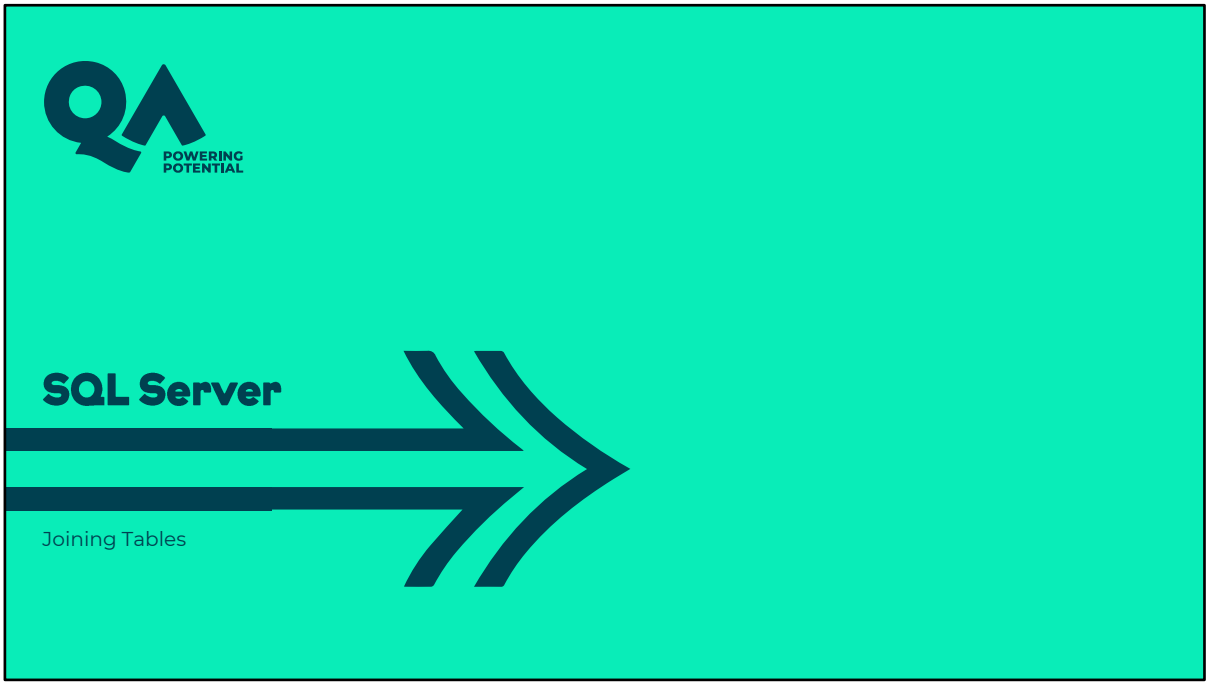


This module will enable the learner to write SQL to display data from multiple tables.
Specifically the learner will be able to:

- Create queries using various INNER, OUTER and FULL JOINs.
- Resolve ambiguous column names by use of qualifiers.

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

This module provides an introduction to JOINs covering both the INNER and OUTER JOIN with a short review including a quiz at the end. We will start by looking at the objectives for the module.

JOIN of Department with Salesperson

Use Tables to visualise

- *"List the department and salesperson details for the departments that have at least one salesperson assigned to it"*

Department

Dept_No (PK)	Dept_name
1	HR
2	IT
3	Finance

Salesperson

Emp_No (PK)	Emp_Name	Dept_No (FK)
10	John	1
20	Jack	2
30	James	2
40	Jim	NULL

INNER JOIN

Dept_No	Dept_name	Emp_No	Emp_Name
1	HR	10	John
2	IT	20	Jack
2	IT	30	James

Another way to understand joins is to use the relational database tables to demonstrate the link between the tables.
The default JOIN is an INNER JOIN. From the department table we want to list the department and salesperson details for the departments that have at least one salesperson assigned to it.
Typically the link is made through the Primary Key / Foreign Key relationship that is defined for given tables when the database is set up. In this example the Primary Key of department is dept_no which is linked to the Foreign Key dept_no on salesperson. This is the basis of relational database design and allows data that has been normalised into appropriate tables to be joined back together to answer specific queries.
The records displayed by the INNER JOIN are just the departments with at least one salesperson and the details of those salespersons.

