Systems Design and Databases (CIS1018-N)

Week 7

Sorting and Filtering Data

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				

Overview - Sorting Data

- Filtering Data with Predicates
- Filtering Data with TOP and OFFSET-FETCH
- Working with Unknown Values

Sorting Data

- Using the ORDER BY Clause
- ORDER BY Clause Syntax
- ORDER BY Clause Examples
- Demonstration: Sorting Data

Using the ORDER BY Clause

- ORDER BY sorts rows in results for presentation purposes
  - No guaranteed order of rows without ORDER BY
  - Use of ORDER BY guarantees the sort order of the result
  - Last clause to be logically processed
  - Sorts all NULLs together
- ORDER BY can refer to:
  - Columns by name, alias or ordinal position (not recommended)
  - Columns not part of SELECT list
    - Unless DISTINCT specified
- Declare sort order with ASC or DESC

ORDER BY Clause Syntax

- Writing ORDER BY using column names:

```
SELECT <select list>
FROM <table source>
ORDER BY <column1_name>, <column2_name>;
```

- Writing ORDER BY using column aliases:

```
SELECT <column> AS <alias>
FROM <table source>
ORDER BY <alias1>, <alias2>;
```

- Specifying sort order in the ORDER BY clause:

```
SELECT <column> AS <alias>
FROM <table source>
ORDER BY <column_name|alias> ASC|DESC;
```

ORDER BY Clause Examples

- ORDER BY with column names:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate;
```

- ORDER BY with column alias:

```
SELECT orderid, custid, YEAR(orderdate) AS orderyear
FROM Sales.Orders
ORDER BY orderyear;
```

- ORDER BY with descending order:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate DESC;
```

Use ORDER BY Clause

- Use ORDER BY Clause - **Ascending Order**

  (Ascending Order is by default use ASC or not)

```
SELECT custid, orderdate
FROM Sales.Orders
ORDER BY custid ASC;
```

Results | Messages

	custid	orderdate
1	1	2007-08-25 00:00:00.000
2	1	2007-10-03 00:00:00.000
3	1	2007-10-13 00:00:00.000
4	1	2008-01-15 00:00:00.000
5	1	2008-03-16 00:00:00.000
6	1	2008-04-09 00:00:00.000
7	2	2008-03-04 00:00:00.000
8	2	2007-11-28 00:00:00.000
9	2	2007-08-08 00:00:00.000
10	2	2006-09-18 00:00:00.000
11	3	2006-11-27 00:00:00.000
12	3	2007-04-15 00:00:00.000

```
SELECT custid, orderdate
FROM Sales.Orders
ORDER BY orderdate;
```

Results | Messages

	custid	orderdate
1	85	2006-07-04 00:00:00.000
2	79	2006-07-05 00:00:00.000
3	34	2006-07-08 00:00:00.000
4	84	2006-07-08 00:00:00.000
5	76	2006-07-09 00:00:00.000
6	34	2006-07-10 00:00:00.000

Results | Messages

	custid	orderdate
1	91	2006-12-05 00:00:00.000
2	91	2007-07-25 00:00:00.000
3	91	2007-12-23 00:00:00.000
4	91	2008-02-04 00:00:00.000
5	91	2008-02-25 00:00:00.000
6	91	2008-04-03 00:00:00.000
7	91	2008-04-23 00:00:00.000
8	90	2008-04-07 00:00:00.000
9	90	2008-02-26 00:00:00.000
10	90	2008-02-06 00:00:00.000
11	90	2008-02-10 00:00:00.000
12	90	2007-10-07 00:00:00.000

- Use ORDER BY Clause - **Descending Order**

```
SELECT custid, orderdate
FROM Sales.Orders
ORDER BY custid DESC;
```

```
SELECT custid, orderdate
FROM Sales.Orders
ORDER BY orderdate DESC;
```

Results | Messages

	custid	orderdate
1	73	2008-05-06 00:00:00.000
2	68	2008-05-06 00:00:00.000
3	9	2008-05-06 00:00:00.000
4	65	2008-05-06 00:00:00.000
5	44	2008-05-05 00:00:00.000
6	46	2008-05-05 00:00:00.000
7	20	2008-05-05 00:00:00.000
8	58	2008-05-05 00:00:00.000
9	17	2008-05-04 00:00:00.000

Demonstration A with TSQL: Sorting Data

In this demonstration, you will see how to Sort data using the ORDER BY clause

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

SELECT OrderID, CustID, OrderDate, shippeddate,
shipname FROM Sales.Orders
ORDER BY OrderDate;

-- Demo results for sales.orderid on 2006-07-08
SELECT OrderID, CustID, OrderDate, shippeddate,
shipname FROM Sales.Orders
where OrderDate = '2008-06-01'
ORDER BY OrderDate;

