System Designs & Databases ICA

T-SQL Server – T-SQL Queries to support

European Top Leagues

Name: **Carl Baines**

Course: **System Designs and Databases**

Date**: 18/03/2025**

Tutor Name: **Sumeia Elkazza**

Table of Contents

[T-SQL Server Practitioner Details 3](#_Toc196397324)

[Performance rating 3](#_Toc196397325)

[INTRO 3](#_Toc196397326)

[WHY YOU SHOULD LEARN T-SQL 3](#_Toc196397327)

[T-SQL Server Database Overview 4](#_Toc196397328)

[T-SQL Server DATABASE FOR DEMOS 4](#_Toc196397329)

[T-SQL Server DATABASE DIAGRAMS 4](#_Toc196397330)

[T-SQL Supporting Queries 4](#_Toc196397331)

[T-SQL Part One: SQL Server Coding Basics (T-SQL03 to TSQL08) 5](#_Toc196397332)

[MODULE 3: Writing SELECT Queries with single table 5](#_Toc196397333)

[DEMO 1: Writing Simple SELECT query 5](#_Toc196397334)

[DEMO 2: Eliminating Duplicates with DISTINCT 6](#_Toc196397335)

[DEMO 3: Using Column and Table Aliases 8](#_Toc196397336)

[DEMO 4: Writing SIMPLE Case Expressions 8](#_Toc196397337)

[MODULE 4: Joining and querying multiple tables 9](#_Toc196397338)

[DEMO 1: How to provide data from 2 related tables with a Join 9](#_Toc196397339)

[DEMO 2: How to query with inner joins 10](#_Toc196397340)

[DEMO 3: How to query with outer joins 11](#_Toc196397341)

[DEMO 4: How to query with cross joins and self joins 12](#_Toc196397342)

[MODULE 5: Sorting and filtering data 13](#_Toc196397343)

[DEMO 1: How to Sort Data 14](#_Toc196397344)

[DEMO 2: How to Filter Data with Predicates 15](#_Toc196397345)

[DEMO 3: How to Filter Data with TOP and OFFSET-FETCH 16](#_Toc196397346)

[DEMO 4: How to work with Unknown Values 17](#_Toc196397347)

[MODULE 6: Working with data types 18](#_Toc196397348)

[DEMO 1: Working with Data Type examples 18](#_Toc196397349)

[DEMO 2: Working with Character Data 19](#_Toc196397350)

[DEMO 3: Working with Date and Time Data 21](#_Toc196397351)

[MODULE 7: Using DML To modify data 22](#_Toc196397352)

[DEMO 1: Adding Data to Tables 22](#_Toc196397353)

[DEMO 2: Modifying and Removing Data 23](#_Toc196397354)

[DEMO 3: Generating Automatic Column Values 25](#_Toc196397355)

[MODULE 8: Using Built-in functions 26](#_Toc196397356)

[DEMO 1: Writing Queries with Built-in Functions 26](#_Toc196397357)

[DEMO 2: Using Conversion Functions 27](#_Toc196397358)

[DEMO 3: Using Logical Functions 28](#_Toc196397359)

[DEMO 4: Using Funcitons to Work with NULL 28](#_Toc196397360)

# T-SQL Server Practitioner Details

|  |  |  |  |
| --- | --- | --- | --- |
| **SQL Server - TSQL Practitioner Details:** | | | |
| Name: | **Carl Baines** | | |
| Email Address: | E4092399@live.tees.ac.uk | | |
| Course: | BsC (Hons) Computer Science | | |
| Date: | 18/03/2025 | | |
|  | | Tutor: | Sumeia Elkazza |

## Performance rating

|  |
| --- |
| Graphical user interface, text, website  Description automatically generated |

## INTRO

As a BsC (Hons) Computer Science student at university, I’ve developed a strong interest with software development, particularly in the areas of frontend design and backend databases. I decided upon studying at university to gather experience of using programming languages and data management tools like SQL, whilst also developing my problem-solving skills along the way. I am pursuing my interests as a graduate developer, as I want to demonstrate that I can contribute to projects that require both technical knowledge and creativity.

