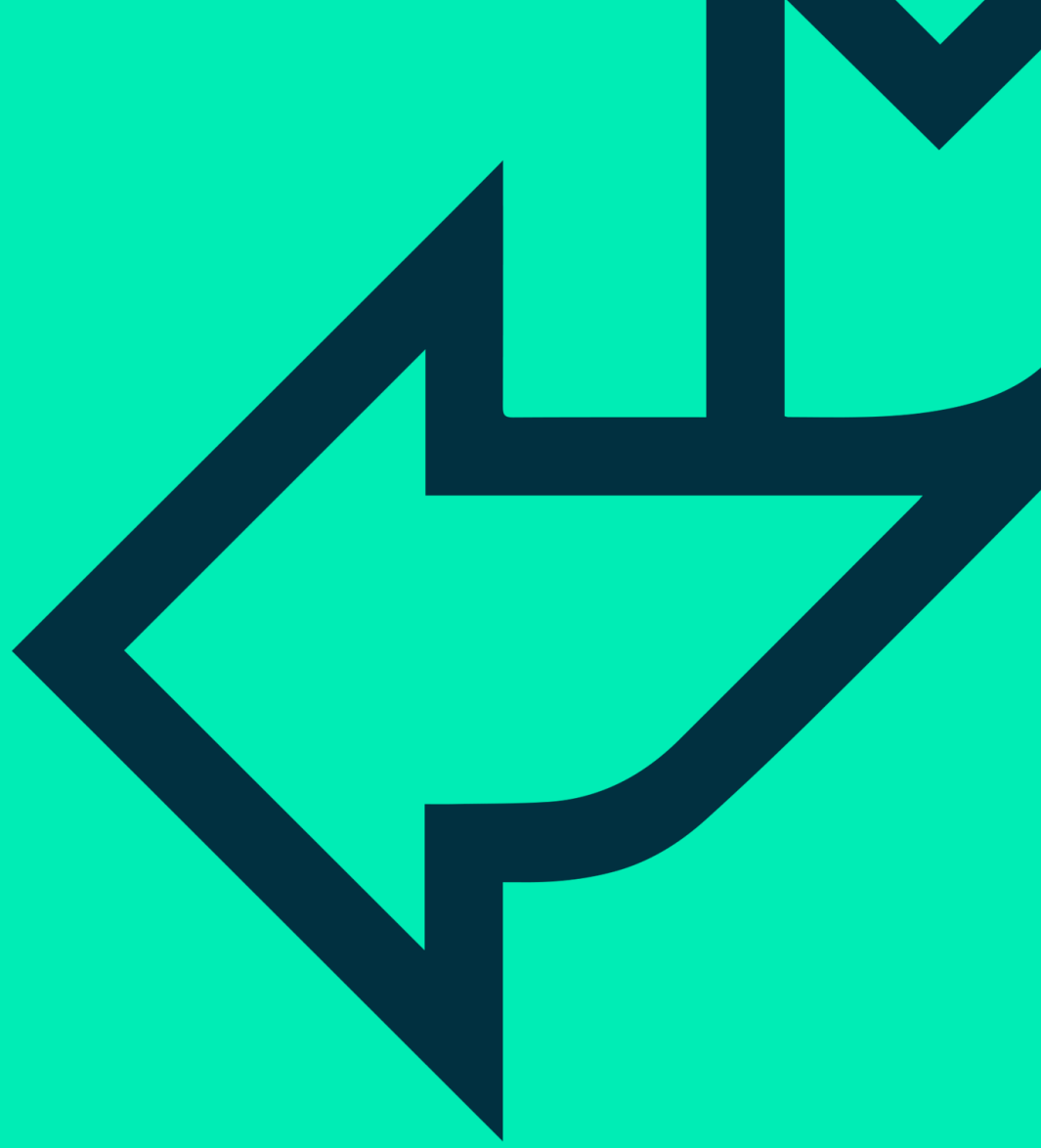
POWERING POTENTIAL

SQL:
Filtering Rows

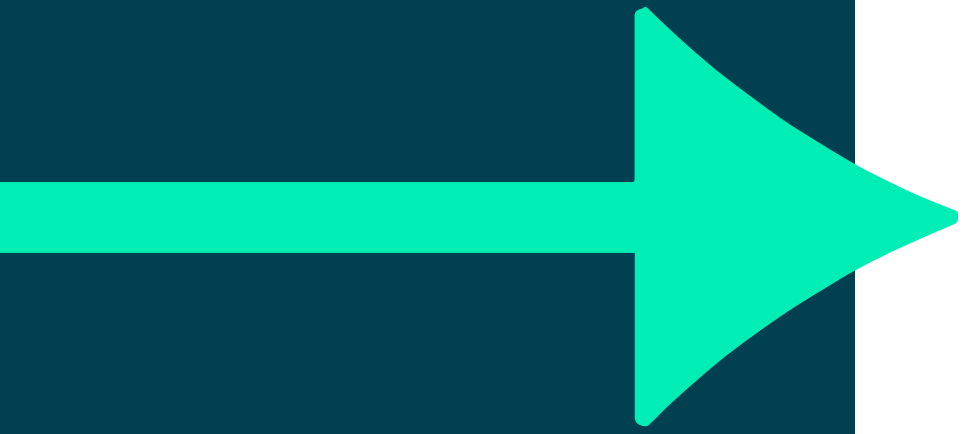
SQL

Lesson Objectives and Contents

- → Filtering rows
- → Comparison, logical and special operators



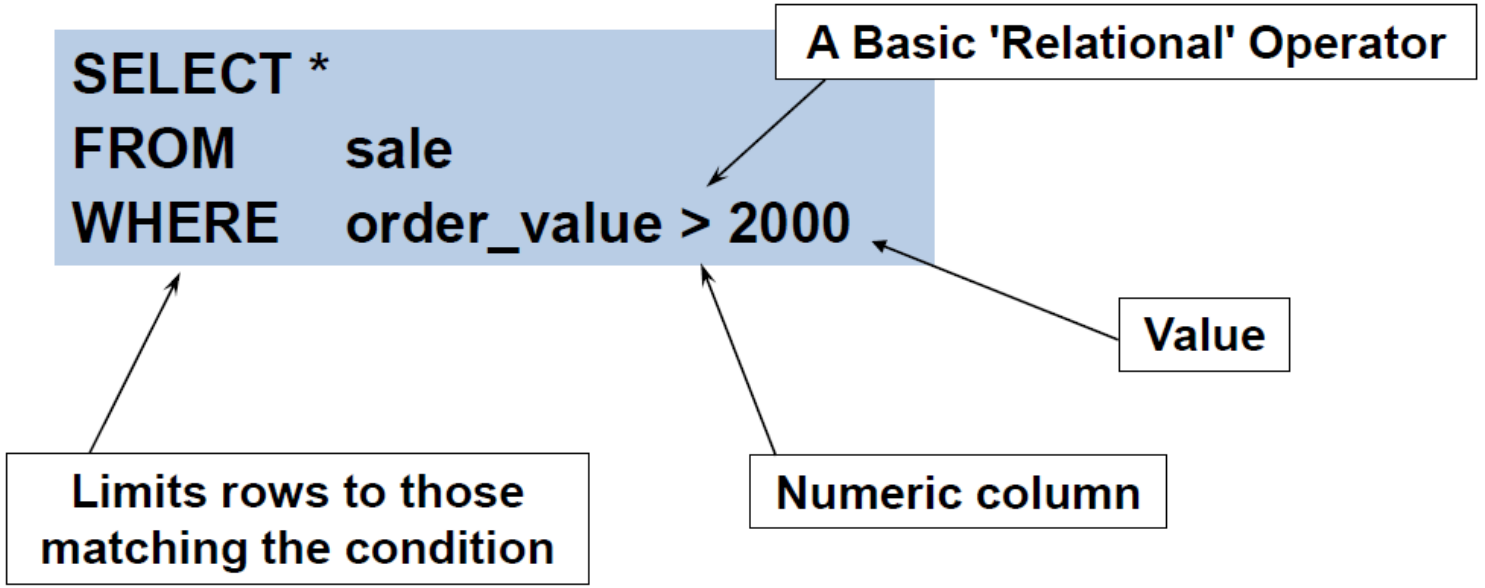
SELECT STATEMENT: WHERE



SELECT <<field(s)>>
FROM <<table(s)>>
WHERE <<condition(s)>>
GROUP BY <<field(s)>>
HAVING <<condition(s)>>
ORDER BY <<field(s)>>

COMPARISON OPERATORS

Using comparison operators



- A row is included in the result set if the test returns true.
- Basic operators include >, >=, <, <=, =, <>

