Advanced Querying to Extract Business Insights

Agenda

- Numeric functions
- String functions
- Date functions
- Joining data

Let’s begin the discussion by answering a few questions

Querying Data with SQL: Quiz

A billing system needs to ensure that discount percentages are always rounded down. Which function is best suited for this?

A ROUND()

B CEIL()

C FLOOR()

D UPPER()

Querying Data with SQL: Quiz

A billing system needs to ensure that discount percentages are always rounded down. Which function is best suited for this?

A ROUND()

B CEIL()

C [✓ FLOOR()]

D UPPER()

Numerical Functions: Rounding Values

Syntax:

```
ROUND(<value>, <decimal_places>)
```

Examples:

```
ROUND(42.56789, 3) ----> 42.568
ROUND(42.56789, 1) ----> 42.6
ROUND(42.56789, 0) ----> 43
```

Syntax:

```
FLOOR(<value>)
```

Examples:

```
FLOOR(42.9) ----> 42
FLOOR(42.1) ----> 42
```

Syntax:

```
CEIL(<value>)
```

Examples:

```
FLOOR(42.9) ----> 43
FLOOR(42.1) ----> 43
```

Other Numerical Functions

Syntax:

```
ABS(<value>)
```

Examples:

`ABS(-4)`	`4`
`ABS(4.1)`	`4.1`
`ABS(0)`	`0`

Syntax:

```
SQRT(<value>)
```

Examples:

`SQRT(4)`	`2.0`
`SQRT(-4)`	`NULL`

Syntax:

```
POW(<base>,<exponent>)
```

Examples:

`POW(2,4)`	`16.0`
`POW(2,0)`	`1.0`
`POW(-2,2)`	`4.0`
`POW(-2,3)`	`-8.0`

Querying Data with SQL: Quiz

A courier partner has shipped an order late, requiring the delivery date to be postponed by 7 days. Which SQL function should be used to update the delivery date in the database?

A DATE(shipping_date, '+7 days')

B DATE_ADD(shipping_date, INTERVAL 7 DAY)

C DATETIME_ADD(shipping_date, 7, 'days')

D DATE_ADD(shipping_date, 7 DAY)

Querying Data with SQL: Quiz

A courier partner has shipped an order late, requiring the delivery date to be postponed by 7 days. Which SQL function should be used to update the delivery date in the database?