## WHY YOU SHOULD LEARN T-SQL

I recommend that someone should learn SQL because it is the standard language for managing and manipulating databases. T-SQL, which is Transact-SQL, is Microsoft’s version extension of SQL; it is a powerful tool worth learning for working with SQL server. Another reason why I recommend that someone should pick up SQL is because it is used predominantly in the industry, across many sectors. I hope to gain experience with writing complex queries and performing data analysis.

# T-SQL Server Database Overview

## T-SQL Server DATABASE FOR DEMOS

I have investigated a European Top Leagues SQL Server database to write a range of tailored T-SQL queries aimed at gaining insights from the mass amount of football data provided. The European Top Leagues database contains many tables which contain football data relating to countries, leagues, matches, players and teams. The queries I have written are designed to meet user needs and support various use cases, from the performance analysis of players and teams to app integration. The objective of these queries is to demonstrate the transformation of raw match and player statistics into meaningful data that can be fed into web or mobile applications. Included in this document are examples of my best T-SQL demos to assist users working with the European Top Leagues database.

## T-SQL Server DATABASE DIAGRAMS

|  |
| --- |
| A screenshot of a computer program  AI-generated content may be incorrect. |
| **Main Tables of Interest for Supporting T-SQL Queries** |
| * The player table to be joined with the player\_attributes table. * The team table to be joined with the team\_attributes table. * The team table to be joined with the match table. |

## T-SQL Supporting Queries

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --SELECT \* queries from the different tables in the EuroLeagues database.  --Used to select all data from every column and row from a specific table in the EuroLeagues database.  SELECT \* FROM country;  SELECT \* FROM league;  SELECT \* FROM match;  SELECT \* FROM player;  SELECT \* FROM player\_attributes;  SELECT \* FROM team;  SELECT \* FROM team\_attributes; |
| --Check the data types of all columns in the different tables stored in the EuroLeagues database.  --Replace the TABLE\_NAME string with the table that is needed for check.  SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.COLUMNS  WHERE TABLE\_NAME = 'team'; |
|  |

# T-SQL Part One: SQL Server Coding Basics (T-SQL03 to TSQL08)

## MODULE 3: Writing SELECT Queries with single table

### DEMO 1: Writing Simple SELECT query

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| USE EuroLeagues  ALTER AUTHORIZATION ON DATABASE:: EuroLeagues TO sa  GO  --SELECT \* queries from the different tables in the EuroLeagues database.  --Explanation: Used to select all data from every column and row from a specific table in the EuroLeagues database.  SELECT \* FROM league; |
| --User Story: Select the total number of goals scored from the EuroLeagues.match table.  --Explanation: Simple SELECT query that creates a calculated column, calling the sum function on the home\_team\_goal and away\_team\_goal columns.  SELECT SUM(home\_team\_goal + away\_team\_goal)  FROM match; |
|  |

### DEMO 2: Eliminating Duplicates with DISTINCT

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --User Story: Eliminate duplicate seasons from the EuroLeagues.match table and order them from earliest to latest.  --Explanation: This query uses a subquery and casts the season as a varchar type (it was initially stored as a text value), so that it can work directly with functions like LEFT().  --It extracts the starting years of the seasons (the first four characters), casts them to an int and orders them.  SELECT season  FROM(  SELECT DISTINCT CAST(season AS VARCHAR(MAX)) AS season  FROM match  ) AS season  ORDER BY CAST(LEFT(season, 4) AS INT); |
| **Result of the subquery:**  SELECT DISTINCT CAST(season AS VARCHAR(MAX)) AS season  FROM match |
| **Result of the entire query:** |
| --User Story: Select unique player names from the EuroLeagues.player table and stores them in a column called 'AllPlayerNames'.  SELECT DISTINCT CAST(player\_name AS VARCHAR(MAX)) AS AllPlayerNames  FROM player; |
| ... |