HANDLING NULLS

- **Basic premise of an RDBMS is the concept of optional columns**
  - NULL means 'not applicable' or 'unknown', different from zero or blank
- **On INSERT of a row, must supply values for mandatory columns**
  - Other columns may be left as NULL (assuming no 'DEFAULT' value)
- **WHERE clause expressions will evaluate to TRUE, FALSE or NULL**
  - Need to think three-way logic
  - Only rows whose expressions evaluate to TRUE are output
- **Can use IS NULL to retrieve rows with NULL entries:**

```
SELECT  *
FROM    salesperson
WHERE   notes IS NULL
```

LOGICAL OPERATORS

Using logical operators AND / OR

Both conditions must be true

```
SELECT    *
FROM      employee
WHERE     county = 'Surrey' AND name = 'Smith'
```

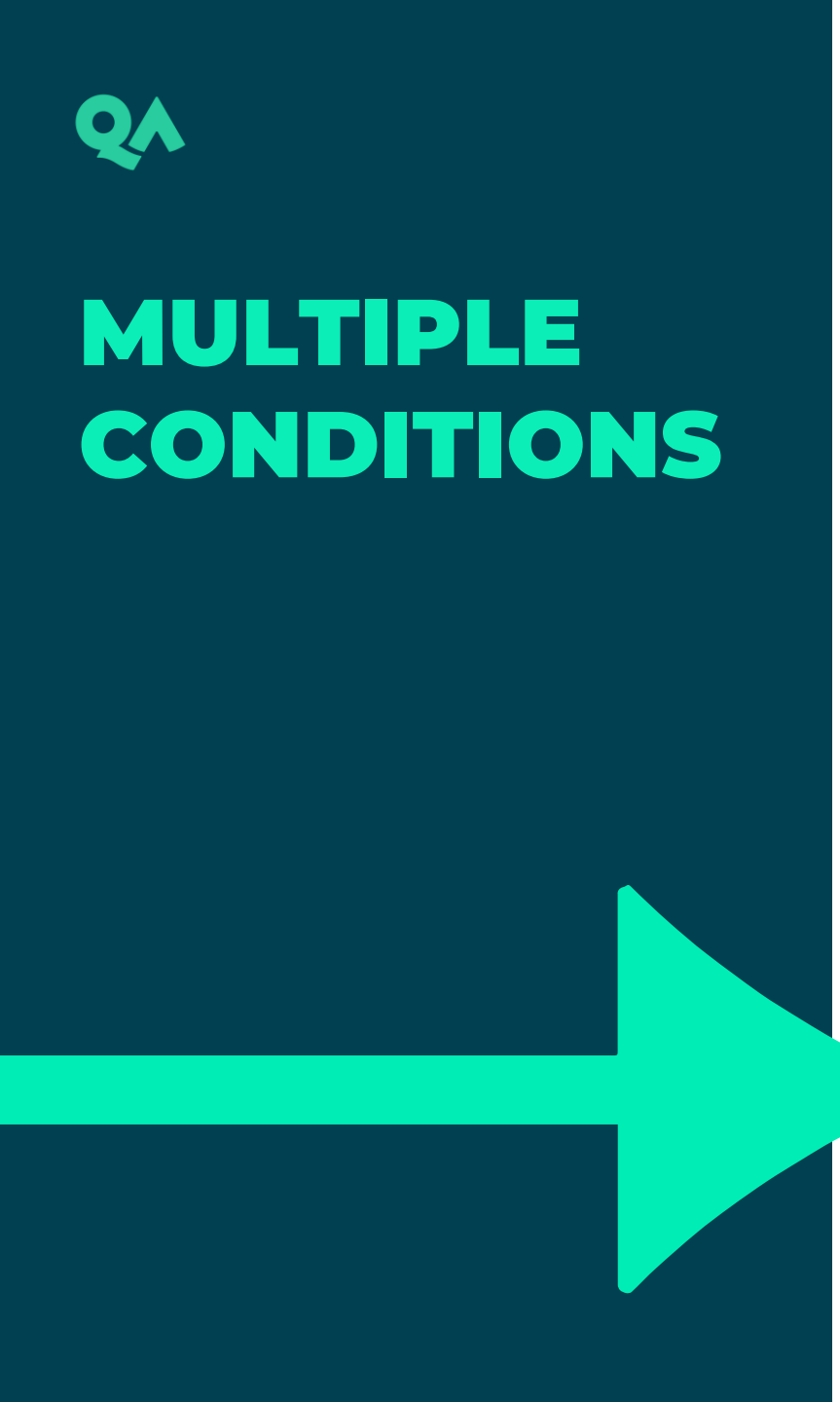
Every time you say AND you are likely to get less rows

