Systems Design and Databases (CIS1018-N)

Week 6

Querying with Multiple Tables
(Joins)

Teaching Team

Module Leader & Lecturer: Dr Yar Muhammad
Email: Yar.Muhammad@tees.ac.uk
Office: G0.39 (Greig Building)

Tutor:

- Dr Mengda He
- Mr Mansha Nawaz
- Mr Vishalkumar Thakor

Academic Hub Time Slots, Room IT1.13:
Yar Muhammad
Monday 10:00 - 11:00 and Tuesday 13:00 - 14:00

Mengda He
Wednesdays 1-2 pm and Fridays 11 am - 12 pm

- See Blackboard Ultra for online materials: https://bb.tees.ac.uk/

Lectures & IT Labs

Lectures – Dr Yar Muhammad	Tuesdays @ 2-3 pm	Thursdays @ 1-2 pm
Week 1 – Week 12	CL1.87	

Tutor – Thursday	IT Lab Session Room #: IT2.42
Mr Mansha Nawaz M.Nawaz@tees.ac.uk	Time: 3 – 5 pm

Tutor – Friday	IT Lab Session Room #: OL3
Dr Yar Muhammad Yar.Muhammad@tees.ac.uk	Time: 9 – 11 am & 11 am – 1 pm
Dr Mengda He M.He@tees.ac.uk	Time: 9 – 11 am
Mr Vishalkumar Thakor V.Thakor@tees.ac.uk	Time: 11 am – 1 pm & 1 – 3 pm
Mr Mansha Nawaz M.Nawaz@tees.ac.uk	Time: 1 – 3 pm

Schedule 1/3

Systems Design and Databases CIS1018-N
Weekly Plan for the Activities

Week	Lecturer	Lecture Demo	Lab Exercises & Solutions	ICA Tasks:
01	• Module Introduction, • System Design, • Introduction Databases (DDL, DML, DCL, TCL)	• Requirement List & • MoSCoW Wireframe Design & Templates, • User Stories	• Team Setup, • Hands-on to collect/pick the Requirements from MoSCoW and write Writing User stories on each • **Tutorial 1**	Requirements List & MosCOW, User stories
02	• UML and UML Tool,	• Use Case Diagrams from Requirements List and Wireframe	• Hands-on Use Case Diagrams Activities • **Tutorial 2**	Each Wireframe has associated Use Case Activity Deadline for Team Setup is Week # 2, by Friday 07/10/2022 before 4pm
03	• Sequence Diagrams	• Class Diagrams	• Hands-on Sequence & Class Diagrams Activities • **Tutorial 3**	Each Wireframe has associated Sequence and Class Diagrams
04	• Entity Relationship Diagrams (ERD) A Data Modelling Case Tool for Relational Databases	• Introduction to SQL Server • Walk-through: SQL Quick Guide 1 - How to use SSMS to build Databases	• **Tutorial 4** • Lab Resources: SQL Quick Guide 1	Each Wireframe has associated Class Diagram

Systems Design - UML

Analysis

Design

Schedule 2/3

Databases - TSQL

Week	Lecturer	Lecture Demo	Lab Exercises & Solutions	ICA Tasks:
05	• Querying with Select	**Demo A** – Writing Simple SELECT Statements **Demo B/C** – Eliminating Duplicates with DISTINCT **Demo D** - Writing Simple CASE	• TSQL-Mod03 Lab-Exercise 1-4 • **Tutorial 5**	SQL Task A: TSQL03 Querying with Select • Writing Simple SELECT Statements • Eliminating Duplicates with DISTINCT • Using Column and Table Aliases • Writing Simple CASE Expressions
06	• Querying with Multiple Tables	**Demo B** – Relating 2 or more tables – Joins & Joining multiple tables – inner, outer and cross.	• TSQL-Mod04 Exercise 1-5 • **Tutorial 6**	SQL Task B: TSQL04 – Querying with Multiple Tables • Relating 2 or more tables – Joins • Joining multiple tables – inner, outer and cross.
07	• Sorting and Filtering Data	**Demo A** – Sort with ORDER BY **Demo B** – Filter with WHERE Clause **Demo C** – Filtering with Top OffsetFetch **Demo D** – Handling NULL	• TSQL-Mod05 Exercise 1 – 4 • **Tutorial 7**	SQL Task C: TSQL05 – Sort and Filtering Data • Sort with Order By • Filter with Where By • Filter with top offsetfetch • Handling Nulls
Submission ICA 1 (Group Submission) -> Deadline is Wednesday 16/11/2022 before 4pm				
08	• Working with SQL Server Data	**Demo A** - Conversion in a Query **Demo B** - collation in a query **Demo C** - date and time functions	• TSQL-Mod06 Exercise 1 – 4 • **Tutorial 8**	SQL Task D: TSQL06 – Working with SQL Server Data • Conversion in a Query • collation in a query • date and time functions

