AXGS-501 · DATA ENGINEERING · FALL 2026

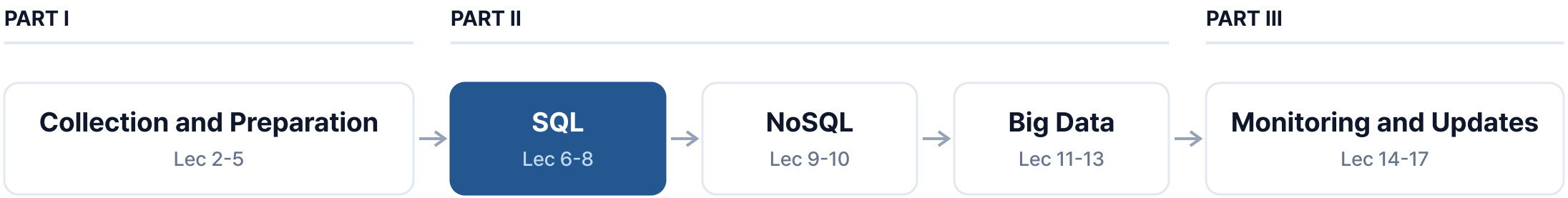
7.

SQL II: Joins

Sangwoo Mo · POSTECH

Industrial & Management Engineering

Where We Are


PART I
PART II
PART III
Collection and Preparation
Lec 2-5
SQL
Lec 6-8
NoSQL
Lec 9-10
Big Data
Lec 11-13
Monitoring and Updates
Lec 14-17

Outline: SQL II

01	**Match Rows**	return one row per matching pair
02	**Keep Unmatched Rows**	keep the left row when no match exists
03	**Test for Matches**	keep each outer row if a match exists
04	**Stack Results**	append compatible result rows

Recap: SFW on One Table

WHERE filters rows, SELECT chooses columns, and DISTINCT removes duplicate results

`WHERE` **selection**: keep the rows that meet a condition

`SELECT` **projection**: keep a subset of the columns

`DISTINCT` drop duplicate rows; a table is a **multiset**

```
SELECT DISTINCT Category
FROM   Product
WHERE  Price < 200
```

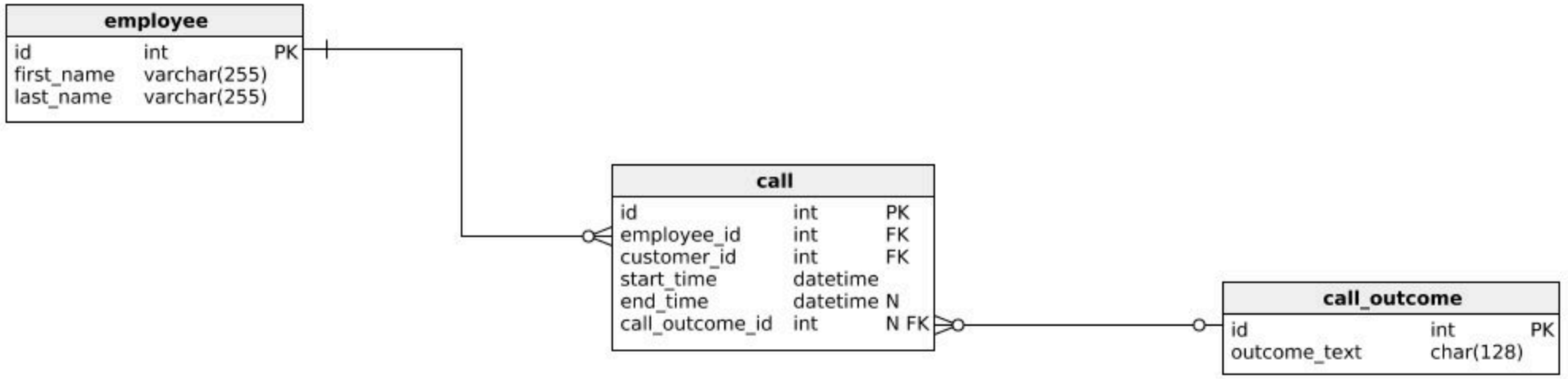
category
Gadgets
Photography

Two Gadgets rows below $200 collapse into **one** with DISTINCT

01 · Match Rows

01	**Match Rows**	return one row per matching pair
02	**Keep Unmatched Rows**	keep the left row when no match exists
03	**Test for Matches**	keep each outer row if a match exists
04	**Stack Results**	append compatible result rows