JOIN of Department with Salesperson

```
SELECT
    D.dept_name,
    D.dept_no AS 'D Dept No',
    SP.dept_no AS 'SP Dept No',
    SP.emp_no,  SP.first_name,
FROM dept D
INNER JOIN salesperson SP
ON D.dept_no = SP.dept_no
ORDER BY D.dept_name,SP.emp_no;
```

dept_name	D Dept No	SP Dept No	emp_no	first_name
Animal Products	1	1	10	Ferne
Business Systems	2	2	20	Gertie
Business Systems	2	2	30	Hattie
Credit Control	3	3	40	Inge
Credit Control	3	3	50	Janene
Credit Control	3	3	60	Karena

The basic format of an INNER JOIN is displayed above.
A SELECT specifies that columns from the two tables that are required to be displayed. This can include expressions and alias names as with SELECTs on single tables. Note the prefix in front of each column name; more later.
Next, specify the tables that are being joined.
The FROM clause specifies the first or left hand table that can be thought of as the table that is driving the query.
The INNER JOIN clause specifies the second or right hand table to be joined to.
Note the default JOIN is an INNER JOIN so the INNER does not need to be coded.
In this case we want to go through all departments and select those that have a salesperson assigned to it. Therefore we have specified dept as the ‘from’ table and salesperson as the ‘joined’ table.
Next, we need to specify the columns that will be used to link the two tables together. This will normally be the Primary Key of the ‘from’ table and the Foreign Key of the ‘joined’ table but it does not have to be. In this case we have linked the tables using dept_no. If the data types are compatible SQL allows any column to be linked to any other column. This may be required for specific joins.
Finally the data selected by the join can be sorted by specifying the ORDER BY clause. In this case by dept_name and then emp_no.

JOIN of Salesperson with Department

```
SELECT
    SP.emp_no,   SP.first_name, SP.last_name,
    SP.dept_no AS 'SP Dept No',
    D.dept_no AS 'D Dept No',
    D.dept_name
FROM salesperson SP
INNER JOIN dept D
ON SP.dept_no = D.dept_no
ORDER BY SP.emp_no;
```

emp_no	first_name	last_name	SP Dept No	D Dept No	dept_name
10	Ferne	Filmore	1	1	Animal Products
20	Gertie	Gatling	2	2	Business Systems
30	Hattie	Hardegree	2	2	Business Systems
40	Inge	Isman	3	3	Credit Control
50	Janene	Jent	3	3	Credit Control
60	Karena	Kilburn	3	3	Credit Control

When coding an INNER JOIN the same data will be selected regardless of order in which tables are specified.
In this example we have switched salesperson to be the 'from' table and dept to be the joined table with the same joining columns of dept_no. As the query lists salespersons with their corresponding department details it makes logically sense to put the salesperson table first but is not necessary.

JOIN of more than Two Tables

```
SELECT
    D.dept_name,
    SP.last_name,
    S.order_no
FROM dept D
INNER JOIN salesperson SP
ON D.dept_no = SP.dept_no
INNER JOIN sale S
ON SP.emp_no = S.emp_no
ORDER BY D.dept_name,SP.last_name,S.order_no;
```

dept_name	last_name	order_no
Animal Products	Filmore	400
Animal Products	Filmore	700
Credit Control	Jent	600
Credit Control	Kilburn	100
Credit Control	Kilburn	200
Credit Control	Kilburn	300

It is possible to JOIN two or more tables.
The INNER JOIN and ON clauses need to repeated accordingly for the relevant joining tables and linking columns.
This query lists all the department names, the last name of the salesperson in each department and the order numbers each salesperson has taken. The data is selected from the dept, salesperson and sale tables. The dept and salesperson tables are linked by the Primary and Foreign Key of dept_no. The salesperson and sales table are linked by the Primary and Foreign Key of emp_no. The joined results are then sorted in department name, last name and order_no order.