Schedule 3/3

Databases - TSQL

09	• Using DML to modify Data	**Demo A** - Adding Data to Tables **Demo B** - Modifying and Removing Data **Demo C** - Generating Automatic Column Values	• TSQL-Mod07 Exercise 1 – 2 • **Tutorial 9**	SQL Task E: TSQL07– Using DML to Modify Data • Adding Data to Tables • Modifying and Removing Data • Generating Automatic Column Values
10	• Using built in Functions	**Demo A** – Scalar Functions **Demo B** – Cast Functions Demo C – If Functions **Demo D** – IsNull Functions	• TSQL-Mod08 Exercise 1 – 3 • **Tutorial 10**	SQL Task F: TSQL08– Using Built-In Functions • Writing Queries with Built-In Functions • Using Conversion Functions • Using Logical Functions • Using Functions to Work with NULL
11	• Walk through SQL Quick Guide 2 - Create a Tables and Relationships via SSMS GUI	• Walk through: • SQL Quick Guide 3 - Create Query, View through Designer	Hands-on: • SQL Server Quick Guide 2	SQL Server – Introduction to SQL Server and SSMS
12	Support	Support	Hands-on: • SQL Server Quick Guide 3	SQL Server – Introduction to SQL Server and SSMS
Submission ICA 2 (Individual Submission) -> Deadline is Wednesday 11/01/2023 before 4pm				

Joins (SQL Server) 1/2

- Joins (SQL Server) - Microsoft Docs
- Joins **indicate how SQL Server should use data from one table to select the rows in another table**.
- A join condition defines the way two tables are related in a query by: Specifying the column from each table to be used for the join.
- SQL Server performs sort, intersect, union, and difference operations using in-memory sorting and hash join technology.

Joins (SQL Server) 2/2

- Using this type of query plan, SQL Server supports vertical table partitioning.
- SQL Server implements logical join operations, as determined by Transact-SQL syntax:
  - Inner join
  - Left outer join
  - Right outer join
  - Full outer join
  - Cross join

Understanding Joins

- The FROM Clause and Virtual Tables
- Join Terminology: Cartesian Product
- Overview of Join Types
- T-SQL Syntax Choices
- Demonstration: Understanding Joins

The FROM Clause and Virtual Tables

- FROM clause determines source tables to be used in SELECT statement
- FROM clause can contain tables and operators
- Result set of FROM clause is virtual table
  - Subsequent logical operations in SELECT statement consume this virtual table
- FROM clause can establish table aliases for use by subsequent phases of query

Join Terminology: Cartesian Product

- Characteristics of a Cartesian product
  - Output or intermediate result of FROM clause
  - **Combine all possible combinations of two sets**
  - In T-SQL queries, usually not desired
    - Special case: table of numbers

Name
Davis
Funk
King

×

Product
Alice Mutton
Crab Meat
Ipoh Coffee

=

Name	Product
Davis	Alice Mutton
Davis	Crab Meat
Davis	Ipoh Coffee
Funk	Alice Mutton
Funk	Crab Meat
Funk	Ipoh Coffee
King	Alice Mutton
King	Crab Meat
King	Ipoh Coffee

Overview of Join Types

- Join types in FROM clauses specify the operations performed on the virtual table:

Join Type	Description
Cross	Combines all rows in both tables (creates Cartesian product)
Inner	Starts with Cartesian product; applies filter to match rows between tables based on predicate
Outer	Starts with Cartesian product; all rows from designated table preserved, matching rows from other table retrieved. Additional NULLs inserted as placeholders

T-SQL Syntax Choices

- ANSI SQL-92
  - Tables joined by JOIN operator in FROM Clause

```
SELECT ...
FROM   Table1 JOIN Table2
       ON <on_predicate>
```

- ANSI SQL-89
  - Tables joined by commas in FROM Clause
    - Not recommended: accidental Cartesian products!

```
SELECT ...
FROM   Table1, Table2
WHERE  <where_predicate>
```

Demonstration A with TSQL: Understanding Joins

In this demonstration, you will see how to Use joins

```
-- Demo Queries are below
USE TSQL;

SELECT c.companyname, o.orderdate
FROM Sales.Customers AS c, Sales.Orders AS o
WHERE c.custid = o.custid;

SELECT c.companyname, o.orderdate
FROM Sales.Customers AS c, Sales.Orders AS o;

SELECT c.companyname, o.orderdate
FROM Sales.Customers AS c
JOIN Sales.Orders AS o ON c.custid = o.custid;

-- Note that the ON clause is deliberately omitted
-- to cause an error, showing the protection
-- against accidental Cartesian products
--THIS WILL INTENTIONALLY CAUSE AN ERROR
SELECT c.companyname, o.orderdate
FROM Sales.Customers AS c JOIN Sales.Orders AS o;
--ON c.custid = o.custid
```