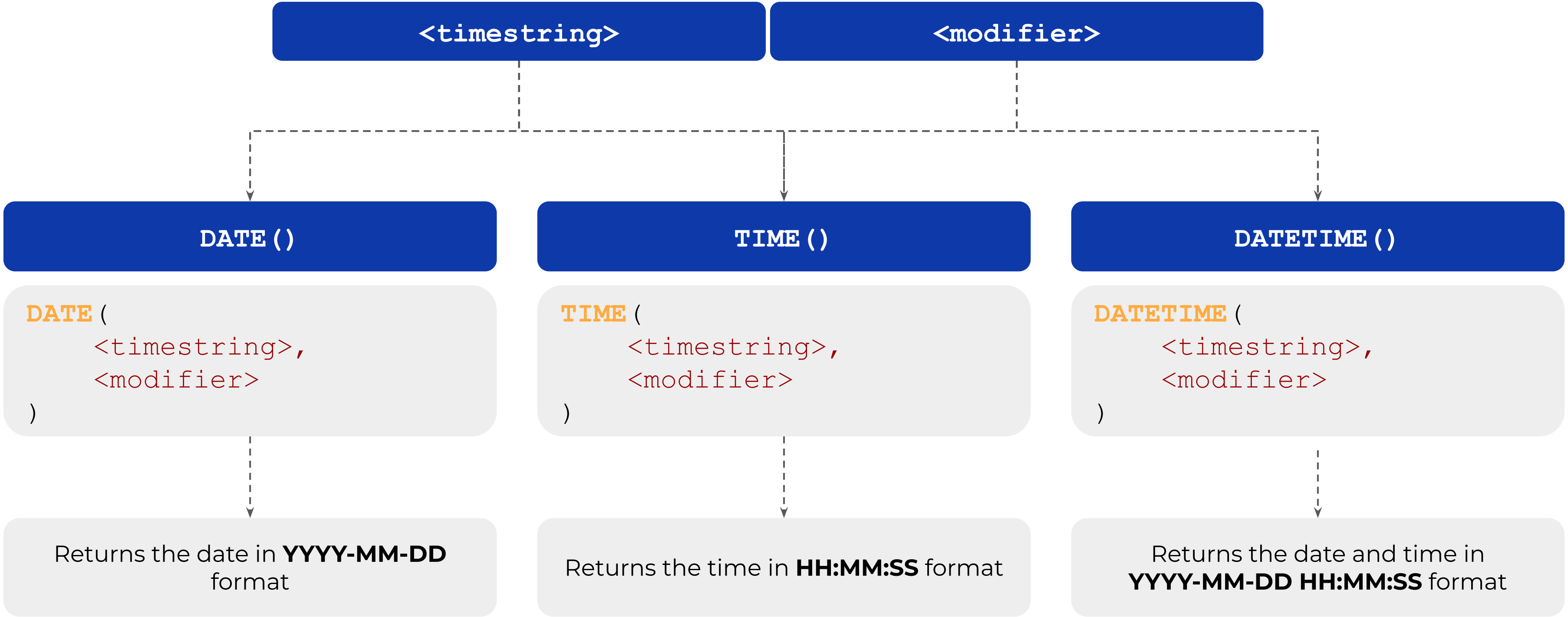
A [✓ DATE(shipping_date, '+7 days')]

B DATE_ADD(shipping_date, INTERVAL 7 DAY)

C DATETIME_ADD(shipping_date, 7, 'days')

D DATE_ADD(shipping_date, 7 DAY)