Either (or both) conditions can be true

```
SELECT    *
FROM      employee
WHERE     county = 'Surrey' OR name = 'Smith'
```

Every time you say OR you are likely to get more rows

Notice with AND or OR in WHERE clause both Column and Value must be specified



Multiple conditions with logical operators

In the operators precedence, AND precedes OR

- In absence of brackets AND will be executed first then OR

Say you want all employees in either Surrey or Harlow but must have Name of Smith

```
SELECT    *
FROM      employee
WHERE     county = 'Surrey'
OR        town = 'Harlow'
AND       name = 'Smith'
```

This will be the order SQL will run the query if no brackets applied

```
SELECT    *
FROM      employee
WHERE     county = 'Surrey'
OR        (town = 'Harlow'
AND       name = 'Smith')
```

Anything different must be explicitly stated with brackets around what to execute first

```
SELECT    *
FROM      employee
WHERE     (county = 'Surrey'
OR        town = 'Harlow')
AND       name = 'Smith'
```

Using special operator BETWEEN

```
SELECT *
FROM     salesperson
WHERE    sales_target BETWEEN 100 AND 500
```

Starting Value

Stopping Value

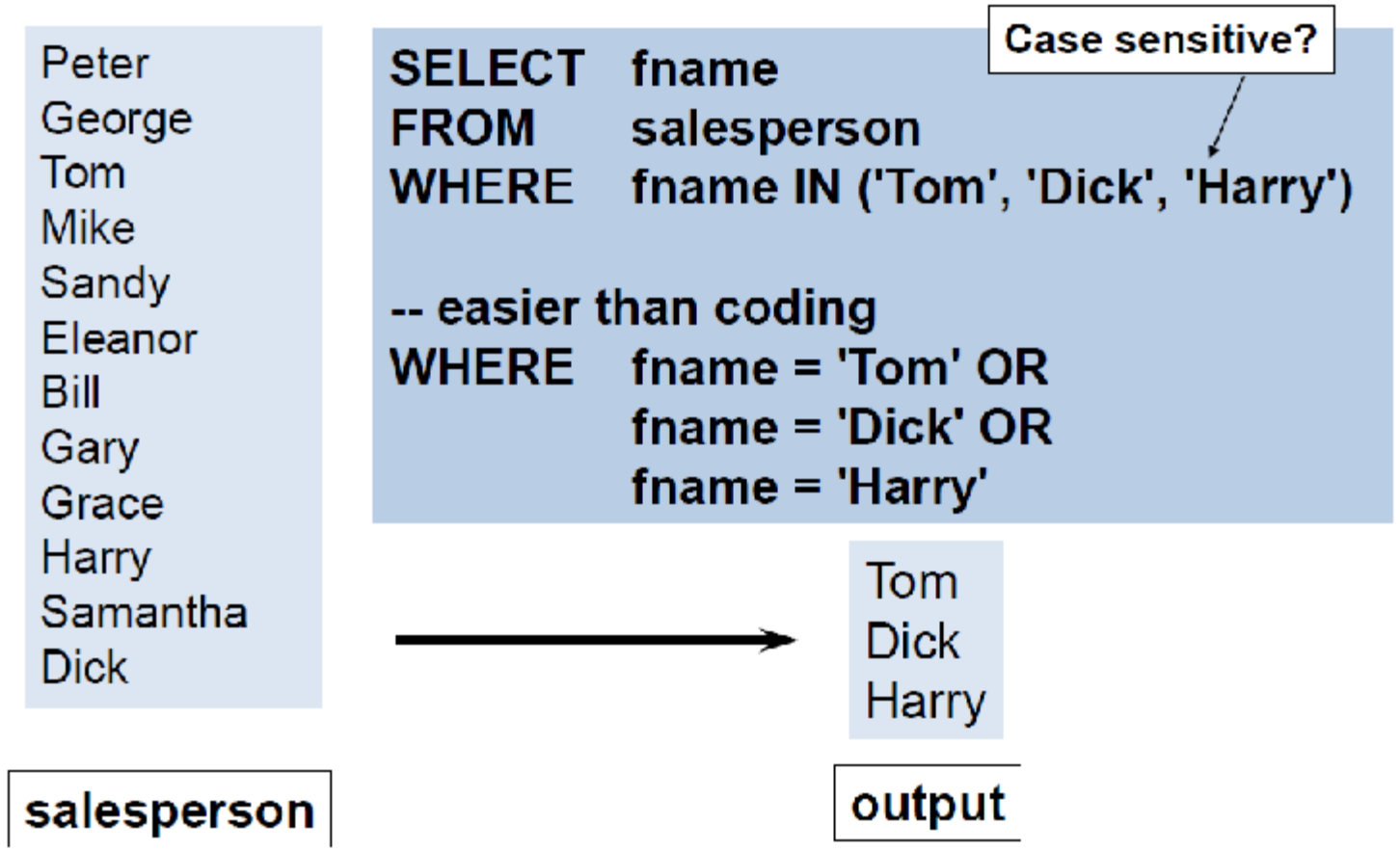
- A lot easier than typing

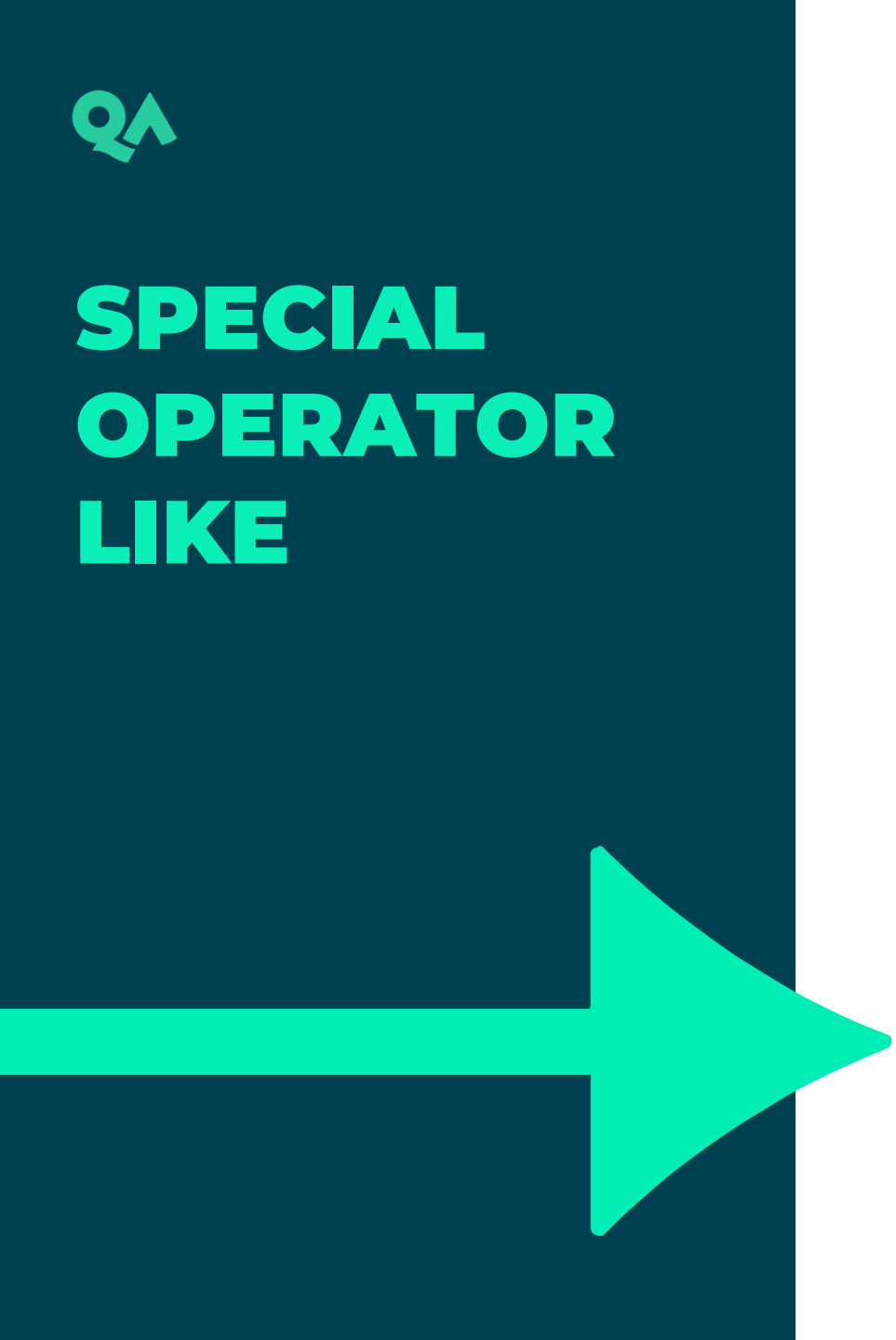
```
WHERE    sales_target >= 100 AND
         sales_target <= 500
```