### DEMO 3: Using Column and Table Aliases

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo A3 Query Three  --Module 3: Using Column and Table Aliases Lesson  --User Story: Select the total number of goals scored from the EuroLeagues.match table and assign the column the 'TotalGoalsScored' alias.  SELECT SUM(home\_team\_goal + away\_team\_goal) AS TotalGoalsScored  FROM match; |
|  |
| --Demo A4 Query Two  --Module 3: Using Column and Table Aliases Lesson  --User Story: Select all columns from the EuroLeagues.team table using the alias 'MiddlesbroughFCInfo', where the team\_long\_name is Middlesbrough.  SELECT id, team\_api\_id, team\_fifa\_api\_id, team\_long\_name, team\_short\_name  FROM team AS MiddlesbroughFCInfo  WHERE CAST(team\_long\_name AS VARCHAR(MAX)) = 'Middlesbrough'; |
|  |

### DEMO 4: Writing SIMPLE Case Expressions

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo A4 Query One  --Module 4: Writing Simple CASE expressions  --Demo A4 Query One  --Module 4: Writing Simple CASE expressions  --User Story: Categorise countries by league tier.  --The name of the countries is casted as a varchar so that it can work directly with functions.  SELECT CAST(name AS VARCHAR(MAX)) AS country\_names,  CASE  WHEN CAST(name AS VARCHAR(MAX)) IN ('England', 'Spain', 'France', 'Germany', 'Italy') THEN 'Top 5 League'  ELSE 'Not in Top 5'  END AS League\_Tier  FROM country; |
|  |
| --Demo A4 Query Two  --Module 4: Writing Simple CASE expressions  --User Story: Determine the result of a match using the match table.  SELECT id AS match\_id,  CASE  WHEN home\_team\_goal > away\_team\_goal THEN 'Home Team Won'  WHEN home\_team\_goal < away\_team\_goal THEN 'Away Team Won'  ELSE 'Draw'  END AS Result  FROM match; |
|  |

## MODULE 4: Joining and querying multiple tables

Why use Joining and Querying Multiple Tables?

### DEMO 1: How to provide data from 2 related tables with a Join

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo B1 Query One  --Module 4: How to provide data from 2 related tables with a Join.  --User Story: Select the league names associated with each country.  SELECT c.name AS country\_name, l.name AS league\_name  FROM country AS c  JOIN league AS l  ON c.id = l.country\_id; |
|  |
| --Demo B1 Query Two  --Module 4: How to provide data from 2 related tables with a Join.  SELECT DISTINCT p.player\_api\_id, CAST(p.player\_name AS varchar(MAX)) AS player\_name, pa.overall\_rating, pa.potential AS potential\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  ORDER BY p.player\_api\_id;  --Notice how there are many duplicate player names and ratings, this is because each player has had multiple ratings assigned to them across many career dates. |
|  |

### DEMO 2: How to query with inner joins

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo B2 Query One  --Module 4: How to query with inner joins.  --User Story: Select all the different ratings of the best player (the best player has the highest overall and potential ratings)  --Lionel Messi.  SELECT p.player\_api\_id, CAST(p.player\_name AS varchar(MAX)) AS player\_name, pa.overall\_rating, pa.potential AS potential\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  WHERE pa.overall\_rating = (SELECT MAX(pa.overall\_rating) FROM player\_attributes AS pa)  ORDER BY p.player\_api\_id |
|  |
| --Demo B2 Query Two  --Module 4: How to query with inner joins  --User Story: Join the match table with the team table to get the home and away team names.  SELECT DISTINCT team\_api\_id, CAST(team\_long\_name AS varchar(MAX)) AS team\_long\_name  FROM team  JOIN match  ON home\_team\_api\_id = team\_api\_id OR away\_team\_api\_id = team\_api\_id  ORDER BY team\_api\_id; |
|  |