Date Functions



<timestring>

<modifier>

DATE()

```
DATE(
    <timestring>,
    <modifier>
)
```

Returns the date in **YYYY-MM-DD** format

TIME()

```
TIME(
    <timestring>,
    <modifier>
)
```

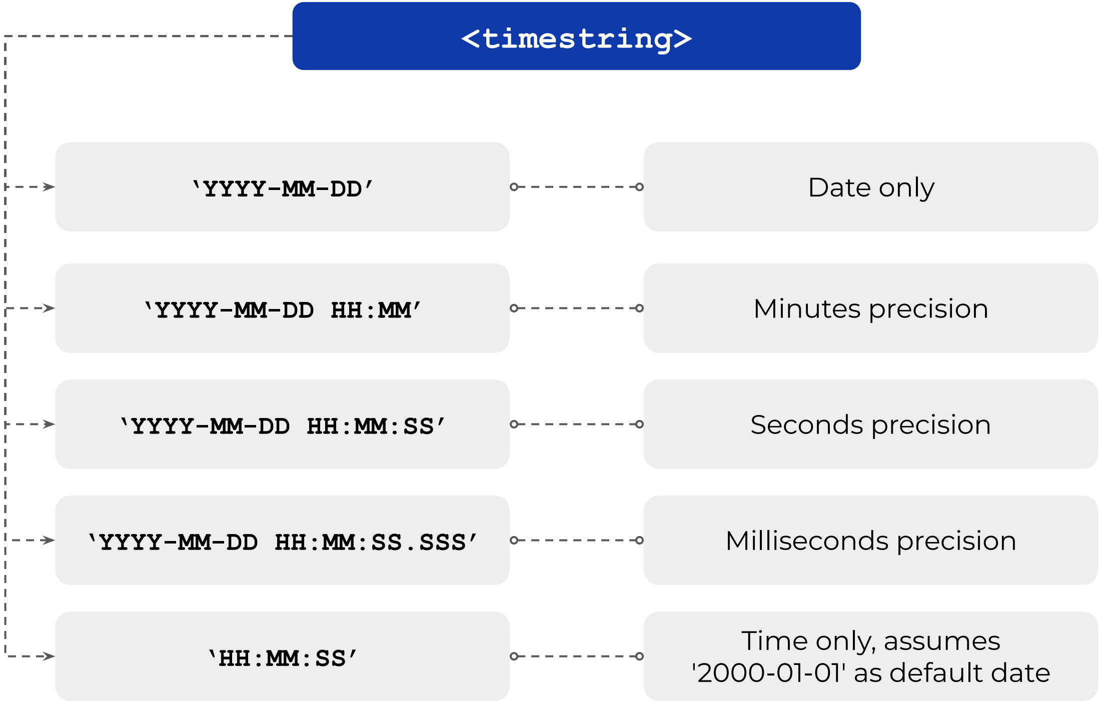
Returns the time in **HH:MM:SS** format

DATETIME()

```
DATETIME(
    <timestring>,
    <modifier>
)
```

Returns the date and time in **YYYY-MM-DD HH:MM:SS** format

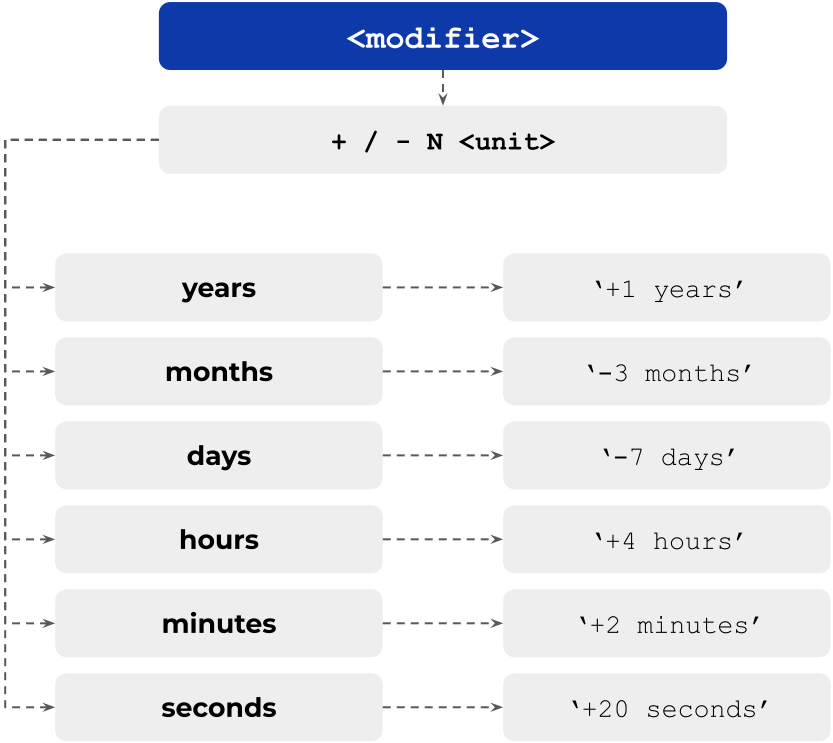
Date Functions



<timestring>

'YYYY-MM-DD'	Date only
'YYYY-MM-DD HH:MM'	Minutes precision
'YYYY-MM-DD HH:MM:SS'	Seconds precision
'YYYY-MM-DD HH:MM:SS.SSS'	Milliseconds precision
'HH:MM:SS'	Time only, assumes '2000-01-01' as default date

Date Functions



`<modifier>`

`+ / - N <unit>`

years	'+1 years'
months	'-3 months'
days	'-7 days'
hours	'+4 hours'
minutes	'+2 minutes'
seconds	'+20 seconds'

Querying Data with SQL: Quiz

An e-commerce platform wants to clean up customer input by removing any leading and trailing spaces from names. Which of the following functions can they use to update the value in their database?

A TRIM(customer_name)

B CLEAN(customer_name)

C LTRIM(customer_name)

D RTRIM(customer_name)

Querying Data with SQL: Quiz

An e-commerce platform wants to clean up customer input by removing any leading and trailing spaces from names. Which of the following functions can they use to update the value in their database?