Results | Messages

	companyname	orderdate
1	Customer ENQZT	2006-07-04 00:00:00.000
2	Customer FAPSM	2006-07-05 00:00:00.000
3	Customer IBVRG	2006-07-08 00:00:00.000
4	Customer NRCSK	2006-07-08 00:00:00.000
5	Customer SFOGW	2006-07-09 00:00:00.000

Results | Messages

	companyname	orderdate
75...	Customer ZRNDE	2008-05-04 00:00:00.000
75...	Customer ZRNDE	2008-05-04 00:00:00.000
75...	Customer ZRNDE	2008-05-04 00:00:00.000
75...	Customer ZRNDE	2008-05-05 00:00:00.000
75...	Customer ZRNDE	2008-05-05 00:00:00.000
75...	Customer ZRNDE	2008-05-05 00:00:00.000
75...	Customer ZRNDE	2008-05-05 00:00:00.000
75...	Customer ZRNDE	2008-05-06 00:00:00.000
75...	Customer ZRNDE	2008-05-06 00:00:00.000

Results | Messages

	companyname	orderdate
1	Customer ENQZT	2006-07-04 00:00:00.000
2	Customer FAPSM	2006-07-05 00:00:00.000
3	Customer IBVRG	2006-07-08 00:00:00.000
4	Customer NRCSK	2006-07-08 00:00:00.000
5	Customer SFOGW	2006-07-09 00:00:00.000
6	Customer IBVRG	2006-07-10 00:00:00.000
7	Customer WNMAF	2006-07-11 00:00:00.000
8	Customer CCKOT	2006-07-12 00:00:00.000

Messages

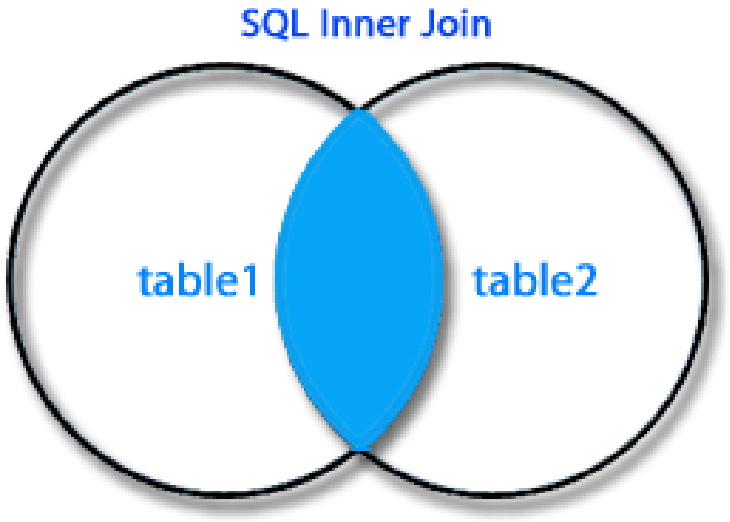
```
Msg 102, Level 15, State 1, Line 46
Incorrect syntax near ';'.
```

Querying with Inner Joins

- Understanding Inner Joins
- Inner Join Syntax
- Inner Join Examples
- Demonstration: Querying with Inner Joins

Inner Join in SQL

- The INNER JOIN selects all rows from both participating tables as long as there is a match between the columns.
- An SQL INNER JOIN is same as JOIN clause, combining rows from two or more tables
- The INNER JOIN in SQL joins two tables according to the matching of a certain criteria using a comparison operator.



Syntax:

```
SELECT *
FROM table1 INNER JOIN table2
ON table1.column_name = table2.column_name;
```

OR

```
SELECT *
FROM table1 JOIN table2
ON table1.column_name = table2.column_name;
```

Understanding Inner Joins

- Returns only rows where a match is found in both input tables
- Matches rows based on attributes supplied in predicate
  - ON clause in SQL-92 syntax (preferred)
  - WHERE clause in SQL-89 syntax
- Why filter in ON clause?
  - Logical separation between filtering for purposes of join and filtering results in WHERE
  - Typically no difference to query optimizer
- If join predicate operator is =, also known as equi-join

Inner Join Syntax

- List tables in FROM Clause separated by JOIN operator
- Table aliases preferred
- Table order does not matter

```
FROM t1 JOIN t2
    ON t1.column = t2.column
```

```
SELECT o.orderid,
       o.orderdate,
       od.productid,
       od.unitprice,
       od.qty
FROM Sales.Orders AS o
     JOIN Sales.OrderDetails AS od
     ON o.orderid = od.orderid;
```

Inner Join Examples

- Join based on single matching attribute

```
SELECT ...
FROM Production.Categories AS C
    JOIN Production.Products AS P
        ON C.categoryid = P.categoryid;
```