### DEMO 3: How to query with outer joins

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo B3 Query One  --Module 4: How to query with outer joins  --User Story: full outer join between team and team\_attributes to retrieve a distinct list of all teams, including those with or without associated attribute data.  SELECT DISTINCT t.team\_fifa\_api\_id, CAST(t.team\_long\_name AS varchar(MAX)) AS team\_long\_name, CAST(t.team\_short\_name AS varchar(MAX)) AS team\_short\_name  FROM team AS t  FULL OUTER JOIN team\_attributes AS ta  ON t.team\_fifa\_api\_id = ta.team\_fifa\_api\_id; |
|  |
| --Demo B3 Query Two  --Module 4: How to query with outer joins  --User Story: full outer join between match and team to retrieve a distinct list of all matches, ensuring that match data is included even if team details are duplicated or missing due to the join condition.  SELECT DISTINCT CAST(m.mdate AS varchar(MAX)) AS match\_date, m.match\_api\_id, m.home\_team\_api\_id, m.away\_team\_api\_id, m.home\_team\_goal, m.away\_team\_goal  FROM match AS m  FULL OUTER JOIN team AS t  ON m.home\_team\_api\_id = t.team\_api\_id OR m.away\_team\_api\_id = t.team\_api\_id; |
|  |

### DEMO 4: How to query with cross joins and self joins

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo B4 Query One  --Module 4: How to query with cross and self joins  --Description: Retrieves distinct player names, their fifa API ids, overall ratings, and preferred foot by cross joining the player and player\_attributes tables together.  SELECT DISTINCT CAST(p.player\_name AS varchar(MAX)) AS player\_name, p.player\_fifa\_api\_id, pa.overall\_rating, CAST(pa.preferred\_foot AS varchar(MAX)) AS preferred\_foot  FROM player AS p  CROSS JOIN player\_attributes AS pa  WHERE p.player\_api\_id = pa.player\_api\_id; |
|  |
| SELECT DISTINCT TOP 100  p1.player\_api\_id AS p1\_api\_id,  CAST(p1\_player.player\_name AS varchar(MAX)) AS player\_for\_comparison,  p1.finishing,  p1.shot\_power,  p2.player\_api\_id AS p2\_api\_id,  CAST(p2\_player.player\_name AS varchar(MAX)) AS p2\_name,  p2.finishing,  p2.shot\_power  FROM player\_attributes p1  JOIN player\_attributes p2  ON p1.player\_api\_id = 2625 AND p1.player\_api\_id <> p2.player\_api\_id  JOIN player p1\_player ON p1.player\_api\_id = p1\_player.player\_api\_id  JOIN player p2\_player ON p2.player\_api\_id = p2\_player.player\_api\_id  ORDER BY p2.player\_api\_id; |
|  |

## MODULE 5: Sorting and filtering data

### DEMO 1: How to Sort Data

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo B5 Query One  --Module 5: How to Sort Data  --Description: Select id and player names from the EuroLeagues.player table, ordering the id in ascending order.  --Simple SELECT query with ORDER by clause.  SELECT id, player\_name  FROM player  ORDER BY id ASC; |
| A screenshot of a black and white screen  AI-generated content may be incorrect. |
| --Demo B5 Query Two  --Module 5: How to Sort Data  --Description: Selects the player\_api\_id, name and height from the EuroLeagues.player table, ordering the players by height in descending order.  SELECT player\_api\_id, player\_name, height  FROM player  ORDER BY height DESC; |
|  |