-- we are adding 10 & 11 years to the
respective dates.
UPDATE Sales.Orders
SETOrderDate = DATEADD(YEAR, 10, '2008-06-01'),
ShippedDate = DATEADD(YEAR,11, '2008-06-08');

-- Demo results for updating dates to 2018
onwards
SELECT OrderID, CustID, OrderDate, shippeddate,
shipname FROM Sales.Orders
where OrderDate = '2018-06-01'
ORDER BY OrderDate;
```

Results | Messages

	OrderID	CustID	OrderDate	shippeddate	shipname
1	10248	85	2006-07-04 00:00:00.000	2006-07-16 00:00:00.000	Ship to 85-B
2	10249	79	2006-07-05 00:00:00.000	2006-07-10 00:00:00.000	Ship to 79-C
3	10250	34	2006-07-08 00:00:00.000	2006-07-12 00:00:00.000	Destination SCQXA
4	10251	84	2006-07-08 00:00:00.000	2006-07-15 00:00:00.000	Ship to 84-A
5	10252	76	2006-07-09 00:00:00.000	2006-07-11 00:00:00.000	Ship to 76-B
6	10253	34	2006-07-10 00:00:00.000	2006-07-16 00:00:00.000	Destination JPAIY
7	10254	14	2006-07-11 00:00:00.000	2006-07-23 00:00:00.000	Destination YUJRD

Results | Messages

OrderID	CustID	OrderDate	shippeddate	shipname

Results | Messages

	OrderID	CustID	OrderDate	shippeddate	shipname
1	10248	85	2018-06-01 00:00:00.000	2019-06-08 00:00:00.000	Ship to 85-B
2	10249	79	2018-06-01 00:00:00.000	2019-06-08 00:00:00.000	Ship to 79-C
3	10250	34	2018-06-01 00:00:00.000	2019-06-08 00:00:00.000	Destination SCQXA
4	10251	84	2018-06-01 00:00:00.000	2019-06-08 00:00:00.000	Ship to 84-A
5	10252	76	2018-06-01 00:00:00.000	2019-06-08 00:00:00.000	Ship to 76-B

Results | Messages

	OrderID	CustID	ordermonth
1	10248	85	6
2	10249	79	6
3	10250	34	6
4	10251	84	6
5	10252	76	6

Results | Messages

	OrderID	CustID	OrderDate
1	10248	85	2018-06-01 00:00:00.000
2	10249	79	2018-06-01 00:00:00.000
3	10250	34	2018-06-01 00:00:00.000
4	10251	84	2018-06-01 00:00:00.000
5	10252	76	2018-06-01 00:00:00.000

Results | Messages

	City	CustID	CompanyName
1	Warszawa	91	Customer CCFIZ
2	Walla Walla	43	Customer UISOJ
3	Versailles	40	Customer EFFTC
4	Vancouver	42	Customer IAIJK
5	Tsawassen	10	Customer EEALV

Results | Messages

	OrderDate	ShipDate
1	2008-06-01	2008-06-08
2	2008-06-01	2008-06-08
3	2008-06-01	2008-06-08
4	2008-06-01	2008-06-08
5	2008-06-01	2008-06-08

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

-- Sorting by column alias name
SELECT OrderID, CustID, MONTH(OrderDate) AS
ordermonth
FROM Sales.Orders
ORDER BY ordermonth DESC;

-- Sorting by column name in descending order
SELECT OrderID, CustID, OrderDate
FROM Sales.Orders
ORDER BY OrderDate DESC;

-- Changing sort order for multiple columns
SELECT City, CustID, CompanyName
FROM Sales.Customers
ORDER BY City DESC, CustID ASC;

-- Step 6: Revert the changes made to date
columns
UPDATE Sales.Orders
SET OrderDate = '2008-06-01',
shippeddate = '2008-06-08';

-- list the records with changes -update changes
Select OrderDate = '2008-06-01',
ShipDate = '2008-06-08'
from Sales.Orders
```

Demonstration A with AdventureWorksLT2019: Sorting Data

In this demonstration, you will see how to Sort data using the ORDER BY clause

```
-- Demo Queries are below
USE AdventureWorksLT2019;

SELECT SalesOrderID, CustomerID, OrderDate,
ShipDate, DueDate
FROM SalesLT.SalesOrderHeader
ORDER BY OrderDate;

SELECT SalesOrderID, CustomerID, OrderDate,
ShipDate, DueDate
FROM SalesLT.SalesOrderHeader
where SalesOrderID = 71782
ORDER BY OrderDate;

-- we are adding 1, 7 or 14 days to the
respective dates.
UPDATE SalesLT.SalesOrderHeader
SETOrderDate = DATEADD(D, 1, '2008-06-01'),
ShipDate = DATEADD(D,7, '2008-06-08'),
DueDate = DATEADD(D,14,'2008-06-13’);

-- Demo results for salesorderid = 71782 &
Sorting by column name
SELECT SalesOrderID, CustomerID, OrderDate,
ShipDate, DueDate
FROM SalesLT.SalesOrderHeader
where SalesOrderID = 71782
ORDER BY OrderDate;
```