- Join based on multiple matching attributes (composite join)

```
-- List cities and countries where both --
customers and employees live
SELECT DISTINCT e.city, e.country
FROM Sales.Customers AS c
JOIN    HR.Employees AS e
    ON c.city = e.city AND
    c.country = e.country;
```

Demonstration B with TSQL: Querying with Inner Joins

In this demonstration, you will see how to Use inner joins

```
-- Demo Queries are below
USE TSQL;

SELECT c.categoryid, c.categoryname, p.productid,
p.productname
FROM Production.Categories AS c
JOIN Production.Products AS p
ON c.categoryid = p.categoryid;


SELECT e.city, e.country
FROM Sales.Customers AS c
JOIN HR.Employees AS e
ON c.city = e.city AND c.country = e.country;

SELECT DISTINCT  e.city, e.country
FROM Sales.Customers AS c
JOIN HR.Employees AS e
ON c.city = e.city AND c.country = e.country;

from a third table have been commented out to join
SELECT c.custid, c.companyname, o.orderid,
o.orderdate-- , od.productid, od.qty
FROM Sales.Customers AS c
JOIN Sales.Orders AS o
ON c.custid = o.custid;
-- JOIN Sales.OrderDetails od
-- ON o.orderid = od.orderid;
```

Results | Messages

	categoryid	categoryname	productid	productname
1	1	Beverages	1	Product HHYDP
2	1	Beverages	2	Product RECZE
3	2	Condiments	3	Product IMEHJ
4	2	Condiments	4	Product KSBRM
5	2	Condiments	5	Product EPEIM
...				
73	8	Seafood	73	Product WEUJZ
74	7	Produce	74	Product BKAZJ
75	1	Beverages	75	Product BWRLG
76	1	Beverages	76	Product JYGFE
77	2	Condiments	77	Product LUNZZ

Results | Messages

	city	country
1	Seattle	USA
2	Kirkland	USA
3	London	UK
4	London	UK
5	London	UK
...		
24	London	UK
25	London	UK
26	London	UK
27	London	UK

Results | Messages

	city	country
1	Kirkland	USA
2	London	UK
3	Seattle	USA

```
-- Demo Queries are below
USE TSQL;

-- 2155 rows will be returned. Why?
SELECT c.custid, c.companyname, o.orderid,
o.orderdate, od.productid, od.qty
FROM Sales.Customers AS c
JOIN Sales.Orders AS o
ON c.custid = o.custid
JOIN Sales.OrderDetails od
ON o.orderid = od.orderid;
```

	custid	companyname	orderid	orderdate	productid	qty
1	85	Customer ENQZT	10248	2006-07-04 00:00:00.000	11	12
2	85	Customer ENQZT	10248	2006-07-04 00:00:00.000	42	10
3	85	Customer ENQZT	10248	2006-07-04 00:00:00.000	72	5
4	79	Customer FAPSM	10249	2006-07-05 00:00:00.000	14	9
5	79	Customer FAPSM	10249	2006-07-05 00:00:00.000	51	40
...						
21...	65	Customer NYUHS	11077	2008-05-06 00:00:00.000	73	2
21...	65	Customer NYUHS	11077	2008-05-06 00:00:00.000	75	4
21...	65	Customer NYUHS	11077	2008-05-06 00:00:00.000	77	2

Results | Messages

	custid	companyname	orderid	orderdate
1	85	Customer ENQZT	10248	2006-07-04 00:00:00.000
2	79	Customer FAPSM	10249	2006-07-05 00:00:00.000
3	34	Customer IBVRG	10250	2006-07-08 00:00:00.000
4	84	Customer NRCSK	10251	2006-07-08 00:00:00.000
5	76	Customer SFOGW	10252	2006-07-09 00:00:00.000
...				
827	73	Customer JMIKW	11074	2008-05-06 00:00:00.000
828	68	Customer CCKOT	11075	2008-05-06 00:00:00.000
829	9	Customer RTXGC	11076	2008-05-06 00:00:00.000
830	65	Customer NYUHS	11077	2008-05-06 00:00:00.000

Querying with Outer Joins

- Understanding Outer Joins
- Outer Join Syntax
- Outer Join Examples
- Demonstration: Querying with Outer Joins

Outer Join in SQL

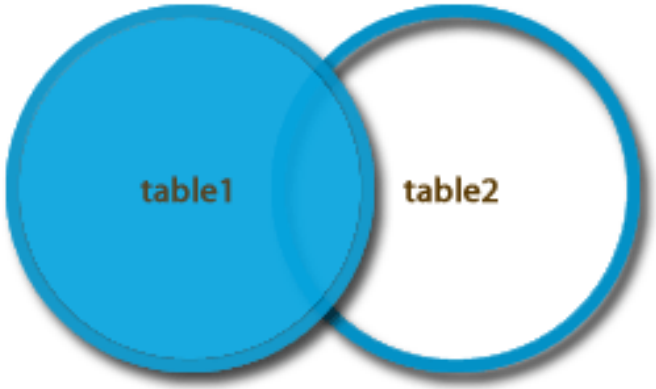
- The SQL OUTER JOIN returns all rows from both the participating tables which satisfy the join condition along with rows which do not satisfy the join condition. The SQL OUTER JOIN operator (+) is used only on one side of the join condition only.