### DEMO 2: How to Filter Data with Predicates

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo C2 Query One  --Module 5: How to filter data with predicates.  --Description: Retrieves a list of distinct players that have an overall rating greater than 80. Each player only appears once with their highest rating.  SELECT DISTINCT pa.player\_fifa\_api\_id, CAST(p.player\_name AS varchar(MAX)) AS player\_name, MAX(pa.overall\_rating) AS overall\_rating  FROM player\_attributes AS pa  JOIN player AS p  ON p.player\_fifa\_api\_id = pa.player\_fifa\_api\_id  WHERE overall\_rating > 80  GROUP BY pa.player\_fifa\_api\_id, CAST(p.player\_name AS varchar(MAX)); |
|  |
| --Demo C2 Query Two  --Module 5: How to filter data with predicates.  --Description: The query retrieves all match details involving Middlesbrough, including the teams they played, the season (as well as its stage), the matchID and the goals scored.  SELECT  home\_team.team\_api\_id AS home\_team\_api\_id,  CAST(home\_team.team\_long\_name AS VARCHAR(MAX)) AS home\_team,  away\_team.team\_api\_id AS away\_team\_api\_id,  CAST(away\_team.team\_long\_name AS VARCHAR(MAX)) AS away\_team,  CAST(m.season AS VARCHAR(MAX)) AS season,  m.stage,  m.match\_api\_id,  m.home\_team\_goal,  m.away\_team\_goal  FROM match AS m  JOIN team AS home\_team ON home\_team.team\_api\_id = m.home\_team\_api\_id  JOIN team AS away\_team ON away\_team.team\_api\_id = m.away\_team\_api\_id  WHERE  CAST(home\_team.team\_long\_name AS varchar(MAX)) = 'Middlesbrough'  OR CAST(away\_team.team\_long\_name AS varchar(MAX)) = 'Middlesbrough'; |
|  |

### DEMO 3: How to Filter Data with TOP and OFFSET-FETCH

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo C3 Query One  --Module 5: How to filter data with TOP and OFFSET-FETCH.  --Description: Selects top 100 players and their api ids from the players table, joining with the player attributes table to get their overall ratings.  --Their overall ratings are ordered in descending order.  SELECT DISTINCT TOP 100 CAST(p.player\_name AS nvarchar(MAX)) AS player\_name, p.player\_api\_id, ISNULL(pa.overall\_rating, '0') AS overall\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  GROUP BY CAST(p.player\_name AS nvarchar(MAX)), p.player\_api\_id, pa.overall\_rating  ORDER BY overall\_rating DESC |
|  |
| --Demo C3 Query Two  --Module 5: How to filter data with TOP and OFFSET-FETCH  --Description: Selects the bottom 10 of the top 100 players with their api ids from the players table, joining with the players attributes table to get their overall ratings.  SELECT DISTINCT CAST(p.player\_name AS nvarchar(MAX)) AS player\_name, p.player\_api\_id, ISNULL(pa.overall\_rating, '0') AS overall\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  GROUP BY CAST(p.player\_name AS nvarchar(MAX)), p.player\_api\_id, pa.overall\_rating  ORDER BY overall\_rating DESC  OFFSET 90 ROWS  FETCH NEXT 10 ROWS ONLY; |
|  |

### DEMO 4: How to work with Unknown Values

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo C4 Query One  --Module 5: How to work with unknown values  --Description: Selects the team\_fifa\_api\_id, the long name and short name of teams from the teams table.  --If the team\_fifa\_api\_id is null, the string 'No FIFA API ID' is replaced in place of the null value.  SELECT ISNULL(CAST(TRY\_CAST(team\_fifa\_api\_id AS INT) AS VARCHAR(255)), 'No FIFA API ID') AS fifa\_api\_id, team\_long\_name, team\_short\_name  FROM team; |
|  |
| --Demo C4 Query Two  --Module 5: How to work with unknown values  --Description: Returns a list of player names with missing overall ratings.  SELECT p.player\_name, pa.overall\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  WHERE pa.overall\_rating IS NULL; |
|  |

