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

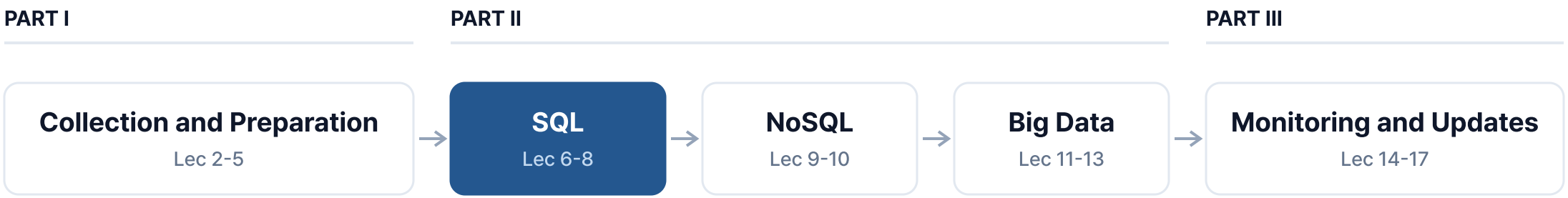
6.

SQL I: Basics

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 I

01	**Read a Table**	find rows and choose columns
02	**Refine a Result**	remove repeats, sort and keep top rows
03	**Handle Missing Values**	choose how absent prices affect the result
04	**Write Safely**	keep constraints and related changes together

01 · Read a Table

01	**Read a Table**	find rows and choose columns
02	**Refine a Result**	remove repeats, sort and keep top rows
03	**Handle Missing Values**	choose how absent prices affect the result
04	**Write Safely**	keep constraints and related changes together

Tables and SQL

A database stores related tables; SQL reads and changes their rows

Term	Today
Table	Named rows and columns
Schema	Column names, types and constraints
SQL	The language for reading and writing tables
SQLite	The database engine running inside this deck

Schema and Rows

A schema defines the columns; rows supply their values

Schema: the definition

```
Product
  pname          TEXT, primary key
  price          REAL
  category       TEXT
  manufacturer   TEXT
```

Rows: the stored values

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

One definition, many product rows

Case study: A Product Catalogue

```
Product(pname, price, category, manufacturer)
```

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

Find products by category, compare prices and prepare a short price list

Four products form the input for filtering, choosing columns and sorting prices

SELECT, FROM and WHERE

The question names one input table, a row condition and output columns

FROM

Read **Product**

→

WHERE

Keep **Gadgets**

→

SELECT

Show **name and price**

```
SELECT pname, price
FROM Product
WHERE category = 'Gadgets';
```

Read the table, keep matching rows, then choose columns

SQL Query: Selection

Before: four products

pname	category
Gizmo	Gadgets
PowerGizmo	Gadgets
SingleTouch	Photography
MultiTouch	Household

After: two Gadgets

pname	category
Gizmo	Gadgets
PowerGizmo	Gadgets

```
SELECT pname, category FROM Product
WHERE category = 'Gadgets';
```

Selection keeps matching rows; their columns remain available

SQL Query: Projection

Input: already filtered rows

pname	price	category
Gizmo	19.99	Gadgets
PowerGizmo	29.99	Gadgets

Output: two chosen columns

pname	price
Gizmo	19.99
PowerGizmo	29.99

```
SELECT pname, price FROM Product
WHERE category = 'Gadgets';
```

Projection chooses columns from the rows already selected

A Query Produces a Table

Product: four rows

pname	price
Gizmo	19.99
PowerGizmo	29.99
SingleTouch	149.99
MultiTouch	203.99

Answer: two rows

pname	price
Gizmo	19.99
PowerGizmo	29.99

```
SELECT pname,price FROM Product WHERE category='Gadgets';
```

The result is another table with named columns and selected rows

Strings and Conditions

```sql
SELECT pname, price
FROM Product
WHERE category = 'Gadgets' AND price > 20;
```

↑ strings in single quotes

Parentheses Change the Filter

```
SELECT pname,category,price FROM Product
WHERE (category='Gadgets' OR category='Photography')
  ↑ OR inside the parentheses first, then AND
  AND price>20;
```

pname	category	price
PowerGizmo	Gadgets	29.99
SingleTouch	Photography	149.99

Parentheses apply the price condition to both selected categories

IN Lists the Allowed Values

```
SELECT pname,category FROM Product
WHERE category IN ('Gadgets','Photography');
```

↑ any value in the list

pname	category
Gizmo	Gadgets
PowerGizmo	Gadgets
SingleTouch	Photography

IN selects the three products in either named category

Recap: The Product Table

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

Predict the remaining rows and output columns before writing the query

Demo: Read the Product Table

1. Return the name and price of every Gadget product
2. Keep only Gadget products priced above 20

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

QUERY RESULT

Write a query on the left, then Run.