- **Syntax:**

```
Select *
FROM table1, table2
WHERE conditions [+];
```

Outer Join in SQL – Left Join

- The SQL LEFT JOIN (specified with the keywords LEFT JOIN and ON) joins two tables and fetches all matching rows of two tables for which the SQL-expression is true, plus rows from the first table that do not match any row in the second table

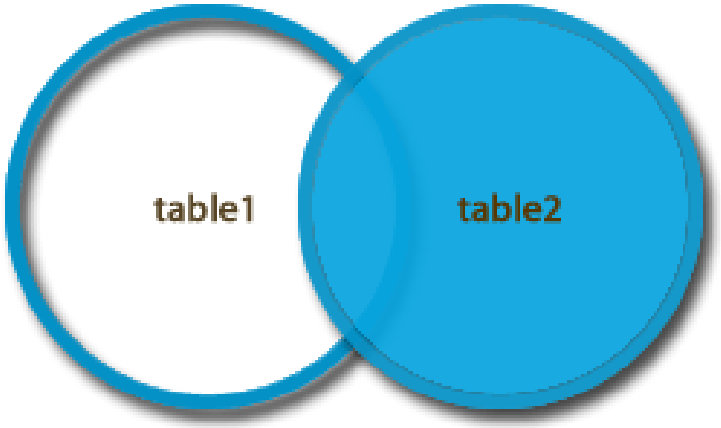


- **Syntax:**

```
SELECT *
FROM table1
LEFT [ OUTER ] JOIN table2
ON table1.column_name=table2.column_name;
```

Outer Join in SQL – Right Join

- The SQL RIGHT JOIN, joins two tables and fetches rows based on a condition, which is matching in both the tables (before and after the JOIN clause mentioned in the syntax below) , and the unmatched rows will also be available from the table written after the JOIN clause (mentioned in the syntax below)

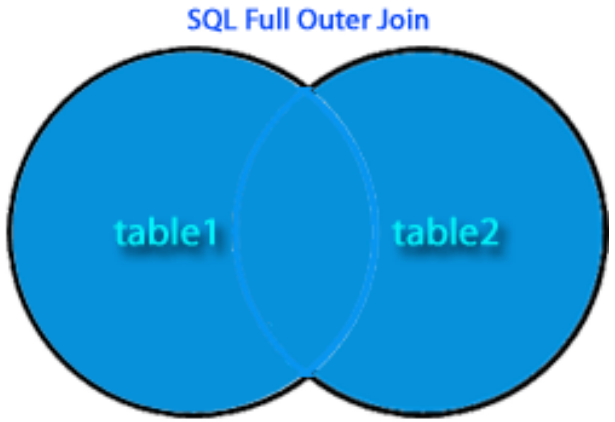


- **Syntax:**

```
SELECT *
FROM table1
RIGHT [ OUTER ] JOIN table2
ON table1.column_name=table2.column_name;
```

Full Outer Join in SQL

- In SQL the FULL OUTER JOIN combines the results of both left and right outer joins and returns all (matched or unmatched) rows from the tables on both sides of the join clause.



- **Syntax:**

```
SELECT *
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

Understanding Outer Joins

- Returns all rows from one table and any matching rows from second table
- One table's rows are "preserved"
  - Designated with LEFT, RIGHT, FULL keyword
  - All rows from preserved table output to result set
- Matches from other table retrieved
- Additional rows added to results for nonmatched rows
  - NULLs added in places where attributes do not match
- Example: return all customers and, for those who have placed orders, return order information; customers without matching orders will display NULL for order details

Outer Join Syntax

- Return all rows from first table, only matches from second:

```
FROM t1 LEFT OUTER JOIN t2 ON
    t1.col = t2.col
```

- Return all rows from second table, only matches from first:

```
FROM t1 RIGHT OUTER JOIN t2 ON
    t1.col = t2.col
```

- Return only rows from first table, with no match in second:

```
FROM t1 LEFT OUTER JOIN t2 ON
    t1.col = t2.col
WHERE     t2.col IS NULL
```

Outer Join Examples

- All customers with order details if present:

```
SELECT c.custid, c.contactname, o.orderid,
o.orderdate
FROM Sales.Customers AS C
LEFT OUTER JOIN Sales.Orders AS O
    ON c.custid = o.custid;
```

- Customers who did not place orders:

