Logo

Description automatically generated with medium confidence

**Systems Design & Databases**

Graphical user interface, application

Description automatically generated

**name: OGEDEGBE JOSAPHAT EKAN**

**course: SYSTEM DESIGN AND DATABASE**

**date: May 8, 2024**

**Tutor Name: Mansha Nawaz**

**SQL Server - TSQL Queries to support:**

**[movies]**

**Table of Appendices**

[SQL Server Practitioner Details: 5](#_Toc121418924)

[a) SQL Server Practitioner Performance Rating: 5](#_Toc121418925)

[b) Introduction to the SQL Practitioner: 5](#_Toc121418926)

[c) Why you should learn SQL: 5](#_Toc121418927)

[SQL Server Database Overview: 6](#_Toc121418928)

[a) SQL Server Database for Demos: 6](#_Toc121418929)

[b) SQL Server Database Diagrams: 6](#_Toc121418930)

[Introduction: 7](#_Toc121418931)

[TSQL Part 1: SQL Server Coding Basics 7](#_Toc121418932)

[1. TSQL03 to TSQL08: SQL Server Basics 7](#_Toc121418933)

[a) Module 3: Writing SELECT Queries with single Table 7](#_Toc121418934)

[Demo A1: Writing Simple SELECT Query 7](#_Toc121418935)

[Demo A2: Eliminating Duplicates with DISTINCT 8](#_Toc121418936)

[Demo A3: Using Column and Table Aliases Lesson 8](#_Toc121418937)

[Demo A4: Writing Simple CASE Expressions 8](#_Toc121418938)

[b) Module 4: Joining and Querying Multiple Tables 8](#_Toc121418939)

[Demo B1: How to provide data from 2 related tables with a Join 9](#_Toc121418940)

[Demo B2: How to Query with Inner Joins 9](#_Toc121418941)

[Demo B3: How to Query with Outer Joins 9](#_Toc121418942)

[Demo B4: How Query with Cross Joins and Self Joins 9](#_Toc121418943)

[c) Module 5: Sorting and Filtering Data 9](#_Toc121418944)

[Demo C1: How to Sort Data 9](#_Toc121418945)

[Demo C2: How to Filter Data with Predicates 9](#_Toc121418946)

[Demo C3: How to Filter Data with TOP and OFFSET-FETCH 9](#_Toc121418947)

[Demo C4: How to work with Unknown Values 9](#_Toc121418948)

[d) Module 6: Working with Data Types 9](#_Toc121418949)

[Demo D1: Working with Data Type examples 9](#_Toc121418950)

[Demo D2: Working with Character Data 9](#_Toc121418951)

[Demo D3: Working with Date and Time Data 9](#_Toc121418952)

[e) Module 7: Using DML to Modify Data 9](#_Toc121418953)

[Demo E1: Adding Data to Tables 9](#_Toc121418954)

[Demo E2: Modifying and Removing Data 9](#_Toc121418955)

[Demo E3: Generating Automatic Column Values 9](#_Toc121418956)

[f) Module 8: Using Built-In Functions 9](#_Toc121418957)

[Demo F1: Writing Queries with Built-In Functions 9](#_Toc121418958)

[Demo F2: Using Conversion Functions 9](#_Toc121418959)

[Demo F3: Using Logical Functions 9](#_Toc121418960)

[Demo F4: Using Functions to Work with NULL 9](#_Toc121418961)

# SQL Server Practitioner Details:

*Please enter your details below:*

|  |  |  |
| --- | --- | --- |
| **SQL Server - TSQL Practitioner Details:** | | |
| A child holding a camera  Description automatically generated with medium confidence | Name: | **OGEDEGBE JOSAPHAT EKAN** |
| Email Address: | C2449032@live.tees.ac.uk |
| Course: | SYSTEM DESIGN AND DATABASE |
| Date: | May 8, 2024 |
|  | Tutor: | MANSHA NAWAZ |

## SQL Server Practitioner Performance Rating:

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

## Introduction to the SQL Practitioner:

**Why you decided upon studying and pursuing your interests in becoming a graduate developer.**

As a junior graduate developer, I am deeply passionate about technology and its potential to drive innovation and solve real-world problems. My aspirations as a developer include continuously learning and growing in my skills, collaborating with diverse teams, and contributing to the development of impactful solutions that positively impact people's lives. I decided to pursue a career as a graduate developer because I see it as a dynamic field where I can apply my analytical thinking, creativity, and love for technology to build meaningful applications and make a difference in the world.