## MODULE 6: Working with data types

### DEMO 1: Working with Data Type examples

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo D1 Query One  --Module 6: Working with data types examples  --Description: This query demonstrates working with data types by casting numeric and text fields using CAST and handling null values with ISNULL.  --The query selects distinct player names and their overall ratings, ordered in ascending order. If a rating is null, it displays 'N/A'.  SELECT DISTINCT  CAST(p.player\_name AS VARCHAR) AS player\_name,  ISNULL(CAST(pa.overall\_rating AS VARCHAR), 'N/A') + ' OVR' AS Overall\_Rating  FROM player\_attributes pa  JOIN player p ON pa.player\_api\_id = p.player\_api\_id  ORDER BY Overall\_Rating ASC; |
|  |
| --Demo D1 Query Two  --Module 6: Working with data types examples  --Description: This query demonstrates working with data types as it selects all columns in the database, ordered by table\_name  --and displays the data type and max character length of each.  SELECT TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH  FROM INFORMATION\_SCHEMA.COLUMNS  ORDER BY TABLE\_NAME; |
|  |

### DEMO 2: Working with Character Data

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo D2 Query One  --Module 6: Working with Character Data  --Original query I wanted to concatenate.  SELECT DISTINCT  CAST(p.player\_name AS varchar(MAX)) AS player\_name,  MAX(pa.overall\_rating) AS overall\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  GROUP BY CAST(p.player\_name AS varchar(MAX));  --Description: This query returns a list of unique players from the player table alongside their highest overall rating  --from the player\_attributes table, formatted as a single string.  SELECT DISTINCT  CONCAT(  CAST(p.player\_name AS varchar(MAX)),  N' (overall\_rating: ',  CAST(MAX(pa.overall\_rating) AS NVARCHAR),  N')'  ) AS playerWithRating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  GROUP BY CAST(p.player\_name AS varchar(MAX)); |
| **Original Query Output** |
|  |
| **Query Output with Concatenation** |
|  |
| --Demo D2 Query Two  --Module 6: Working with Character Data  --Original Query I wanted to concatenate.  SELECT team\_long\_name, team\_short\_name  FROM team;  --Description: This query returns a list of team short and long names, formatted as a single string.  SELECT  CONCAT(  team\_long\_name,  N' (short\_name: ',  team\_short\_name,  N')'  ) AS teamShortAndLongNames  FROM team; |
| **Original Query Output** |
|  |
| **Query Output with Concatenation** |
|  |

### DEMO 3: Working with Date and Time Data

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo D3 Query One  --Module 6: Working with Date and Time Data  --Description: This query returns the difference between the first match date and the last match date stored in the match table.  SELECT DATEDIFF(  DAY,  (SELECT TOP 1 CAST(mdate AS varchar(MAX)) AS mdate FROM match ORDER BY mdate ASC),  (SELECT TOP 1 CAST(mdate AS varchar(MAX)) AS mdate FROM match ORDER BY mdate DESC)  ) AS daysBetween |
|  |
| --Demo D3 Query Two  --Module 6: Working with Date and Time Data  --Description: This query returns all player names and their birthdays from the player table along with their age. The query results are ordered by the oldest birthday.  --The age is calculated from the birthday datetime values by using the DATEDIFF function.  --The birthday column is converted from a text type and is first casted to a varchar, so that it can then be casted to a date type, since SQL server  --does not allow for text types to be converted straight to a date/datetime type.  SELECT player\_name,  CAST(CAST(birthday AS varchar(MAX)) AS DATE) AS birthday,  DATEDIFF(YEAR, CAST(CAST(birthday AS varchar(MAX)) AS DATE), '2025-04-16') as Age  FROM player  ORDER BY birthday; |
|  |

## MODULE 7: Using DML To modify data

Why use DML to modify data?

DML is the short name for Data Manipulation Language which deals with data manipulation. Examples of DML in SQL include statements such as SELECT, INSERT, UPDATE and DELETE etc, which are used to store, modify, retrieve, delete and update data within a database.