Composite JOIN

```
SELECT
    S.order_no,
    S.company_no,
    S.contact_code,
    C.company_no
FROM sale S
INNER JOIN contact C
ON S.company_no = C.company_no
AND S.contact_code = C.contact_code
ORDER BY S.order_no;
```

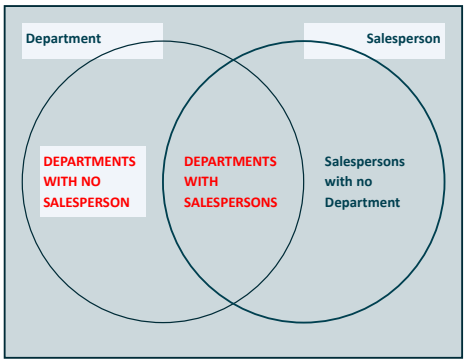
order_no	company_no	contact_code	company_no
100	1000	MM	1000
200	3000	QQ	3000
300	2000	OO	2000
400	1000	MM	1000

Sometimes a join will require more than one column to link two tables together. For example when the Primary Key and Foreign Key are a Composite Key that consists of two or more columns.
This query joins the sale and contact tables. The Primary Key of contact is a composite key that consists of the columns company_no and contact code. These need to be linked to the corresponding Foreign Key on sale.
An additional AND clause needs to be added to the ON clause for each additional column in the Composite Key.
The query displays the order_no, company_no and contact_code for all sales with the corresponding name of that contact from the contact table.
If only the company_no is coded in the ON clause the results will display all the contacts at each company for each sale regardless of which contact made the actual sale.
If only the contact_code is coded in the ON clause the results will display all the contacts with the same code regardless of which company they were from for each sale.

In both cases this will give inaccurate results.

OUTER JOIN

- ***List the department and salesperson details for all departments including those that do not have a salesperson assigned to it***



Again a way to understand joins is to visualise the intersection or overlap of two or more tables using a Venn diagram.
For the default or INNER JOIN the only records displayed those for departments with at least one salesperson. If we also want to see the records for departments without any salesperson we have to use an OUTER JOIN. This is illustrated by the left hand side and intersection with the upper case text in red. We are not interested in salespersons that are not assigned to a department.
The link is made through the Primary Key / Foreign Key relationship that is defined for given tables when the database is set up. In this example the Primary Key of department is dept_no which is linked to the Foreign Key dept_no on salesperson. This is the basis of relational database design and allows data that has been normalised into appropriate tables to be joined back together to answer specific queries.
The records in the intersection are the departments with at least one salesperson in them. The records in the left hand side are department with no salesperson in them.

OUTER JOIN of all Departments – Tables

- *List the department and salesperson details for all departments including those that do not have a salesperson assigned to it*

Department

Dept_No (PK)	Dept_name
1	HR
2	IT
3	Finance

Salesperson

Emp_No (PK)	Emp_Name	Dept_No (FK)
10	John	1
20	Jack	2
30	James	2
40	Jim	NULL

OUTER JOIN

Dept_No	Dept_name	Emp_No	Emp_Name
1	HR	10	John
2	IT	20	Jack
2	IT	30	James
3	Finance	NULL	NULL

As before another way to understand joins is to use the relational database tables to demonstrate the link between the tables.
The default JOIN is an INNER JOIN. From the department table we want to list the department and salesperson details for the departments that have at least one salesperson assigned to it. If we also want to see the records for departments without any salesperson we have to use an OUTER JOIN.

