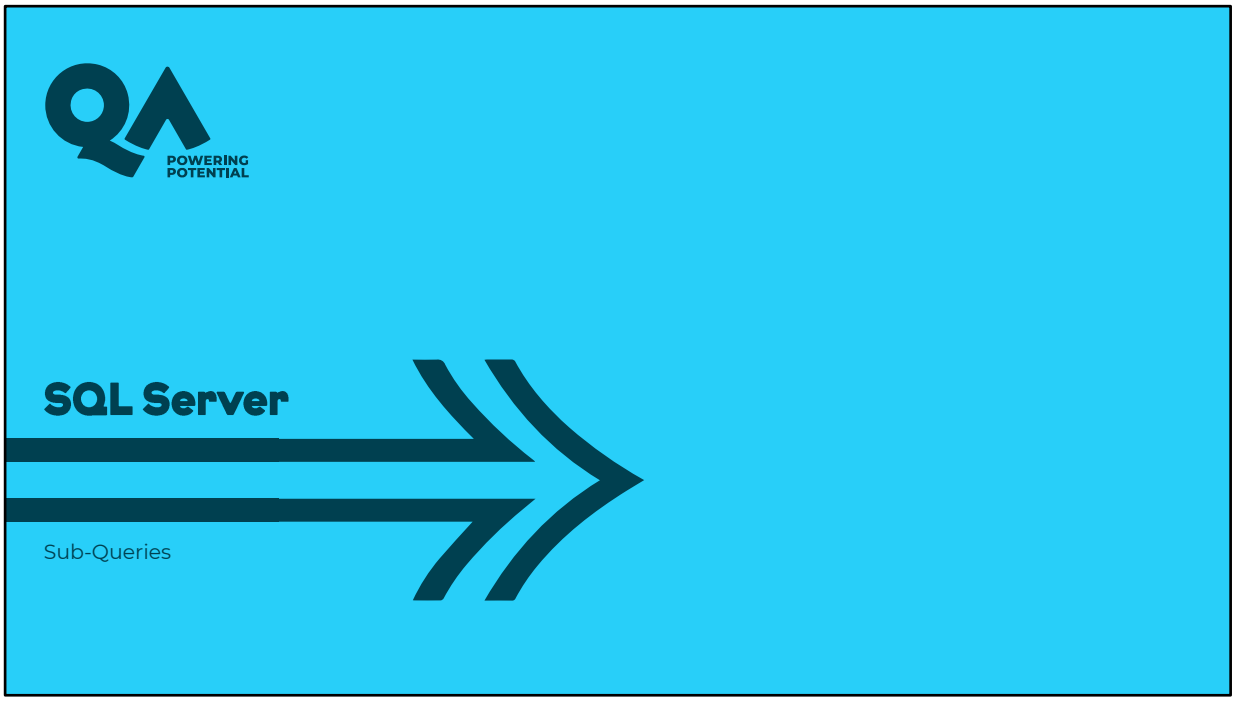
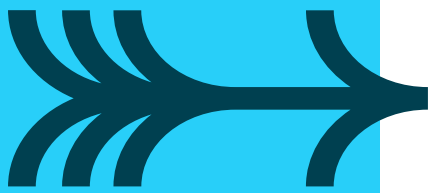


This module will enable the learner to write SQL to display data for more complex requests.
Specifically the learner will be able to:

- Create subqueries using the IN, equals and EXISTS syntax.

Code examples will be used to explain and demonstrate the content and for leaners to imitate. Supporting exercises will provide practice opportunities. At the end of the module there is a short quiz to review the topic.

CONTENTS



This section covers the IN subquery option.
The IN subquery executes an inner query that selects a list of values that are used to match with a field in the WHERE clause of the outer query.
Similarly the NOT IN option executes an inner query that creates a list of values but the outer query only selects records that do not match with any value on the list.