Answer: Read the Product Table

```
SELECT pname,price FROM Product
WHERE category='Gadgets' AND price>20;
```

pname	price
PowerGizmo	29.99

Adding the price condition keeps PowerGizmo and removes Gizmo

02 · Refine a Result

01	Read a Table	find rows and choose columns
02	**Refine a Result**	remove repeats, sort and keep top rows
03	Handle Missing Values	choose how absent prices affect the result
04	Write Safely	keep constraints and related changes together

Caveat: Duplicates in the Output

Four product rows

pname	category
Gizmo	Gadgets
PowerGizmo	Gadgets
SingleTouch	Photography
MultiTouch	Household

→

SELECT category

category
Gadgets
Gadgets
Photography
Household

→

SELECT DISTINCT category

category
Gadgets
Photography
Household

Two products share Gadgets; selecting a column keeps both copies until DISTINCT removes one

ORDER BY: Sorting the Results

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Category='Gadgets' AND Price > 20
ORDER BY Price, PName
```

by Price, ties by PName

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

LIMIT and Deterministic Order

```
SELECT pname, price FROM Product
ORDER BY price DESC, pname ASC LIMIT 2;
```

↑ first 2 rows after the sort

pname	price
MultiTouch	203.99
SingleTouch	149.99

Sort before limiting; an explicit tie-breaker makes the chosen rows reproducible

Recap: Categories and Prices

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

The input repeats Gadgets twice; the two Gadget prices are different

Demo: Prepare a Price List

1. List Gadget names and prices, highest price first; return only the first product
2. List each category once with `DISTINCT`

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

QUERY RESULT

Write a query on the left, then Run.

Answer: Prepare a Price List

```sql
SELECT pname,price FROM Product
WHERE category='Gadgets'
ORDER BY price DESC,pname LIMIT 1;
```

pname	price
PowerGizmo	29.99

ORDER BY defines the order; LIMIT keeps the first row

Answer: Unique Categories

```sql
SELECT DISTINCT category FROM Product;
```

category
Gadgets
Photography
Household

Two Gadgets inputs become one category row

03 · Handle Missing Values

01	Read a Table	find rows and choose columns
02	Refine a Result	remove repeats, sort and keep top rows
03	**Handle Missing Values**	choose how absent prices affect the result
04	Write Safely	keep constraints and related changes together

NULL and Missing Values

pname	price
Gizmo	19.99
PowerGizmo	NULL
SingleTouch	149.99
MultiTouch	203.99

NULL means the price is missing; test it with IS NULL

This copy has a missing PowerGizmo price; NULL is different from a zero price

NULL Comparisons

Expression	Result
NULL = NULL	UNKNOWN
NULL <> 5	UNKNOWN
NULL IS NULL	TRUE
NULL IS NOT NULL	FALSE

WHERE keeps only TRUE; FALSE and UNKNOWN both drop the row

An absent value needs IS NULL; equality cannot identify a missing match

Caveat: Missing Prices in a Filter

Missing-price input

pname	price
Gizmo	19.99
PowerGizmo	NULL
SingleTouch	149.99
MultiTouch	203.99

WHERE price > 20

pname	price
SingleTouch	149.99
MultiTouch	203.99

PowerGizmo: the price is unknown, so the comparison cannot be true

PowerGizmo disappears because its price is unknown; NULL does not become zero

Recap: The Missing-Price Copy

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

PowerGizmo is the only product with a missing price in this copy

Demo: Handle Missing Prices

1. Find products with a missing price
2. Return products priced above 20, and also include products with a missing price

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

QUERY RESULT

Write a query on the left, then Run.

Answer: Handle Missing Prices