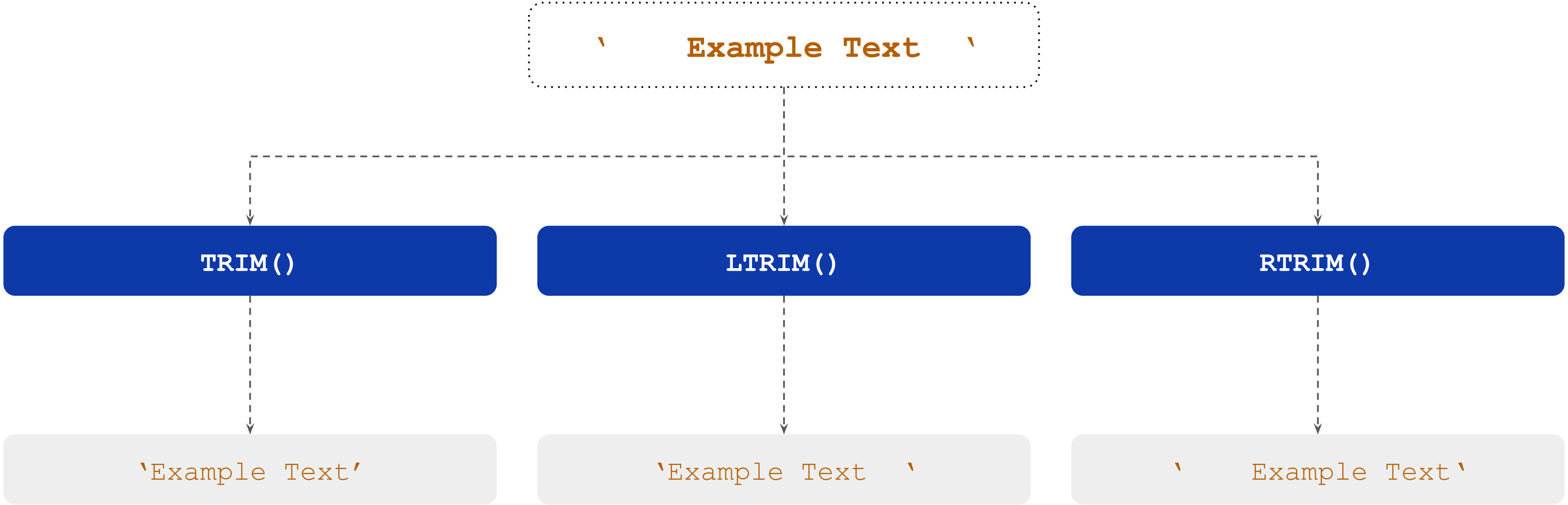
[✓ A TRIM(customer_name)]

B CLEAN(customer_name)

C LTRIM(customer_name)

D RTRIM(customer_name)

String Functions: Trimming



`     Example Text   `

TRIM()

LTRIM()

RTRIM()

`Example Text’

`Example Text   `

`     Example Text`

Querying Data with SQL: Quiz

A warehouse needs to extract the first four characters from a product code, which represents the product category.

How can they achieve this?

A REPLACE(product_code, 1, 4)

B SUBSTR(product_code, 1, 4)

C SUBSTR(product_code, 4, 1)

D REPLACE(product_code, 4, 1)

Querying Data with SQL: Quiz

A warehouse needs to extract the first four characters from a product code, which represents the product category.

How can they achieve this?