### DEMO 1: Adding Data to Tables

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo D4 Query One  --Module 7: Using DML to Modify Data.  --Description: Add the country San Marino to the countries table with a designated ID.  INSERT INTO country (id, name)  VALUES('26518','San Marino'); |
| --Using the \* wildcard to return all rows and values from the country table.  SELECT \* FROM country; |
| --Demo D4 Query Two  --Module 7: Adding data to tables using DML.  --Description: Adding the second tier leagues of each country into the league table.  INSERT INTO league(id, country\_id, name)  VALUES  ('101','1','Challanger Pro League'),  ('102','1729','EFL Championship'),  ('103','4769','Ligue 2'),  ('104','7809','Bundesliga 2'),  ('105','10257','Serie B'),  ('106','13274','Eerste Divisie'),  ('107','15722','Betclic l liga'),  ('108','17642','Liga Portugal 2'),  ('109','19694','Scottish Championship'),  ('110','21518','La Liga 2'),  ('111','24558','Swiss Challenge League'); |
|  |

### DEMO 2: Modifying and Removing Data

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo E1 Query One  --Module 7: Modifying and Removing Data  --The query which returns a result I want to update.  --It selects the best player based on highest overall\_rating, returning the name, api id, overall\_rating, with their dribbling and ball control statistics.  SELECT DISTINCT TOP 1 CAST(p.player\_name AS nvarchar(MAX)) AS player\_name, p.player\_api\_id, pa.overall\_rating AS overall\_rating, pa.dribbling, pa.ball\_control  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  GROUP BY CAST(p.player\_name AS nvarchar(MAX)), p.player\_api\_id, pa.overall\_rating, pa.dribbling, pa.ball\_control  ORDER BY overall\_rating DESC  --Description: This is the query I used to update the ball control statistic of the best player (Lionel Messi), based on highest overall\_rating.  --It finds Messi and increases his ball\_control stat by 3.  WITH TopPlayer AS (  SELECT TOP 1 pa.player\_api\_id AS player\_attribute\_id  FROM player AS p  JOIN player\_attributes AS pa ON p.player\_api\_id = pa.player\_api\_id  ORDER BY pa.overall\_rating DESC  )  UPDATE player\_attributes  SET overall\_rating = overall\_rating + 2  WHERE id IN (SELECT player\_attribute\_id FROM TopPlayer); |
| **Query Output Before UPDATE Statement** |
|  |
| **Query Output After UPDATE Statement** |
|  |
| --Demo E1 Query Two  --Module 7: Modifying and Removing Data  --The query which returns a result I want to update.  SELECT TOP 10 \* FROM team  ORDER BY team\_api\_id  --Description: Deletes the team 'Ruch Chorzów' from the team table.  DELETE FROM team WHERE team\_long\_name = 'Ruch Chorzów'; |
| **Query Output Before DELETE Statement** |
|  |
| **Query Output After DELETE Statement** |
|  |

### DEMO 3: Generating Automatic Column Values

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo E2 Query One  --Module 7: Generating Automatic Column Values  --Description: Adds a new column to the player\_attributes table. The values within the column are automatically generated by performing addition on the dribbling  --and ball control statistics of each player.  ALTER TABLE player\_attributes  ADD skill\_score AS (dribbling + ball\_control);  --Query which selects the api ids, overall ratings, potential ratings and skill scores of each player from the player\_attributes table. It is ordered by skill\_score in descending order.  SELECT player\_api\_id, overall\_rating, potential, ISNULL(skill\_score, '0') AS skill\_score FROM player\_attributes  ORDER BY skill\_score DESC; |
|  |
| --Demo E2 Query Two  --Module 7: Generating Automatic Column Values  --Description: Adds two new columns to the match table. One column calculates the home team goal difference whereas the other calculates the away team goal difference.  --The values are automatically generated by performing subtraction each way on the number of home team and away team goals scored within a match.  ALTER TABLE match  ADD home\_team\_goal\_difference AS (home\_team\_goal - away\_team\_goal),  away\_team\_goal\_difference AS (away\_team\_goal - home\_team\_goal);  --Query which selects the home team and away team goals scored in each match with the calculated goal difference for each team in said match.  SELECT home\_team\_goal, away\_team\_goal, home\_team\_goal\_difference, away\_team\_goal\_difference  FROM match; |
|  |

## MODULE 8: Using Built-in functions

Why do programmers use built-in functions?