```
SELECT c.custid, c.contactname, o.orderid,
o.orderdate
FROM Sales.Customers AS C LEFT OUTER JOIN
Sales.Orders AS O
    ON c.custid = o.custid
WHERE o.orderid IS NULL;
```

Demonstration C AdventureWorksLT2019: Querying with Outer Joins

In this demonstration, you will see how to Use outer joins

```sql
-- Demo Queries are below
USE AdventureWorks2019;


-- to show only matching customers and orders
SELECT c.CustomerID, soh.SalesOrderID
FROM Sales.Customer c JOIN Sales.SalesOrderHeader soh
ON c.CustomerID = soh.CustomerID;

SELECT *
FROM Sales.Customer c LEFT OUTER JOIN
Sales.SalesOrderHeader soh
ON c.CustomerID = soh.CustomerID;
```

Results | Messages

	CustomerID	PersonID	StoreID	TerritoryID	AccountNumber	rowguid
1	29825	1045	1046	5	AW00029825	0C6B3FA2-F1D
2	29672	721	722	5	AW00029672	8EE350F7-ED5
3	29734	851	852	6	AW00029734	789F3CEE-056

...

Results | Messages

(32166 rows affected)

Results | Messages

	CustomerID	SalesOrderID
1	11000	43793
2	11000	51522
3	11000	57418
4	11001	43767
5	11001	51493

...

31...	30118	53480
31...	30118	58928
31...	30118	65221
31...	30118	71803

Results | Messages

(31465 rows affected)

```sql
-- Demo Queries are below
USE TSQL;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
LEFT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
LEFT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid
WHERE o.orderid IS NULL;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
RIGHT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
RIGHT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid
WHERE c.custid IS NULL;
```

Results | Messages

	custid	companyname	orderid	orderdate
1	85	Customer ENQZT	10248	2006-07-04 00:00:00.000
2	79	Customer FAPSM	10249	2006-07-05 00:00:00.000
3	34	Customer IBVRG	10250	2006-07-08 00:00:00.000
4	84	Customer NRCSK	10251	2006-07-08 00:00:00.000
5	76	Customer SFOGW	10252	2006-07-09 00:00:00.000

...

829	9	Customer RTXGC	11076	2008-05-06 00:00:00.000
830	65	Customer NYUHS	11077	2008-05-06 00:00:00.000
831	22	Customer DTDMN	NULL	NULL
832	57	Customer WVAXS	NULL	NULL

Results | Messages

	custid	companyname	orderid	orderdate
1	22	Customer DTDMN	NULL	NULL
2	57	Customer WVAXS	NULL	NULL

Results | Messages

	custid	companyname	orderid	orderdate
1	85	Customer ENQZT	10248	2006-07-04 00:00:00.000
2	79	Customer FAPSM	10249	2006-07-05 00:00:00.000
3	34	Customer IBVRG	10250	2006-07-08 00:00:00.000
4	84	Customer NRCSK	10251	2006-07-08 00:00:00.000
5	76	Customer SFOGW	10252	2006-07-09 00:00:00.000

Results | Messages

custid	companyname	orderid	orderdate

Demonstration C with TSQL: Querying with Outer Joins

```sql
-- Demo Queries are below
USE TSQL;

SELECT c.CustID, soh.OrderID
FROM Sales.Customers c JOIN Sales.Orders soh
ON c.CustID = soh.CustID;

SELECT *
FROM Sales.Customers c LEFT OUTER JOIN Sales.Orders soh
ON c.CustID = soh.CustID;
-- (832 row(s) affected)

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
LEFT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
LEFT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid
WHERE o.orderid IS NULL;
```

Results | Messages

	CustID	OrderID
1	1	10643
2	1	10692
3	1	10702
4	1	10835
5	1	10952

...

Results | Messages

(830 rows affected)

Results | Messages

	custid	companyname	orderid	orderdate
1	85	Customer ENQZT	10248	2006-07-04 00:00:00.000
2	79	Customer FAPSM	10249	2006-07-05 00:00:00.000
3	34	Customer IBVRG	10250	2006-07-08 00:00:00.000
4	84	Customer NRCSK	10251	2006-07-08 00:00:00.000
5	76	Customer SFOGW	10252	2006-07-09 00:00:00.000

Results | Messages

	custid	companyname	orderid	orderdate
1	22	Customer DTDMN	NULL	NULL
2	57	Customer WVAXS	NULL	NULL

```sql
-- Demo Queries are below
USE TSQL;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
RIGHT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid;