A REPLACE(product_code, 1, 4)

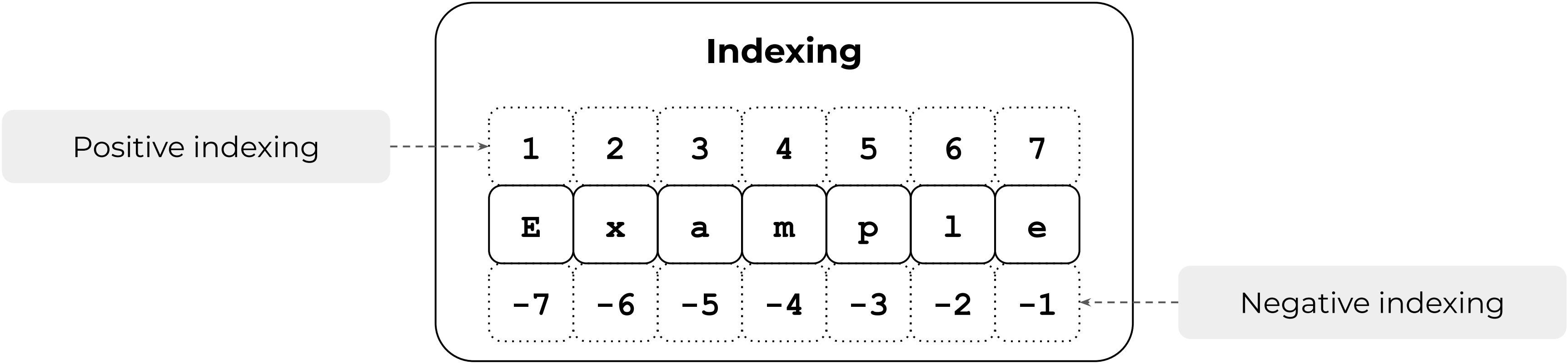
[✓ B SUBSTR(product_code, 1, 4)]

C SUBSTR(product_code, 4, 1)

D REPLACE(product_code, 4, 1)

String Functions: Substring

```
SUBSTR(<string>, <start_index>, <length>)
```



Indexing

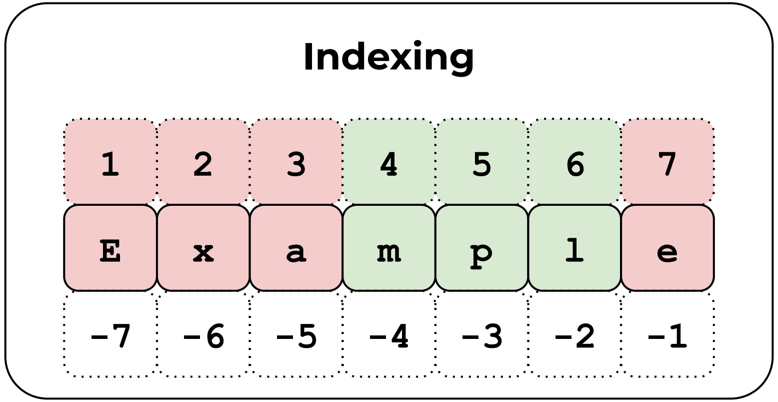
Positive indexing

1	2	3	4	5	6	7
E	x	a	m	p	l	e
-7	-6	-5	-4	-3	-2	-1

Negative indexing

String Functions: Substring

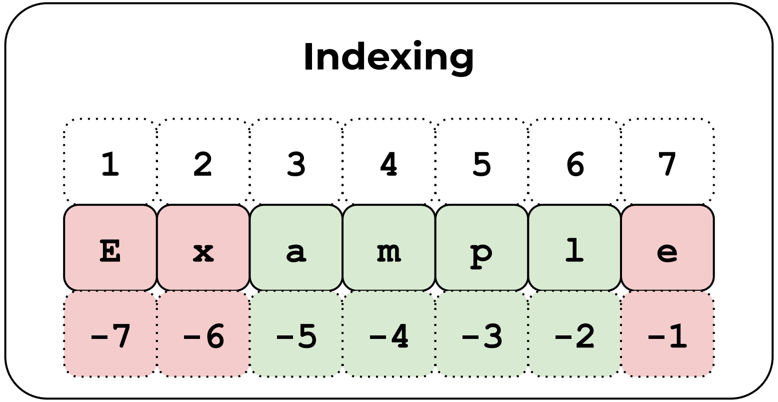
```
SUBSTR(‘Example’, 4, 3)
```



Indexing

1	2	3	4	5	6	7
E	x	a	m	p	l	e
-7	-6	-5	-4	-3	-2	-1

```
SUBSTR(‘Example’, -5, 4)
```