## [Why you should learn SQL](https://uk.search.yahoo.com/search?fr=mcafee&type=E210GB714G0&p=why+learn+sql+server):

**Why you recommend someone should learn SQL and what you should hope to gain from it.**

By mastering SQL Server, I gain the ability to efficiently create, read, update and delete data stored in databases, which is fundamental to almost every organization's operation.

# SQL Server Database Overview:

## SQL Server Database for Demos:

I am going to investigate the SQL Server Database **Movies Database** to develop range of useful TSQL Queries and Scripts to support **business functions** or **user requirements**.

The aim is to provide useful patterns of data to serve front end development technologies such as Web or Mobile Applications.

Provided below in this document are examples of my best TSQL Demos (Queries and Scripts) to support users of **Movies Database**.

## SQL Server Database Diagrams:



**SQL Practitioners TSQL Demos:**

## Introduction:

Provided below is an audit trail of my best examples of TSQL querying skills to support ‘business functions’ or user requirements for **movies** database.

# TSQL Part 1: SQL Server Coding Basics

# TSQL03 to TSQL08: SQL Server Basics

This section covers the basics skills in using SSMS and scoping TSQL Queries either by code or by using the [**Design Query in Editor**](https://uk.search.yahoo.com/search;_ylt=AwrkNcIUMGBjOwE9UxsM34lQ;_ylu=Y29sbwNpcjIEcG9zAzEEdnRpZAMEc2VjA3Fydw--?type=E210GB714G0&fr=mcafee&ei=UTF-8&p=sql+server+design+query+in+editor&fr2=12642) how to use [Select](https://uk.search.yahoo.com/search;_ylt=AwrLApgXMGBj2Ks8V4AM34lQ;_ylc=X1MDMTM1MTIxMjgxMgRfcgMyBGZyA21jYWZlZQRmcjIDc2ItdG9wBGdwcmlkA081VzZURjNaUmllTVYyVGphdUhGTkEEbl9yc2x0AzAEbl9zdWdnAzQEb3JpZ2luA3VrLnNlYXJjaC55YWhvby5jb20EcG9zAzAEcHFzdHIDBHBxc3RybAMwBHFzdHJsAzI3BHF1ZXJ5A3NxbCUyMHNlcnZlciUyMHNlbGVjdCUyMHN0YXRlbWVudAR0X3N0bXADMTY2NzI0ODIxMw--?p=sql+server+select+statement&fr2=sb-top&fr=mcafee&type=E210GB714G0) statements to query data from table(s), [Join](https://uk.search.yahoo.com/search;_ylt=AwrIfJlTMGBjh2o8ntcM34lQ;_ylc=X1MDMTM1MTIxMjgxMgRfcgMyBGZyA21jYWZlZQRmcjIDc2ItdG9wBGdwcmlkA2FIX0I1S2w5U0d5b1MyOEkub0F4REEEbl9yc2x0AzAEbl9zdWdnAzAEb3JpZ2luA3VrLnNlYXJjaC55YWhvby5jb20EcG9zAzAEcHFzdHIDBHBxc3RybAMwBHFzdHJsAzI1BHF1ZXJ5A3NxbCUyMHNlcnZlciUyMGpvaW4lMjBzdGF0ZW1lbnQEdF9zdG1wAzE2NjcyNDgyNDE-?p=sql+server+join+statement&fr2=sb-top&fr=mcafee&type=E210GB714G0) across related tables, sort and filtering with [Where](https://uk.search.yahoo.com/search;_ylt=AwrLAphwMGBj3BQ9WhQM34lQ;_ylc=X1MDMTM1MTIxMjgxMgRfcgMyBGZyA21jYWZlZQRmcjIDc2ItdG9wBGdwcmlkA0VIWHZ5SHJwUlBDOHZQVTBwVGhHdUEEbl9yc2x0AzAEbl9zdWdnAzAEb3JpZ2luA3VrLnNlYXJjaC55YWhvby5jb20EcG9zAzAEcHFzdHIDBHBxc3RybAMwBHFzdHJsAzI2BHF1ZXJ5A3NxbCUyMHNlcnZlciUyMHdoZXJlJTIwc3RhdGVtZW50BHRfc3RtcAMxNjY3MjQ4MzAw?p=sql+server+where+statement&fr2=sb-top&fr=mcafee&type=E210GB714G0), modifying data and using built in functions for **movies**:

|  |  |
| --- | --- |
| **.sql File for**  **TSQL03-08 Demos:** | [SQL\_FILE](ekan_ogedegbe%20Queries.sql) |

## Module 3: Writing SELECT Queries with single Table

**Why write Select queries?**

The purpose of the SELECT statement is to query database tables, apply logical manipulation to the data, and result a result set.