SQL, as well as the vast majority of programming languages, use functions. They are blocks of reusable code that can be repeatedly called upon to perform an instruction or set of instructions. The biggest reason programmers use built-in functions is because it allows complex programs to be decomposed.

### DEMO 1: Writing Queries with Built-in Functions

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo E3 Query One  --Module 8: Writing Queries with Built-In Functions  --Description: The query calculates the total number of matches each team has played, regardless of whether they are home or away.  --It uses the count built-in function.  WITH teams AS (  SELECT DISTINCT t.team\_api\_id, CAST(t.team\_long\_name AS varchar(MAX)) AS team\_long\_name  FROM team AS t  JOIN match AS m  ON t.team\_api\_id = m.home\_team\_api\_id OR t.team\_api\_id = m.away\_team\_api\_id  )  SELECT  t.team\_long\_name,  COUNT(m.match\_api\_id) AS total\_matches  FROM teams AS t  JOIN match AS m  ON t.team\_api\_id = m.home\_team\_api\_id OR t.team\_api\_id = m.away\_team\_api\_id  GROUP BY t.team\_long\_name  ORDER BY total\_matches DESC; |
| **Subquery Output** |
|  |
| **Entire Query Output** |
|  |
| --Demo E3 Query Two  --Module 8: Writing Queries with Built-In Functions  --Description: This query selects the player with the highest average overall and potential ratings, using the built-in AVG function.  SELECT TOP 1 CAST(p.player\_name AS nvarchar(MAX)) AS player\_name, p.player\_api\_id, AVG(pa.overall\_rating) AS average\_overall\_rating, AVG(pa.potential) AS average\_potential\_rating  FROM player AS p  JOIN player\_attributes AS pa  ON p.player\_api\_id = pa.player\_api\_id  GROUP BY CAST(p.player\_name AS nvarchar(MAX)), p.player\_api\_id  ORDER BY average\_overall\_rating DESC |
|  |

### DEMO 2: Using Conversion Functions

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
| --Demo E4 Query One  --Module 8: Using Conversion Functions  --Description: Counts the total matches played per year in the match table.  --The query extracts just the year part of the mdate and the count function is used to count all the number of matches for each year.  --The result is grouped and ordered by the myear.  SELECT YEAR(CAST(CAST(mdate AS varchar(MAX)) AS datetime)) AS myear, COUNT(\*) AS total\_matches  FROM match  GROUP BY YEAR(CAST(CAST(mdate AS varchar(MAX)) AS datetime))  ORDER BY myear; |
|  |
| --Demo E4 Query Two  --Module 8: Using Conversion Functions  --Description: This query selects distinct player names and their overall ratings by joining onto the player\_attributes table.  --It uses a case statement to classify the players into rating tiers based on their overall ratings.  --The query uses a conversion function as it converts the data type of the player name from text to varchar, so that it can be selected as distinct.  SELECT DISTINCT CAST(p.player\_name AS varchar(MAX)) AS player\_name, pa.overall\_rating,  CASE  WHEN pa.overall\_rating >= 90 THEN 'Goats (OVR 90+'  WHEN pa.overall\_rating >= 85 THEN 'Professionals (OVR 85-89)'  WHEN pa.overall\_rating >= 75 THEN 'Rising Stars (OVR 75-84)'  WHEN pa.overall\_rating >= 65 THEN 'Average (OVR 65-74)'  ELSE 'Flops (OVR UNDER 65)'  END AS player\_rating  FROM player AS p  JOIN player\_attributes AS pa ON p.player\_api\_id = pa.player\_api\_id; |
|  |

### DEMO 3: Using Logical Functions

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
|  |

### DEMO 4: Using Functions to Work with NULL

|  |
| --- |
| **TSQL Demo Code Evidence/Results in SSMS** |
|  |