Typically the link is made through the Primary Key / Foreign Key relationship that is defined for given tables when the database is set up. In this example the Primary Key of department is dept_no which is linked to the Foreign Key dept_no on salesperson. This is the basis of relational database design and allows data that has been normalised into appropriate tables to be joined back together to answer specific queries.
The records displayed by the OUTER JOIN are all the departments with the details of salespersons where they are assigned to a department and NULL values where there are no salespersons assigned to a department.

OUTER JOIN of all Departments

```
SELECT
    D.dept_name,
    SP.first_name,
    SP.last_name
FROM dept D
LEFT OUTER JOIN salesperson SP
ON D.dept_no = SP.dept_no
ORDER BY D.dept_name,SP.first_name,SP.last_name;
```

dept_name	first_name	last_name
Animal Products	Ferne	Filmore
Business Systems	Gertie	Gatling
Business Systems	Hattie	Hardegree
Credit Control	Inge	Isman
Credit Control	Janene	Jent
Credit Control	Karena	Kilburn
Desktop Systems	NULL	NULL
Electrical Repairs	NULL	NULL

The basic format of an OUTER JOIN is the same as an INNER JOIN but with the word INNER if included, remember it is a default, replaced by OUTER.
The LEFT indicates that we want to display records in the 'from' table that do not link with any records in the 'joined' table. Think of left hand table in the Venn diagram.
This query displays the department name of all departments. Where there is one or more salespersons in that department their first and last names are displayed. If there are no salespersons in the department NULL values are displayed.

LEFT v RIGHT OUTER JOIN

```
-- 4.2.5 LEFT v RIGHT OUTER JOIN
SELECT
    D.dept_name,
    SP.first_name,
    SP.last_name
FROM salesperson SP
RIGHT OUTER JOIN dept D
ON SP.dept_no = D.dept_no
ORDER BY D.dept_name,SP.first_name,SP.last_name;
```

What does the SQL do?

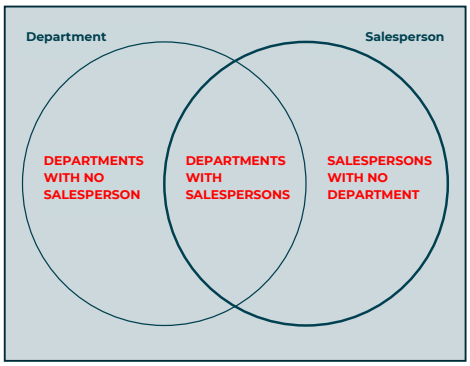
- Calculate the expected result
- Switch LEFT with RIGHT and the 'from' and 'joined' tables

As mentioned previously a LEFT and RIGHT OUTER JOIN produce the same results if the 'from' and 'joined' tables are switched. Best practice is to code the tables in the order that makes most sense for the requirements of the query and chose the appropriate LEFT or RIGHT OUTER JOIN.

FULL OUTER JOIN

Use a Venn diagram to visualise

- *"List the department and salesperson details for all departments including those that do not have a salesperson assigned to it and for all salespersons including those that do not have a department assigned to them"*



Finally we might want to see all records from the joined tables regardless of whether is a link between them. This is known as a FULL OUTER JOIN.
This query will display departments with no salespersons, department with salesperson and salesperson not assigned to a department.