### Demo A1: Writing Simple SELECT Query

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to retrieve all columns,  -- all rows from dbo.movie table  SELECT \*  FROM dbo.movie;    -- Step 7: Simple SELECT query with calculated column  -- Select and execute the following query to manipulate columns from the dbo.movie table.  -- Note the lack of name for the new calculated column.  SELECT movie\_id, title, budget, popularity, runtime, (popularity \* runtime)  FROM dbo.movie; |

### Demo A2: Eliminating Duplicates with DISTINCT

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Step 1. Select and execute the following query without distinct  SELECT budget, revenue, runtime, vote\_average, vote\_count  FROM dbo.movie;    -- Step 2. Select and execute the following query to distinct budget, revenue, runtime, vote\_average and vote\_count columns,  -- all rows from dbo.movie table  SELECT DISTINCT budget, revenue, runtime, vote\_average, vote\_count  FROM dbo.movie; |

### Demo A3: Using Column and Table Aliases Lesson

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to use column and table aliases,  -- from dbo.language table  -- Using Table Aliases  SELECT l.language\_id, l.language\_code, l.language\_name  FROM language AS l    -- Using Column Aliases  SELECT language\_id AS LanguageID, language\_code AS LanguageCode, language\_name AS LanguageName  FROM language; |

### Demo A4: Writing Simple CASE Expressions

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to write simple CASE expression,  -- from dbo.movie table  SELECT  movie\_id,  title,  overview,  CASE movie\_status  WHEN 'released' THEN 'Available for viewing'  WHEN 'post-production' THEN 'Currently in post-production'  WHEN 'filming' THEN 'Currently filming'  ELSE 'Unknown status'  END AS movie\_status\_description  FROM movie; |

## Module 4: Joining and Querying Multiple Tables

**Why use Joining and Querying Multiple Tables?**

By making multiple queries and joining the data in code will make multiple requests to your database, one for each table you need data from. The advantage of using a join in the SQL query will reduce the number of connection made to just one. This is especially advantageous if your database server is on a separate machine.

### Demo B1: How to provide data from 2 related tables with a Join

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to provide data from 2 related tables with a Join,  -- from dbo.department and dbo.movie\_crew table  SELECT d.department\_id, d.department\_name, mc.job  FROM department AS d JOIN movie\_crew AS mc ON d.department\_id = mc.department\_id; |

### Demo B2: How to Query with Inner Joins

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to query with Inner Joins,  -- from dbo.person and dbo.movie\_cast table  SELECT p.person\_id, p.person\_name, mc.character\_name, mc.cast\_order  FROM person AS p  INNER JOIN movie\_cast AS mc ON p.person\_id = mc.person\_id; |

### Demo B3: How to Query with Outer Joins

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to query with Outer Joins,  -- from dbo.department and dbo.movie\_crew table  -- Using Left JOIN  SELECT department.department\_name, movie\_crew.job  FROM department  LEFT JOIN movie\_crew ON department.department\_id = movie\_crew.department\_id; |

### Demo B4: How Query with Cross Joins and Self Joins

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to query with Cross Joins and Self Joins,  -- from dbo.language and dbo.movie\_languages table  -- Using Cross JOIN  SELECT \* FROM language  CROSS JOIN movie\_languages;  -- Using Self Join  SELECT mc.person\_id, mc.job, mc2.department\_id  FROM movie\_crew AS mc  JOIN movie\_crew AS mc2 ON mc.department\_id = mc2.person\_id; |

## Module 5: Sorting and Filtering Data

### Demo C1: How to Sort Data

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to sort data,  -- from dbo.movie table  -- Using ASC ascending order  SELECT \*  FROM movie  ORDER BY popularity;    -- Using DESC descending order  SELECT \*  FROM movie  ORDER BY budget DESC; |