IN

```
SELECT *
FROM salesperson
WHERE dept_no IN
    (
    SELECT dept_no
    FROM dept
    WHERE dept_name LIKE '%Systems'
    );
```

dept_no
2
4

emp_no	first_name	last_name	dept_no	salary	sales_target	county
20	Gertie	Gatling	2	11	14	Hampshire
30	Hattie	Hardegree	2	12	7	Hampshire

The inner query selects a list of dept_no that have a dept_name ending in ‘Systems’. The outer query then selects details of all the salespersons who work in one of the dept_no on the list.
Note this query could use the JOIN syntax to get exactly the same outcome.

Alternative using JOIN

```
-- 6.1.2 Alternative using JOIN
SELECT SP.*
FROM salesperson SP
INNER JOIN dept D
ON SP.dept_no = D.dept_no
WHERE dept_name LIKE '%Systems';
```

- **What does the SQL do?**

The same query using the JOIN syntax.

NOT IN

```
-- 6.1.3 NOT IN
SELECT *
FROM salesperson
WHERE dept_no NOT IN
    (
    SELECT dept_no
    FROM dept
    WHERE dept_name LIKE '%Systems'
    );
```

dept_no
2
4

emp_no	first_name	last_name	dept_no	salary	sales_target	county
10	Ferne	Filmore	1	10	9	Surrey
40	Inge	Isman	3	13	11	London
50	Janene	Jent	3	14	12	Surrey
60	Karena	Kilburn	3	15	13	Surrey

Again the inner query selects a list of dept_no that have a dept_name ending in ‘Systems’. The outer query then selects details of all the salespersons who do not work for one of the dept_no on the list.

Equals

```
-- 6.2.1 Equals
SELECT *
    FROM sale
WHERE order_value =
    (
    SELECT MIN(order_value)
    FROM sale
    );
```

2.55

order_no	emp_no	company_order_no	company_no	contact_code	order_value
500	60	DPF78	4000	TT	2.55

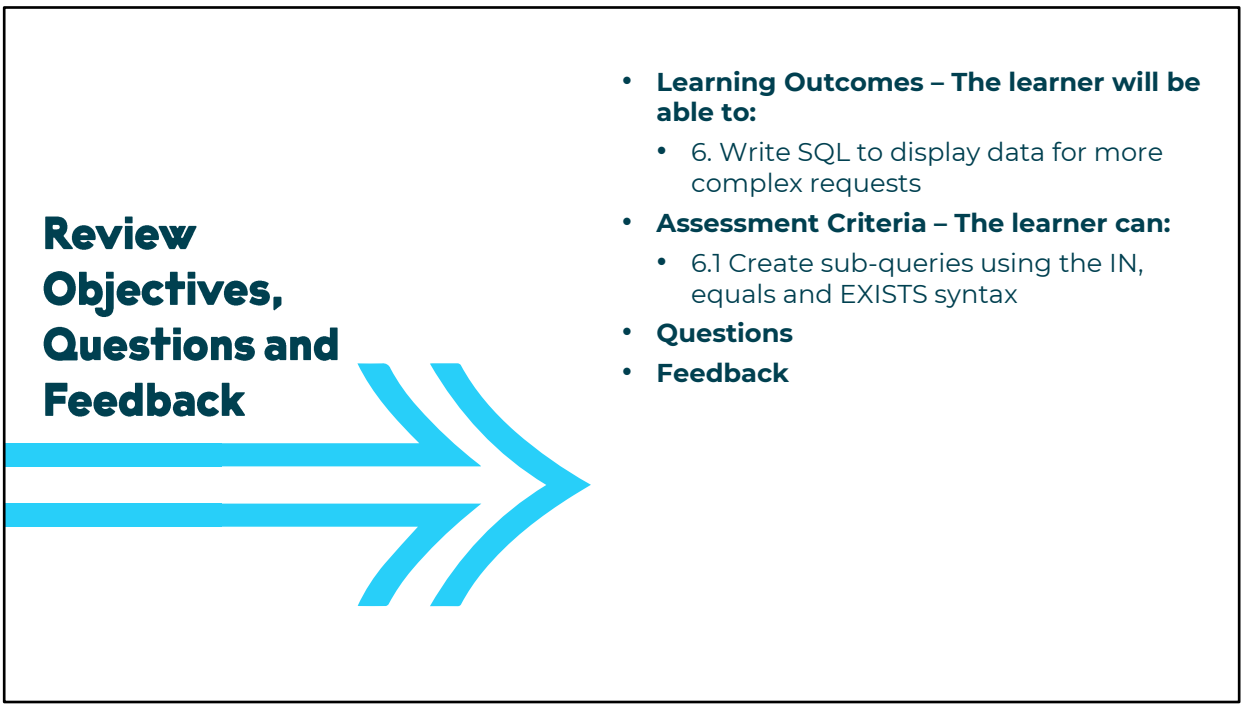
The inner query selects the minimum order_value from the sale table. The outer query then uses that value to find out all the details for the sales, it could be more than one, the have that order_value. This gets around some of the issues of using summarised queries.