```
SELECT pname,price FROM Product
WHERE price>20 OR price IS NULL;
```

pname	price
PowerGizmo	NULL
SingleTouch	149.99
MultiTouch	203.99

The explicit NULL condition includes PowerGizmo alongside the two known prices above 20

04 · Write Safely

01	Read a Table	find rows and choose columns
02	Refine a Result	remove repeats, sort and keep top rows
03	Handle Missing Values	choose how absent prices affect the result
04	**Write Safely**	keep constraints and related changes together

Keys and References

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

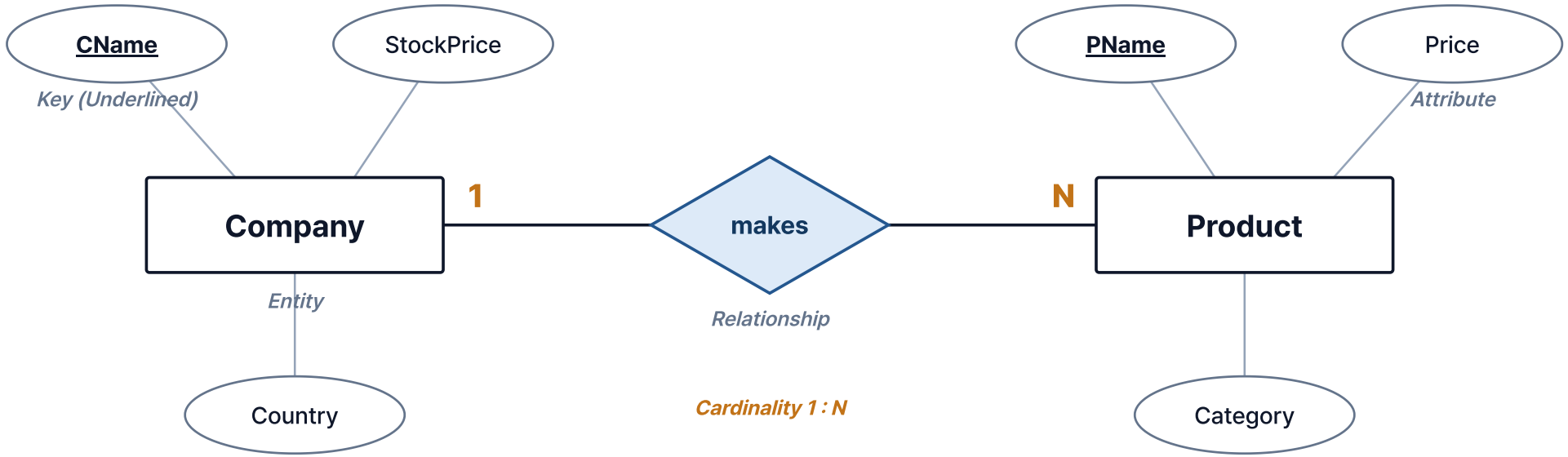
pname	manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks
SingleTouch	Canon
MultiTouch	Hitachi

A primary key identifies a row; a foreign key requires an existing referenced key

Each product names one existing company; the key columns identify the referenced rows

Entity–Relationship (ER) Diagram

An ER diagram describes entities, attributes and relationships before table design



One company **makes** many products: **1** on one side, **N** on the other

ER to Tables

Each relationship becomes either a foreign key column or a table of its own

1 : N



N : M



Use a **foreign key** for 1:N and a **junction table** for N:M

Junction Table

If a product may sit in several categories, Category leaves Product and gets its own table

Product

PName	Manufacturer
Gizmo	GizmoWorks
PowerGizmo	GizmoWorks
SingleTouch	Canon
MultiTouch	Hitachi

ProductCategory

PName	CatName
Gizmo	Gadgets
PowerGizmo	Gadgets
SingleTouch	Photography
SingleTouch	Gadgets
MultiTouch	Household

Category

CatName	Department
Gadgets	Electronics
Photography	Electronics
Household	Home

`ProductCategory: PRIMARY KEY (PName, CatName) · PName → Product.PName · CatName → Category.CatName`

SingleTouch has two **category links** but one product row

A Composite Key Identifies a Pair

```
CREATE TABLE ProductCategory (
  pname TEXT NOT NULL REFERENCES Product(pname),
  catName TEXT NOT NULL,
  PRIMARY KEY (pname,catName)   the pair is unique
);
```

CREATE TABLE and INSERT