SELECT c.custid, c.companyname, o.orderid, o.orderdate
FROM Sales.Customers AS c
RIGHT OUTER JOIN Sales.Orders AS o
ON c.custid =o.custid
WHERE c.custid IS NULL;
```

Results | Messages

	custid	companyname	orderid	orderdate
1	85	Customer ENQZT	10248	2006-07-04 00:00:00.000
2	79	Customer FAPSM	10249	2006-07-05 00:00:00.000
3	34	Customer IBVRG	10250	2006-07-08 00:00:00.000
4	84	Customer NRCSK	10251	2006-07-08 00:00:00.000
5	76	Customer SFOGW	10252	2006-07-09 00:00:00.000
...				
826	58	Customer AHXHT	11073	2008-05-05 00:00:00.000
827	73	Customer JMIKW	11074	2008-05-06 00:00:00.000
828	68	Customer CCKOT	11075	2008-05-06 00:00:00.000
829	9	Customer RTXGC	11076	2008-05-06 00:00:00.000
830	65	Customer NYUHS	11077	2008-05-06 00:00:00.000

Results | Messages

custid	companyname	orderid	orderdate

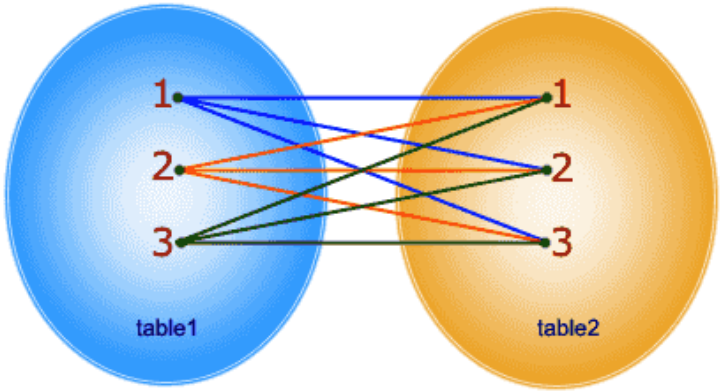
Querying with Cross Joins and Self Joins

- Understanding Cross Joins
- Cross Join Syntax
- Cross Join Examples
- Understanding Self Joins
- Self Join Examples
- Demonstration: Querying with Cross Joins and Self Joins

Cross Join in SQL (Creates Cartesian Product)

- The SQL CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN. This kind of result is called as Cartesian Product.

SELECT * FROM table1 CROSS JOIN table2;



In CROSS JOIN, each row from 1st table joins with all the rows of another table.
If 1st table contain x rows and y rows in 2nd one the result set will be x * y rows.

Syntax:

```
SELECT *
FROM table1
CROSS JOIN table2;
```

Understanding Cross Joins

- Combine each row from first table with each row from second table
- All possible combinations output
- Logical foundation for inner and outer joins
  - Inner join starts with Cartesian product, adds filter
  - Outer join takes Cartesian output, filtered, adds back nonmatching rows (with NULL placeholders)
- Due to Cartesian product output, not typically a desired form of join
- Some useful exceptions:
  - Table of numbers, generating data for testing

Cross Join Syntax

- No matching performed, no **ON clause** used
- Return all rows from left table combined with each row from right table (ANSI SQL-92 syntax):

```
SELECT  ...
FROM t1 CROSS JOIN t2
```

- Return all rows from left table combined with each row from right table (ANSI SQL-89 syntax):

```
SELECT  ...
FROM t1, t2
```

Cross Join Examples

- Extract firstname and lastname from HR.Employees table

```
SELECT firstname, lastname
FROM HR.Employees;
```

- Create test data by returning all combinations of two inputs:

```
SELECT e1.firstname, e2.lastname
FROM HR.Employees AS e1
CROSS JOIN HR.Employees AS e2;
```

Demonstration D with TSQL: Querying with Cross Joins and Self Joins

In this demonstration, you will see how to Use self joins and cross joins

```
-- Demo Queries are below
USE TSQL;

 SELECT e.empid ,e.lastname as empname,e.title,e.mgrid,
m.lastname as mgrname
  FROM HR.Employees AS e
  JOIN HR.Employees AS m
  ON e.mgrid=m.empid;

 SELECT e.empid ,e.lastname as empname,e.title,e.mgrid,
m.lastname as mgrname
  FROM HR.Employees AS e
  LEFT OUTER JOIN HR.Employees AS m
  ON e.mgrid=m.empid;