NOT Equals

```
-- 6.2.2 Not Equals
SELECT *
    FROM sale
WHERE order_value !=
    (
    SELECT MIN(order_value)
    FROM sale
    );
```

- **What does the SQL do?**

Again the inner query selects the minimum order_value from the sale table. The outer query then uses that value to find out all the details for all the sales that are not equal to that order_value.



Have we achieved the Learning Outcomes for this module? As a learner can you:

- 6. Write SQL to display data for more complex requests.

Also the Assessment Criteria? As a learner can you:

- 6.1 Create subqueries using the IN, equals and EXISTS syntax.

Do you have any further questions on the content covered or feedback on the module?

EXISTS

```
SELECT *
FROM sale
WHERE EXISTS
    (
    SELECT MAX(order_value)
    FROM sale
    HAVING MAX(order_value) > 10
    );
```

27.66

- **What does the SQL do?**

The inner query will select a maximum value if that maximum value is greater than 10. If this occurs the outer query will SELECT the details of all sales otherwise it will do nothing.

NOT EXISTS

```
-- 6.3.2 NOT EXISTS
SELECT *
FROM sale
WHERE NOT EXISTS
    (
    SELECT MAX(order_value)
    FROM sale
    HAVING MAX(order_value) > 30
    );
```

- **What does the SQL do?**

Again the inner query will select a maximum value if that maximum value is greater than 10. If this does not occurs the outer query will SELECT the details of all sales otherwise it will do nothing.

Multiple Level

```
-- 6.4.1 Salesperson with Maximum Number of Orders
SELECT SP.emp_no,SP.last_name,COUNT(S.order_no) AS 'No of Orders'
FROM salesperson SP
JOIN sale S
ON SP.emp_no = S.emp_no
GROUP BY SP.emp_no, SP.last_name
HAVING COUNT(S.order_no) =  (
    SELECT MAX(count_order_no)
    FROM
      (
        SELECT SP.emp_no,COUNT(S.order_no) AS 'count_order_no'
        FROM salesperson SP
        JOIN sale S
        ON SP.emp_no = S.emp_no
        GROUP BY SP.emp_no
      )
        AS maximum_count
    );
```

The technique to developing and understanding multiple level sub-queries is worked out from the inner most sub-query. The results of that subquery become the input for the next outer level and so on until the top or outermost level.
In this example we want to display the salesperson details for the salesperson who has made the most sales.
To start the inner most sub-query counts the number of sales for each salesperson by joining the two tables in a summarised query. Note that the count column is given an alias name of 'count_order_no'. This can be executed independently to test that it gives the expected results; a list of all emp_no and a count of the number of sales they have made.
The next level sub-query selects the maximum value from the column 'count_order_no' from the result set created by the inner sub-query. This the number of sales for the salesperson who made the most sales. Note that this result set is given a table alias name of 'maximum_count'. It is required to make the sub-query work but is not actually used in this example. Again this can be executed to check the expected result; just the maximum number of sales made by any salesperson.
Finally the top level query runs a similar summarised query to the inner query to output the required salesperson details and a count of the number of sales they have made. The HAVING clause will filter the results to just those that the count equals the maximum from the inner sub-query.

POWERING POTENTIAL

Thank you