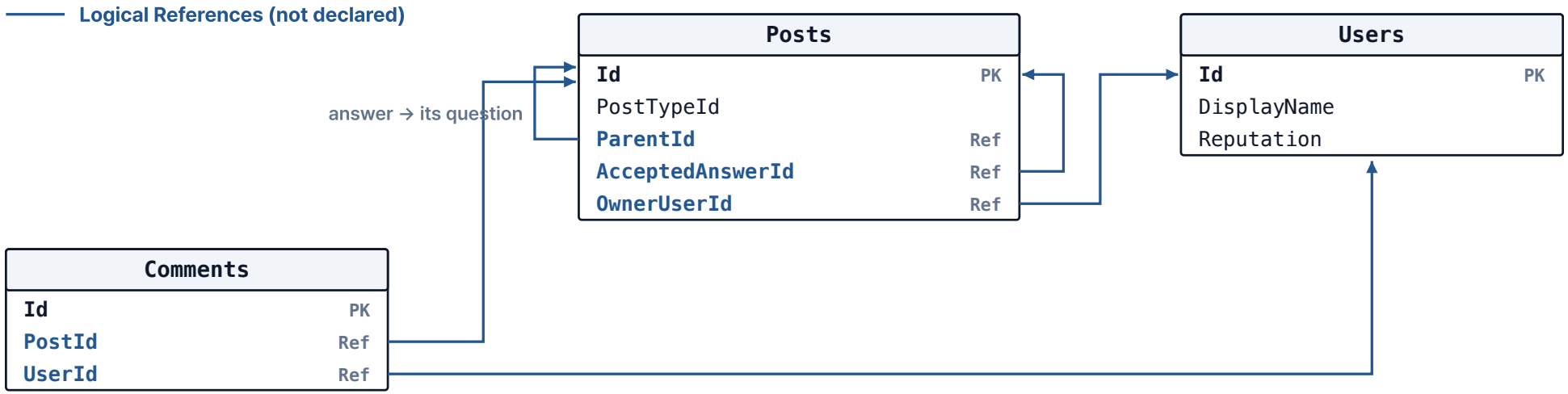
Multi-Table Queries



Foreign keys link employees, calls and outcomes; follow those keys when combining records

Case study: Stack Overflow

One question page is spread over several tables linked by Id columns



Answers reference questions; reconstructing a page combines **posts, users and comments**

Recap: Product and Company

pname	price	category	manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
PowerGizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

cname	stockprice	country
GizmoWorks	25.0	USA
Canon	65.0	Japan
Hitachi	15.0	Japan

Price belongs to Product; country belongs to Company; manufacturer links the two

Matching a Product to Its Company

Product

pname	manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks
SingleTouch	Canon
MultiTouch	Hitachi

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

manufacturer = cname

Match the highlighted key columns; two products point to the same GizmoWorks row

Table Aliases

```
SELECT P.pname,P.price,C.country
FROM Product AS P JOIN Company AS C
  ↑ P and C: short names for the tables
  ON P.manufacturer=C.cname;
```

JOIN on Related Keys

Product → manufacturer

pname	manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks
SingleTouch	Canon
MultiTouch	Hitachi

Company → country

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Matched pairs

pname	country
Gizmo	USA
PowerGizmo	USA
SingleTouch	Japan
MultiTouch	Japan

```
FROM Product P JOIN Company C ON P.manufacturer=C.cname
```

Each matched pair contributes one result row; GizmoWorks supplies USA twice

Every Pair Before the Match

Product × Company	GizmoWorks	Canon	Hitachi
Gizmo	✓	—	—
PowerGizmo	✓	—	—
SingleTouch	—	✓	—
MultiTouch	—	—	✓

