# Systems Design and Databases (CIS1018-N) TSQL Tutorial 6: Querying with Multiple Tables

## Before You Start

Finish the previous tutorials before attempting this one. Try to complete this set of tasks before your next tutorial.

**Hint:** It is important before start the lab, you should walk-through lecture(s) and demonstration exercises

## Introduction

This session aims to familiarise you with Querying with Multiple Tables .

You are an Adventure Works business analyst who will be writing reports using corporate databases stored in SQL Server 2019. You have been given a set of business requirements for data and you will write T-SQL queries to retrieve the specified data from the databases.

You notice that the data is stored in separate tables, so you will need to write queries using various join operations.

## Querying Multiple Tables

You notice that the data is stored in separate tables, so you will need to write queries using various join operations.

* Exercise 1: Writing Queries That Use Inner Joins
* Exercise 2: Writing Queries That Use Multiple-Table Inner Joins
* Exercise 3: Writing Queries That Use Self Joins
* Exercise 4: Writing Queries That Use Outer Joins
* Exercise 5: Writing Queries That Use Cross Joins

## Exercise 1: Writing Queries That Use Inner Joins

* Write a SELECT statement that will return the productname column from the Production.Products table (use table alias “p”) and the categoryname column from the Production.Categories table (use table alias “c”) using an inner join.
* Execute the written statement and compare the results that you got with the desired results shown in the below “Possible Result of Query” - Lab Exercise 1 - Task 1 Result.txt.
* Which column did you specify as a predicate in the ON clause of the join? Why?
* Let us say that there is a new row in the Production.Categories table and this new product category does not have any products associated with it in the Production.Products table. Would this row be included in the result of the SELECT statement written in task 1? Please explain.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
| … |

## Exercise 2: Writing Queries That Use Multiple-Table Inner Joins

Task 1:

* Execute the query exactly as written inside a query window and observe the result.
* Execute the query exactly as written inside a query window and observe the result.
* You get an error. What is the error message? Why do you think you got this error?

|  |
| --- |
| -- Task1 Query is here |
| USE TSQL;  SELECT  custid, contactname, orderid  FROM Sales.Customers  INNER JOIN Sales.Orders ON Customers.custid = Orders.custid; |

Task 2:

* Notice that there are full source table names written as table aliases.
* Apply the needed changes to the SELECT statement so that it will run without an error. Test the changes by executing the T-SQL statement.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
| … |

Task 3:

* Copy the T-SQL statement from task 2 and modify it to use the table aliases “C” for the Sales.Custumers table and “O” for the Sales.Orders table.
* Execute the written statement and compare the results with the results in task 2.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
| … |

Task 4:

* Copy the T-SQL statement from task 3 and modify it to include three additional columns from the Sales.OrderDetails table: productid, qty, and unitprice.
* Execute the written statement and compare the results that you got with the below Possible Result of Query.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
| … |

## Exercise 3: Writing Queries That Use Self Joins

Task 1:

* In order to better understand the needed tasks, you will first write a SELECT statement against the HR.Employees table showing the empid, lastname, firstname, title, and mgrid columns.
* Execute the written statement and compare the results that you got with showing below “Possible Result of Query”.

**Notice**: the values in the mgrid column. The mgrid column is in a relationship with empid column. This is called a self-referencing relationship.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
|  |

Task 2:

* Copy the SELECT statement from task 1 and modify it to include additional columns for the manager information (lastname, firstname) using a self-join. Assign the aliases mgrlastname and mgrfirstname, respectively, to distinguish the manager names from the employee names.
* Execute the written statement and compare the results that you got with below possible results of query.
* Is it mandatory to use table aliases when writing a statement with a self-join? Can you use a full source table name as alias? Please explain.
* Why did you get fewer rows in the T-SQL statement under task 2 compared to task 1?

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
|  |

## Exercise 4: Writing Queries That Use Outer Joins

Task 1:

* Write a SELECT statement to retrieve the custid and contactname columns from the Sales.Customers table and the orderid column from the Sales.Orders table. The statement should retrieve all rows from the Sales.Customers table.
* Execute the written statement and compare the results that you got with the below given possible result of query.

**Notice**: the values in the column orderid. Are there any missing values (marked as NULL)? Why?

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
| …    … |

## Exercise 5: Writing Queries That Use Cross Joins

Task 1

* Execute the T-SQL code under Task 1. Do not worry if you do not understand the provided T-SQL code, as it is used here to provide a more realistic example for a cross join in the next task.

|  |
| --- |
| -- Insert Query here |
| USE TSQL;  SET NOCOUNT ON;  IF OBJECT\_ID('HR.Calendar') IS NOT NULL  DROP TABLE HR.Calendar;  CREATE TABLE HR.Calendar (  calendardate DATE CONSTRAINT PK\_Calendar PRIMARY KEY  );  DECLARE  @startdate DATE = DATEFROMPARTS(YEAR(SYSDATETIME()), 1, 1),  @enddate DATE = DATEFROMPARTS(YEAR(SYSDATETIME()), 12, 31);  WHILE @startdate <= @enddate  BEGIN  INSERT INTO HR.Calendar (calendardate)  VALUES (@startdate);  SET @startdate = DATEADD(DAY, 1, @startdate);  END;  SET NOCOUNT OFF;  -- observe the HR.Calendar table  SELECT  calendardate  FROM HR.Calendar; |

Task 2:

* Write a SELECT statement to retrieve the empid, firstname, and lastname columns from the HR.Employees table and the calendardate column from the HR.Calendar table.
* Execute the written statement and compare the results that you got with the possible result of query.
* What is the number of rows returned by the query? There are nine rows in the HR.Employees table. Try to calculate the total number of rows in the HR.Calendar table.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

|  |
| --- |
| **Possible Result of Query** |
| … |

Task 3:

* Write a statement to remove the HR.Calendar table.

|  |
| --- |
| -- Insert Query here |
| USE TSQL; |

## Document History

Prepared by Dr Yar Muhammad,   
Revision 0. (29-Aug-22): This is the initial version of the 2022/23 exercise.