### Demo C2: How to Filter Data with Predicates

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to filter data with predicates,  -- from dbo.language and dbo.movie table  SELECT \*  FROM language  WHERE language\_id = 24574;    SELECT \*  FROM movie  WHERE movie\_id > 300000 AND budget >= 50000000; |

### Demo C3: How to Filter Data with TOP and OFFSET-FETCH

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to filter data using TOP and OFFSET\_FETCH,  -- from dbo.movie table  -- Using TOP  SELECT TOP 7 \*  FROM movie  ORDER BY revenue DESC;    -- Using OFFSET-FETCH  SELECT \*  FROM movie  ORDER BY budget DESC  OFFSET 7 ROWS  FETCH NEXT 5 ROWS ONLY; |

### 

### Demo C4: How to work with Unknown Values

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query to work with Unknown values,  -- from dbo.movie table  SELECT  CASE  WHEN budget IS NULL THEN 'Unknown'  ELSE budget  END AS budget,  CASE  WHEN popularity IS NULL THEN 'Unknown'  ELSE popularity  END AS popularity,  CASE  WHEN revenue IS NULL THEN 'Unknown'  ELSE revenue  END AS revenue,  CASE  WHEN runtime IS NULL THEN 'Unknown'  ELSE runtime  END AS runtime  FROM movie; |

## Module 6: Working with Data Types

### Demo D1: Working with Data Type examples

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Working with Data Type  -- Step 1. Create a table  CREATE TABLE Film\_Name (  movie\_id INT,  title VARCHAR(200),  popularity DECIMAL(10, 6),  vote\_average DECIMAL (10, 2),  release\_date DATE  );  -- Step 2. Insert data into the Film\_Name table  INSERT INTO Film\_Name(movie\_id, title, popularity, vote\_average, release\_date)  VALUES  (1, 'Avatar', 290.945555, 65.50, '2024-02-19'),  (2, 'Star Wars', 126.393695, 44.44, '1977-05-25'),  (3, 'Forrest Gump', 138.133331, 37.89, '1994-07-06')  -- Step 3. Query the Film\_Name table  SELECT \* FROM Film\_Name; |

### Demo D2: Working with Character Data

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Working with Character Data  -- Step 1. Create a table  CREATE TABLE Movie\_Actor (  movie\_actor\_id INT,  movie\_actor\_name VARCHAR(100),  country VARCHAR(50),  languages CHAR(30),  film\_acted TEXT  );  -- Step 2. Insert data into the Movie\_Actor table  INSERT INTO Movie\_Actor(movie\_actor\_id, movie\_actor\_name, country, languages, film\_acted)  VALUES (1, 'Jason Statam', 'United States of America', 'English', 'The Beekeeper'),  (2, 'Shah Rukh Khan', 'India', 'Hindi', 'Asoka'),  (3, 'Chris Hemsworth', 'Australia', 'English', 'Furiosa: A mad maga saga')  -- Step 3. Query the Movie\_Actor table  SELECT \* FROM Movie\_Actor; |

### Demo D3: Working with Date and Time Data

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Working with Data and Time Data  -- Step 1. Create a table  CREATE TABLE Movie\_Purchase (  purchase\_id INT,  movie\_id INT,  purchase\_date DATE,  purchase\_time TIME  );  -- Step 2. Insert data into the Order table  INSERT INTO Movie\_Purchase (purchase\_id, movie\_id, purchase\_date, purchase\_time)  VALUES (1, 6, '2021-02-12', '10:30:00'),  (2, 5, '2021-09-24', '15:24:30'),  (3, 1, '2022-01-10', '07:12:48')  -- Step 3. Query the Order table  SELECT \* FROM Movie\_Purchase; |

## Module 7: Using DML to Modify Data

Why use Using DML to Modify Data?