4 products × 3 companies = **12 candidate pairs**

manufacturer = cname keeps **4 matching pairs**

Twelve candidate cells contain four valid manufacturer matches

Predict Join Cardinality

One company

cname
GizmoWorks

Two products

pname	manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks

Two result rows

cname	pname
GizmoWorks	Gizmo
GizmoWorks	PowerGizmo

One company matches two products, so the company appears in two different result pairs

Filter the Joined Rows

Matched products

pname	country
Gizmo	USA
PowerGizmo	USA
SingleTouch	Japan
MultiTouch	Japan

country = Japan

pname	price
SingleTouch	149.99
MultiTouch	203.99

Also price ≤ 200

pname	price
SingleTouch	149.99

Japan keeps two products; the price condition then removes MultiTouch

Read the Query in Logical Order

Step	Effect
FROM / JOIN / ON	Build matching pairs
WHERE	Keep the pairs that satisfy the condition
SELECT	Choose output columns
ORDER BY / LIMIT	Order and trim the result

Logical query order explains the result without prescribing the physical execution

Name the Output Columns

```
SELECT P.pname AS product,C.country AS
manufacturer_country
   ↑ name of the output column
FROM Product P JOIN Company C ON
P.manufacturer=C.cname;
```

product	manufacturer_country
Gizmo	USA
PowerGizmo	USA
SingleTouch	Japan
MultiTouch	Japan

Output aliases describe what each selected column means

Caveat: Rows Multiply at a Join

```
SELECT C.cname,P.pname
FROM Company C JOIN Product P ON
P.manufacturer=C.cname;
```

↑ one row per matching product

cname	pname
GizmoWorks	Gizmo
GizmoWorks	PowerGizmo
Canon	SingleTouch
Hitachi	MultiTouch

GizmoWorks appears twice because it matches two different products

Recap: The Join Input

pname	price	manufacturer
Gizmo	19.99	GizmoWorks
PowerGizmo	29.99	GizmoWorks
SingleTouch	149.99	Canon
MultiTouch	203.99	Hitachi

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

Two companies are in Japan; the price condition decides which products remain

Caveat: A Match Needs the Right Meaning

Decision

part	time
P-104	09:14

Inspections

part	time	result
P-104	08:30	defect
P-104	09:20	pass

This decision's outcome

part	time	result
P-104	09:20	pass

Matching only P-104 also attaches an inspection that happened before the decision

Demo: Match Products to Companies

1. Return names and prices of products made in Japan and priced at most 200
2. Remove the price limit and compare the rows

```
-- write your query here, then Run
```

QUERY RESULT

Write a query on the left, then Run.

Answer: Japanese Products

```
SELECT P.pname,P.price FROM Product P
JOIN Company C ON P.manufacturer=C.cname
WHERE C.country='Japan';
```

pname	price
SingleTouch	149.99
MultiTouch	203.99

Without the price limit, both SingleTouch and MultiTouch remain

02 · Keep Unmatched Rows

01	Match Rows	return one row per matching pair
02	**Keep Unmatched Rows**	keep the left row when no match exists
03	Test for Matches	keep each outer row if a match exists
04	Stack Results	append compatible result rows

A Company Without Products

cname	stockprice	country
GizmoWorks	25.0	USA
Canon	65.0	Japan
Hitachi	15.0	Japan
NewCo	0.0	Korea

pname	manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks
SingleTouch	Canon
MultiTouch	Hitachi

NewCo is a company with no matching product in this copy

The Unmatched Row

LEFT JOIN preserves the company even when no product matches

Company input

cname	country
NewCo	Korea

INNER JOIN

No product match

No output row

LEFT JOIN

cname	pname
NewCo	NULL

No match becomes NULL on the product side

INNER JOIN and LEFT JOIN

```
SELECT C.cname,P.pname FROM Company C
JOIN Product P ON P.manufacturer=C.cname;
```

cname	pname
GizmoWorks	Gizmo
GizmoWorks	PowerGizmo
Canon	SingleTouch
Hitachi	MultiTouch

```
SELECT C.cname,P.pname FROM Company C
LEFT JOIN Product P ON P.manufacturer=C.cname;
```