SELECT e1.firstname, e2.lastname
FROM HR.Employees AS e1 CROSS JOIN HR.Employees AS e2;
```

Results | Messages

	empid	empname	title	mgrid	mgrname
1	2	Funk	Vice President, Sales	1	Davis
2	3	Lew	Sales Manager	2	Funk
3	4	Peled	Sales Representative	3	Lew
4	5	Buck	Sales Manager	2	Funk
5	6	Suurs	Sales Representative	5	Buck
6	7	King	Sales Representative	5	Buck
7	8	Cameron	Sales Representative	3	Lew
8	9	Dolgopyatova	Sales Representative	5	Buck

Results | Messages

	empid	empname	title	mgrid	mgrname
1	1	Davis	CEO	NULL	NULL
2	2	Funk	Vice President, Sales	1	Davis
3	3	Lew	Sales Manager	2	Funk
4	4	Peled	Sales Representative	3	Lew
5	5	Buck	Sales Manager	2	Funk
6	6	Suurs	Sales Representative	5	Buck
7	7	King	Sales Representative	5	Buck
8	8	Cameron	Sales Representative	3	Lew
9	9	Dolgopyatova	Sales Representative	5	Buck

Results | Messages

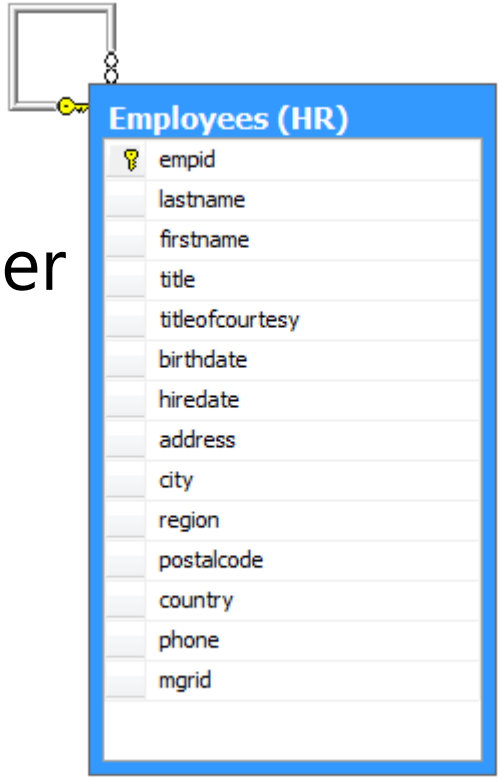
	firstname	lastname
1	Sara	Buck
2	Don	Buck
3	Judy	Buck
4	Yael	Buck
5	Sven	Buck

...

75	Judy	Suurs
76	Yael	Suurs
77	Sven	Suurs
78	Paul	Suurs
79	Russell	Suurs
80	Maria	Suurs
81	Zoya	Suurs

Understanding Self Joins

- Why use self joins?
  - Compare rows in same table to each other
- Create two instances of same table in FROM clause
  - At least one alias required
- Example: Return all employees and the name of the employee's manager



Self Join Examples

- Return all employees with ID of employee's manager when a manager exists (inner join):

```
SELECT  e.empid, e.lastname,
        e.title, e.mgrid, m.lastname
FROM    HR.Employees AS e
JOIN HR.Employees AS m
ON e.mgrid=m.empid;
```

- Return all employees with ID of manager (outer join). This will return NULL for the CEO:

```
SELECT  e. empid, e.lastname,
            e.title, m.mgrid
FROM HR.Employees AS e
LEFT OUTER JOIN HR.Employees AS m
ON e.mgrid=m.empid;
```

Joins (SQL Server)

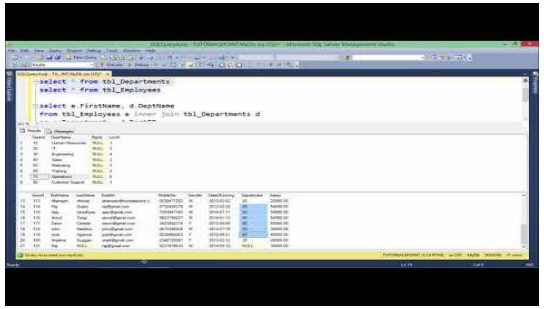
SQL Server Joins Video link:

- Joins in sql server - Part 12

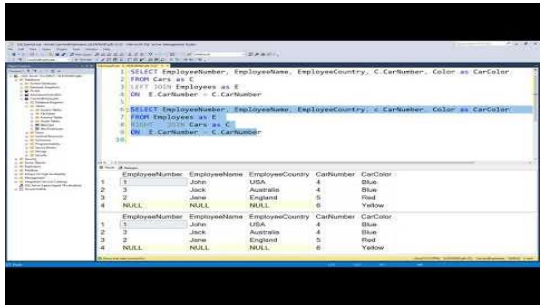
In this video we will learn about

1) The different types of Joins in sql server Cross Join
2) Inner Join
3) Outer Join -- Left , Right and Full Outer Join

T-SQL - Joins



The different types of JOINs in Microsoft SQL Server - INNER, LEFT, RIGHT, FULL and CROSS



Supporting Material

SQL Server Joins Web Resource:

- Microsoft Docs | Joins (SQL Server)
- W3Schools | SQL Joins
- SQL Server Tutorial.net | SQL Server Joins
- Tutorialspoint | SQL Joins
- JavaTpoint | SQL Joins