- Values are inclusive

SPECIAL OPERATOR IN

Using special operator IN





Using special operator LIKE

- When used with 'Like'
  - '_' Matches any single character
  - '%' Matches any number of characters (incl 0!)

```
WHERE   lname LIKE 'a%'      <-- starts with 'A'
WHERE   lname LIKE '%a'      <-- ends with 'A'
WHERE   lname LIKE '%a%'     <-- contains an 'A'
WHERE   lname LIKE '_t%n%r_'
```

has a 'T' in pos 2
has an 'R' 1 char from end
and an 'N' between them

- Note 'Like' is only used with character columns | e.g 'stationary'

LIMIT RESULTS: TOP N

- **SELECT TOP *n***
- **SELECT TOP (*n* PERCENT)**
- **SELECT TOP … WITH TIES**

```
SELECT TOP 6 WITH TIES
   order_no,
   order_value
FROM
   sale
ORDER BY
   order_value DESC
```

Results | Messages

	order_no	order_value
1	600	27
2	300	12
3	100	7
4	200	6
5	400	5
6	700	3
7	800	3

7) | QAStore | 00:00:00 | 7 rows

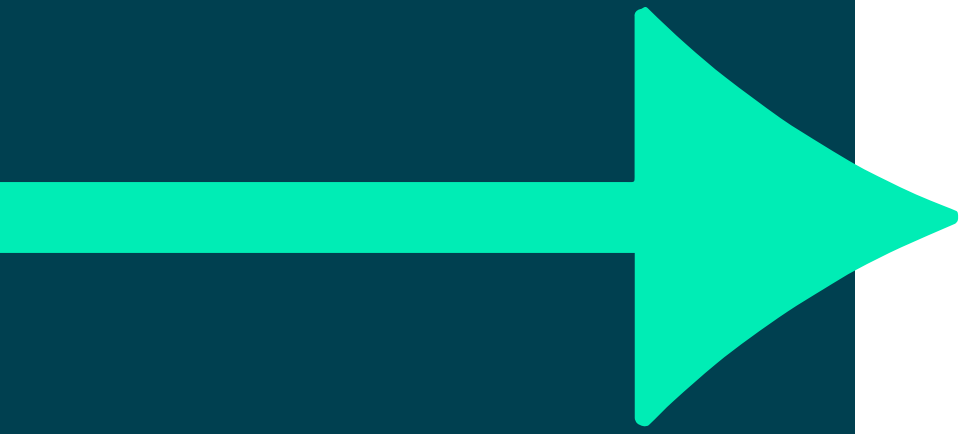
THE CASE EXPRESSION

- **Row level flow of control**
- The closest you get to If..elseif….elseif….else logic in SQL
- **2 Formats: *Simple* Case Expression and *Searched* Case Expression**
  - Example below is ***Searched*** Case Expression => more flexibility allowed

```
SELECT SALES_TARGET, CASE
        WHEN SALES_TARGET <= 5  THEN 'LEVEL 0'
        WHEN SALES_TARGET > 5 AND  SALES_TARGET<= 10 THEN 'LEVEL 1'
        WHEN SALES_TARGET > 10 AND SALES_TARGET <= 15 THEN 'LEVEL 2'
        WHEN SALES_TARGET > 15 AND SALES_TARGET <= 25 THEN 'LEVEL 3'
        WHEN SALES_TARGET> 25 AND SALES_TARGET <= 50 THEN 'LEVEL 4'
        ELSE 'OTHER'
        END AS 'TARGET LEVEL'
FROM DEPT
ORDER BY 'TARGET LEVEL'
```

NOTE: The conditions must not overlap, otherwise the result is reliant on their order.

SELECT STATEMENT: WHERE



A few examples using Northwind:

Comparison operators: =, <, >, <>, !=, >=, <=

```
SELECT ProductID, ProductName FROM Products
WHERE UnitPrice >= 30
```

Logical operators: AND, OR, NOT

```
SELECT ProductID, ProductName FROM Products
WHERE UnitPrice >= 30 AND UnitsInStock > 0
```

“Special” operators: BETWEEN, IN

```
SELECT ProductID, ProductName FROM Products
WHERE UnitPrice BETWEEN 30 AND 40
```

```
SELECT ProductID, ProductName FROM Products
WHERE CategoryID IN (1,3,6)
```