↑ companies with no product stay, with NULL

cname	pname
Canon	SingleTouch
GizmoWorks	Gizmo
GizmoWorks	PowerGizmo
Hitachi	MultiTouch
NewCo	NULL

LEFT JOIN adds NewCo with a NULL product; INNER JOIN has no row for it

Find an Unmatched Company

```
SELECT C.cname FROM Company C
LEFT JOIN Product P ON P.manufacturer=C.cname
WHERE P.pname IS NULL;
```

↑ no match: the product side is NULL

cname
NewCo

The right-side product key is NULL only for NewCo

Start from the Rows You Must Keep

```
SELECT C.cname,P.pname FROM Company C
LEFT JOIN Product P ON P.manufacturer=C.cname
ORDER BY C.cname,P.pname;
```

cname	pname
Canon	SingleTouch
GizmoWorks	Gizmo
GizmoWorks	PowerGizmo
Hitachi	MultiTouch
NewCo	NULL

The left input determines which unmatched rows survive

Caveat: ON and WHERE

```
SELECT C.cname,P.pname FROM Company C
LEFT JOIN Product P ON P.manufacturer=C.cname AND
P.price>100;
```

↑ in ON: every company is kept

cname	pname
Canon	SingleTouch
GizmoWorks	NULL
Hitachi	MultiTouch
NewCo	NULL

```
SELECT C.cname,P.pname FROM Company C
LEFT JOIN Product P ON P.manufacturer=C.cname
WHERE P.price>100;
```

↑ in WHERE: NULL rows are dropped

cname	pname
Canon	SingleTouch
Hitachi	MultiTouch

ON limits matches while keeping companies; WHERE removes rows after the join

Recap: The Unmatched Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan
NewCo	Korea

Keep all four companies, including NewCo with no product match

Demo: Keep Companies Without Products

1. List every company with its product names, including companies with no products
2. Keep only companies with no matching product

```
-- write your query here, then Run
```

QUERY RESULT

Write a query on the left, then Run.

Answer: Companies Without Products

```
SELECT C.cname FROM Company C
LEFT JOIN Product P ON P.manufacturer=C.cname
WHERE P.pname IS NULL;
```

cname
NewCo

NewCo exists in Company but has no matching product

03 · Test for Matches

01	Match Rows	return one row per matching pair
02	Keep Unmatched Rows	keep the left row when no match exists
03	**Test for Matches**	keep each outer row if a match exists
04	Stack Results	append compatible result rows

Naming a Query Result

```
WITH japanese_products AS (
  ↑ a name for this result
  SELECT P.pname,P.price FROM Product P
  JOIN Company C ON P.manufacturer=C.cname
  WHERE C.country='Japan'
)
SELECT * FROM japanese_products;
```

pname	price
SingleTouch	149.99
MultiTouch	203.99

WITH gives a query result a name that another query can read

Pairs or One Company

Two related product rows

manufacturer	pname
GizmoWorks	Gizmo
GizmoWorks	PowerGizmo

JOIN: matching pairs

cname
GizmoWorks
GizmoWorks

EXISTS: keep the company

cname
GizmoWorks

Two product matches can answer one company-level question

EXISTS and Query Blocks

```
SELECT C.cname FROM Company C
WHERE EXISTS (
  SELECT 1 FROM Product P
  WHERE P.manufacturer=C.cname AND P.category='Gadgets'
);
```

true if the inner query returns a row

Inspect the Inner Result First

```
SELECT manufacturer FROM Product WHERE
category='Gadgets';
```

manufacturer
GizmoWorks
GizmoWorks

Repeated candidate values do not repeat the outer company

IN Reads a Query Result

```
SELECT cname FROM Company
WHERE cname IN (SELECT manufacturer FROM Product
  ↑ the list comes from a query
  WHERE category='Gadgets');
```