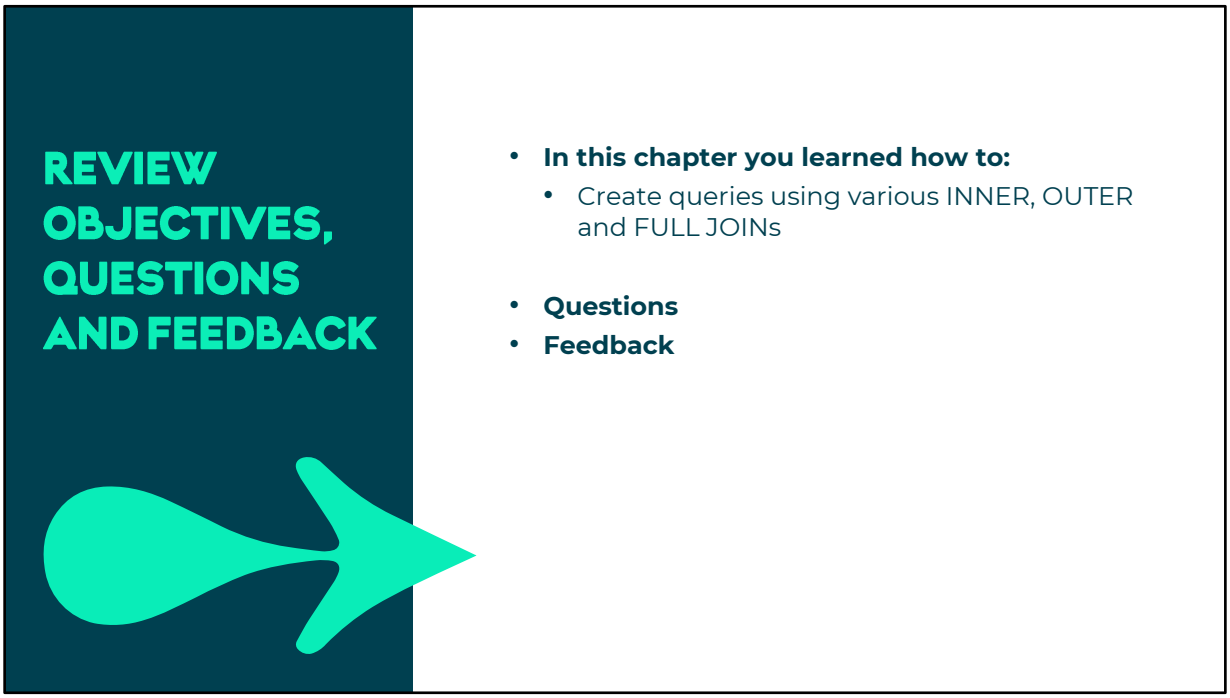
FULL OUTER JOIN

```
-- 4.2.6 FULL OUTER JOIN
SELECT
    D.dept_name,
    SP.first_name,
    SP.last_name
FROM dept D
FULL OUTER JOIN salesperson SP
ON D.dept_no = SP.dept_no
ORDER BY dept_name,first_name,last_name;
```

What does the SQL do?

- Calculate the expected result
- Switch the 'from' and 'joined' tables in the query

SQL Server does support a FULL OUTER JOIN; Other SQL versions, including MySQL so not.
Instead a FULL OUTER JOIN is achieved by a UNION of a LEFT and RIGHT OUTER JOIN.



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

- 4. Write SQL to display data from multiple tables.

Also the Assessment Criteria? As a learner can you:

- 4.1 Create queries using various INNER, OUTER and FULL JOINs.
- 4.2 Resolve ambiguous column names by use of qualifiers.

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

Self JOIN

```
-- 4.1.5 Self JOIN
SELECT
    SP1.county,
    SP1.last_name,
    SP2.last_name
FROM salesperson SP1
INNER JOIN salesperson SP2
ON SP1.county = SP2.county
WHERE SP1.emp_no > SP2.emp_no
ORDER BY SP1.county,SP1.last_name;
```

What does the SQL do?

- Calculate the expected result
- Why is the WHERE clause required?

It possible to join a table with itself. The table short names or prefixes are required to determine which data comes form which table. Other than that the JOIN follows the same format.
This query displays the counties and names of salesperson where there is more than one salesperson in the same county. Look at the records on the table salesperson to calculate the expected result and compare with your actual result.
The WHERE clause eliminates the records when it is the same salesperson in the same county and the duplicate record for each combination of salesperson in the same county.

POWERING POTENTIAL

Thank you