DML is an abbreviation for Data Manipulation Language. Represents a collection of programming languages explicitly used to make changes to the data

### Demo E1: Adding Data to Tables

### 

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Step 1. Select and execute the following query to see all data in dbo.department table  SELECT \* FROM department    -- Step 2. Select and execute the following query to add data to dbo.department table  INSERT INTO department(department\_id, department\_name)  VALUES  (13, 'Script Writing'),  (14, 'VFX Editing')    -- Step 3. Select and execute the following query to see all data in dbo.gender table  SELECT \* FROM dbo.gender    -- Step 4. Select and execute the following query to add data to dbo.gender table  INSERT INTO gender(gender\_id, gender)  VALUES  (3, 'Transgender'),  (4, 'Genderqueer'),  (5, 'Cisgender') |

### Demo E2: Modifying and Removing Data

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Modifying data  -- Step 1. Select and execute the following query to see all data of a single row in dbo.movie table  SELECT \* FROM movie WHERE movie)id = 5    -- Step 2. Modify or Update data in a single row in dbo.movie table  UPDATE movie  SET homepage = 'http://www.fourrooms.com/movie/four-rooms/', popularity = 25.900000  WHERE movie\_id = 5;    -- Removing data  -- Step 3. Select and execute the following query to see all data in dbo.movie\_actor table  SELECT \* FROM Movie\_Actor    -- Step 4. Execute the following query to Remove Data from the movie\_actor table  DELETE FROM Movie\_Actor  WHERE languages = 'English'; |

### Demo E3: Generating Automatic Column Values

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Step 1. Select and execute the following query to Generate Automatic column values,  CREATE TABLE movie\_rating (  rating\_id INT IDENTITY(1,1) PRIMARY KEY,  rating DECIMAL(4, 2),  rating\_source VARCHAR(100),  );  -- Step 2. Inserting data into the movie table  INSERT INTO movie\_rating (rating, rating\_source)  VALUES (3.5, 'Imdb'),  (5.0, 'rottentomatoe'),  (4.9, 'Metacritic'); |

## Module 8: Using Built-In Functions

Why do programmers use built in functions?

TSQL and programming languages use functions. The biggest reasons are functions allow you to do calculation and break programming into more manageable pieces.

### Demo F1: Writing Queries with Built-In Functions

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query using built-in functions,  -- 1. Aggregate Function  -- a. COUNT(\*)  SELECT COUNT(\*) AS movie\_rows FROM movie;    -- b. SUM(column\_name)  SELECT SUM(budget) AS total\_budget FROM movie;    -- c. AVG(column\_name)  SELECT AVG(vote\_count) AS avg\_vote\_count FROM movie;    -- d. MIN(column\_name)  SELECT MIN(popularity) AS minimum\_popularity FROM movie;    -- e. MAX(column\_name)  SELECT MAX(revenue) AS maximum\_revenue FROM movie;    -- 2. String Function  -- a. CONCAT()  SELECT CONCAT(title, ' ', 'was budgeted at', ' ', budget) AS full\_information FROM movie;    -- b. UPPER(column\_name)  SELECT UPPER(character\_name) AS character\_name FROM movie\_cast;    -- c. LOWER(column\_name)  SELECT LOWER(job) AS job FROM movie\_crew; |

### 

### Demo F2: Using Conversion Functions

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query using conversion functions,  -- 1. CAST and CONVERT Functions  -- a. CAST Function  SELECT CAST(vote\_average AS INT) AS vote\_average FROM movie    -- b. CONVERT Function  SELECT CONVERT(INT, runtime) AS runtime FROM movie; |

### Demo F3: Using Logical Functions

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query using logical functions,  -- 1. Logical Functions  -- a. ISNULL Function  SELECT ISNULL(homepage, 'Unknown') AS homepage FROM movie;    -- b. COALESCE Function  SELECT COALESCE(vote\_average, vote\_count, 0) AS vote\_average FROM movie;    -- c. CASE Function  SELECT  title,  CASE  WHEN vote\_count > 5000 THEN 'High'  ELSE 'Low'  END AS vote\_count\_flag  FROM movie; |

### Demo F4: Using Functions to Work with NULL

|  |
| --- |
| **TSQL Demo Code and SSMS Screenshot Results or Evidence:** |
| USE movies;  GO  -- Select and execute the following query using functions to work with NULL,  -- 1. ISNULL Function  SELECT homepage FROM movie WHERE homepage IS NULL;    -- b. IS NOT NULL Function  SELECT homepage FROM movie WHERE homepage IS NOT NULL;    -- c. CASE Function  SELECT  title,  CASE  WHEN revenue IS NULL THEN 'No Revenue'  ELSE 'Revenue Exists'  END AS revenue\_flag  FROM movie; |