cname
GizmoWorks

The inner query supplies candidate manufacturer names

JOIN or EXISTS

```sql
SELECT C.cname FROM Company C JOIN Product P
ON P.manufacturer=C.cname WHERE
P.category='Gadgets';
```

cname
GizmoWorks
GizmoWorks

```sql
SELECT C.cname FROM Company C WHERE EXISTS (
  SELECT 1 FROM Product P
  WHERE P.manufacturer=C.cname AND
P.category='Gadgets');
```

cname
GizmoWorks

JOIN returns two Gadget matches; EXISTS keeps one company row

NOT EXISTS

```
SELECT C.cname FROM Company C WHERE NOT EXISTS (
  ↑ keep companies with no such product
  SELECT 1 FROM Product P
  WHERE P.manufacturer=C.cname AND
P.category='Gadgets');
```

cname
Canon
Hitachi
NewCo

NOT EXISTS keeps Canon, Hitachi and NewCo, which have no Gadget match

Recap: Company-Level Questions

cname
Canon
GizmoWorks
Hitachi
NewCo

Each answer should contain company rows, without repeating a company per product

Demo: Ask Whether a Match Exists

1. Find companies that make a Gadget with `EXISTS`
2. Find companies that make no Gadgets with `NOT EXISTS`

```
-- write your query here, then Run
```

QUERY RESULT

Write a query on the left, then Run.

Answer: Match Existence

```
SELECT C.cname FROM Company C WHERE NOT EXISTS (
  SELECT 1 FROM Product P
  WHERE P.manufacturer=C.cname AND
P.category='Gadgets');
```

cname
Canon
Hitachi
NewCo

Three companies have no Gadget; the matching company is GizmoWorks

04 · Stack Results

01	Match Rows	return one row per matching pair
02	Keep Unmatched Rows	keep the left row when no match exists
03	Test for Matches	keep each outer row if a match exists
04	**Stack Results**	append compatible result rows

Matching and Stacking Answer Different Questions

Question	Operation
Which rows belong together?	JOIN
Does a related row exist?	EXISTS
Which rows should survive without a match?	LEFT JOIN / NOT EXISTS
Which compatible lists should be combined?	UNION ALL / UNION

The question determines whether to match pairs, test existence or stack rows

Stacking Overlapping Lists

Gadgets

pname
Gizmo
PowerGizmo

Below 200

pname
Gizmo
PowerGizmo
SingleTouch

Output policy

UNION ALL

5 rows

UNION

3 names

Repeated names survive UNION ALL; UNION keeps one copy

UNION ALL and UNION

```
SELECT pname FROM Product WHERE category='Gadgets'
UNION ALL
SELECT pname FROM Product WHERE price<200;
```

stack both results, duplicates kept

UNION ALL stacks compatible result rows; UNION also removes duplicate rows

Compatible Result Columns

First query	Second query	Combination
pname	pname	One name column
pname, price	pname, price	Two corresponding columns
pname, price	pname	Invalid: different column counts

Stack corresponding columns in the same order

Recap: Overlapping Product Lists

pname	price	category
Gizmo	19.99	Gadgets
PowerGizmo	29.99	Gadgets
SingleTouch	149.99	Photography
MultiTouch	203.99	Household

The Gadget list overlaps the below-200 list at Gizmo and PowerGizmo

Demo: Stack Two Product Lists

1. Combine Gadget names and below-200 product names with `UNION ALL`
2. Change it to `UNION` and explain the difference

```
-- write your query here, then Run
```

QUERY RESULT

Write a query on the left, then Run.

Answer: Stack Product Lists

```
SELECT pname FROM Product WHERE category='Gadgets'
UNION
SELECT pname FROM Product WHERE price<200;
```

pname
Gizmo
PowerGizmo
SingleTouch

UNION keeps three distinct names; UNION ALL keeps all five input rows

Takeaway: SQL II

1. A join returns **matching pairs**; keys determine how rows multiply
2. The question determines **which unmatched rows survive** and whether only existence matters
3. Stacking results requires **compatible columns and an explicit duplicate policy**