```
CREATE TABLE Product (
  pname TEXT PRIMARY KEY NOT NULL,
  price REAL, category TEXT, manufacturer TEXT NOT NULL,
  FOREIGN KEY (manufacturer) REFERENCES Company(cname)
    ↑ must exist in Company
);
INSERT INTO Product VALUES ('Gizmo',19.99,'Gadgets','GizmoWorks');
```

Caveat: A Repeated Primary Key

```
INSERT INTO Product VALUES
('Gizmo',99,'Gadgets','GizmoWorks');
```

SQLITE SAYS

```
UNIQUE constraint failed: Product.pname
```

A repeated product key fails even when the other values differ

UPDATE and DELETE

```
UPDATE Product SET price = 24.99 WHERE pname =
'Gizmo';
  ↑ WHERE picks the row to change
SELECT pname, price FROM Product;
```

pname	price
Gizmo	24.99
PowerGizmo	29.99
SingleTouch	149.99
MultiTouch	203.99

```
DELETE FROM Product WHERE pname = 'Gizmo';
-- without WHERE, UPDATE or DELETE affects every row
```

The condition identifies the write target; Gizmo changes while the other prices remain

Caveat: A Write Without WHERE

```
UPDATE Product SET price=24.99;
```

↑ no WHERE: every row changes

```
SELECT pname,price FROM Product;
```

pname	price
Gizmo	24.99
PowerGizmo	24.99
SingleTouch	24.99
MultiTouch	24.99

All four prices change when UPDATE has no target condition

Caveat: Deleting a Referenced Company

```
DELETE FROM Company WHERE cname='GizmoWorks';
```

SQLITE SAYS

```
FOREIGN KEY constraint failed
```

A company referenced by products cannot be deleted under the default policy

Transactions

Acct	Balance
a10	20,000
a20	15,000

Transfer $3k from a10 to a20:
1. Debit $3k from a10
2. Credit $3k to a20

Acct	Balance
a10	17,000
a20	18,000

A transaction commits related changes together; ROLLBACK cancels uncommitted changes

Both balance changes must complete together; one completed change would lose money

ACID Describes Transaction Guarantees

Property	Meaning
Atomicity	Related changes succeed or fail together
Consistency	Declared constraints hold after the transaction
Isolation	Isolation level controls concurrent observations
Durability	Committed work survives a failure

ACID connects the write boundary to constraints, concurrency and recovery

Caveat: A Crash Between Two Writes

Two separately committed writes can leave a half-finished transfer

Before

Account	Balance
a10	20,000
a20	15,000

→

Debit only: crash

Account	Balance
a10	17,000
a20	15,000

→

Both writes

Account	Balance
a10	17,000
a20	18,000

The debit-only state loses 3,000 from the combined balance

BEGIN, COMMIT and ROLLBACK

```
BEGIN;
UPDATE Product SET price=24.99 WHERE pname='Gizmo';
SELECT pname,price FROM Product WHERE pname='Gizmo';
ROLLBACK;  undo everything since BEGIN
SELECT pname,price FROM Product WHERE pname='Gizmo';
```

Recap: Valid Writes

cname
Canon
GizmoWorks
Hitachi

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

Gizmo starts at 19.99; Google is absent from Company

Demo: Change a Price Safely

1. Change Gizmo to 24.99 inside a transaction, inspect it, then roll back and inspect it again
2. Try inserting a product made by Google, which is absent from Company

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

QUERY RESULT

Write a query on the left, then Run.

Answer: Roll Back a Price Change

```
BEGIN;
UPDATE Product SET price=24.99 WHERE pname='Gizmo';
SELECT price FROM Product WHERE pname='Gizmo';
ROLLBACK;
SELECT price FROM Product WHERE pname='Gizmo';
```

price
19.99

ROLLBACK restores Gizmo to 19.99 in the current transaction

Answer: A Missing Manufacturer

```
INSERT INTO Product
VALUES('NewGizmo',10,'Gadgets','Google');
```

SQLITE SAYS

```
FOREIGN KEY constraint failed
```

The foreign key rejects a manufacturer absent from Company

Takeaway: SQL I

1. A query defines **which rows and columns** answer the question
2. **Duplicates, order and missing values** change what the result means
3. **Keys constrain valid data**; transactions keep related writes together