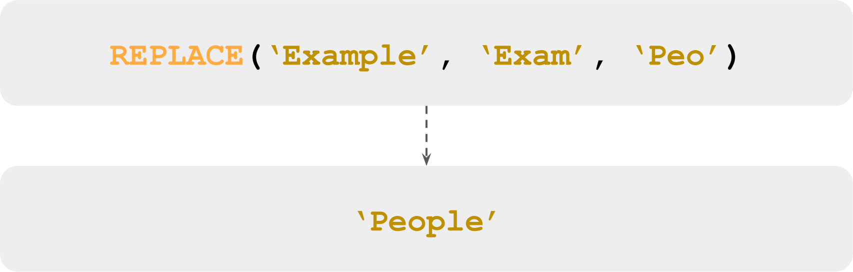
Indexing

1	2	3	4	5	6	7
E	x	a	m	p	l	e
-7	-6	-5	-4	-3	-2	-1

String Functions: Replace

```
REPLACE (<value>, <search_string>, <replacement_string>)
```

The **REPLACE()** function in SQLite **replaces all occurrences** of search_string with replacement_string in original_string, and **it is case-sensitive**.



```
REPLACE('Example', 'Exam', 'Peo')
```

```
'People'
```

Querying Data with SQL: Quiz

A company needs a report showing only employees who are assigned to a department. Employees without a department should be excluded. Which SQL join should be used?

A employees JOIN departments

B employees LEFT JOIN departments

C employees RIGHT JOIN departments

D employees FULL OUTER JOIN departments

Querying Data with SQL: Quiz

A company needs a report showing only employees who are assigned to a department. Employees without a department should be excluded. Which SQL join should be used?

A [✓ employees JOIN departments]

B employees LEFT JOIN departments

C employees RIGHT JOIN departments

D employees FULL OUTER JOIN departments

Joins: Inner Join

JOIN / INNER JOIN

Returns only rows where there is a **match in both tables**

employees

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL

departments

dept_id	dept_name
10	HR
20	IT
40	Finance

employees JOIN departments

emp_id	emp_name	dept_id	dept_id	dept_name
1	Alice	10	10	HR
2	Bob	20	20	IT

Joins: Left Join

LEFT JOIN

Returns **all rows from the left table**, plus matching rows from the right table (or NULL if no match)

employees

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL

departments

dept_id	dept_name
10	HR
20	IT
40	Finance

employees LEFT JOIN departments

emp_id	emp_name	dept_id	dept_id	dept_name
1	Alice	10	10	HR
2	Bob	20	20	IT
3	Charlie	30	NULL	NULL
4	David	NULL	NULL	NULL

Joins: Right Join

RIGHT JOIN

Returns **all rows from the right table**, plus matching rows from the left table (or NULL if no match)

`employees`

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL

`departments`

dept_id	dept_name
10	HR
20	IT
40	Finance

`employees RIGHT JOIN departments`

emp_id	emp_name	dept_id	dept_id	dept_name
1	Alice	10	10	HR
2	Bob	20	20	IT
NULL	NULL	NULL	40	Finance

Joins: Full Outer Join

FULL OUTER JOIN

Returns **all rows from both tables**, filling unmatched columns with NULL

employees

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL

departments

dept_id	dept_name
10	HR
20	IT
40	Finance

employees FULL OUTER JOIN departments

emp_id	emp_name	dept_id	dept_id	dept_name
1	Alice	10	10	HR
2	Bob	20	20	IT
3	Charlie	30	NULL	NULL
4	David	NULL	NULL	NULL
NULL	NULL	NULL	40	Finance

Happy Learning !