Results | Messages

	SalesOrderID	CustomerID	OrderDate	ShipDate	DueDate
1	71774	29847	2008-06-01 00:00:00.000	2008-06-08 00:00:00.000	2008-06-13 00:00:00.000
2	71776	30072	2008-06-01 00:00:00.000	2008-06-08 00:00:00.000	2008-06-13 00:00:00.000
3	71780	30113	2008-06-01 00:00:00.000	2008-06-08 00:00:00.000	2008-06-13 00:00:00.000
4	71782	29485	2008-06-01 00:00:00.000	2008-06-08 00:00:00.000	2008-06-13 00:00:00.000
5	71783	29957	2008-06-01 00:00:00.000	2008-06-08 00:00:00.000	2008-06-13 00:00:00.000

Results | Messages

	SalesOrderID	CustomerID	OrderDate	ShipDate	DueDate
1	71782	29485	2008-06-01 00:00:00.000	2008-06-08 00:00:00.000	2008-06-13 00:00:00.000

Results | Messages

	SalesOrderID	CustomerID	OrderDate	ShipDate	DueDate
1	71782	29485	2008-06-02 00:00:00.000	2008-06-15 00:00:00.000	2008-06-27 00:00:00.000

Results | Messages

	SalesOrderID	CustomerID	ordermonth
1	71774	29847	6
2	71776	30072	6
3	71780	30113	6
4	71782	29485	6
5	71783	29957	6

Results | Messages

	SalesOrderID	CustomerID	OrderDate
1	71774	29847	2008-06-02 00:00:00.000
2	71776	30072	2008-06-02 00:00:00.000
3	71780	30113	2008-06-02 00:00:00.000
4	71782	29485	2008-06-02 00:00:00.000
5	71783	29957	2008-06-02 00:00:00.000

Results | Messages

	ModifiedDate	CustomerID	CompanyName
1	2009-05-16 16:33:33.123	544	Valley Bicycle Specialists
2	2009-05-16 16:33:33.123	29492	Valley Bicycle Specialists
3	2009-05-16 16:33:33.107	488	Bicycle Accessories and Kits
4	2009-05-16 16:33:33.107	29490	Bicycle Accessories and Kits
5	2009-05-16 16:33:33.090	491	Area Bike Accessories

Results | Messages

	OrderDate	ShipDate	DueDate
1	2008-06-01	2008-06-08	2008-06-13
2	2008-06-01	2008-06-08	2008-06-13
3	2008-06-01	2008-06-08	2008-06-13
4	2008-06-01	2008-06-08	2008-06-13
5	2008-06-01	2008-06-08	2008-06-13

```
-- Demo Queries are below
USE AdventureWorksLT2019;

-- Sorting by column alias name
SELECT SalesOrderID, CustomerID,
MONTH(OrderDate) AS ordermonth
FROM SalesLT.SalesOrderHeader
ORDER BY ordermonth DESC;

-- Sorting by column name in descending order
SELECT SalesOrderID, CustomerID, OrderDate
FROM SalesLT.SalesOrderHeader
ORDER BY OrderDate DESC;

-- Changing sort order for multiple columns
SELECT ModifiedDate, CustomerID, CompanyName
FROM SalesLT.Customer
ORDER BY ModifiedDate DESC, CustomerID ASC;

-- Revert the changes made to date columns
UPDATE SalesLT.SalesOrderHeader
SET OrderDate = '2008-06-01',
ShipDate = '2008-06-08',
DueDate = '2008-06-13’;


-- list the records with changes -update changes
Select OrderDate = '2008-06-01',
ShipDate = '2008-06-08',
DueDate = '2008-06-13'
from SalesLT.SalesOrderHeader
```

Filtering Data with Predicates

- Filtering Data in the WHERE Clause with Predicates
- WHERE Clause Syntax
- Demonstration: Filtering Data with Predicates

Filtering Data in the WHERE Clause with Predicates

- WHERE clauses use predicates
  - Must be expressed as logical conditions
  - Only rows for which predicate evaluates to TRUE are accepted
  - Values of FALSE or UNKNOWN filtered out
- WHERE clause follows FROM, precedes other clauses
  - Can’t see aliases declared in SELECT clause
- Can be optimized by SQL Server to use indexes
- Data filtered server-side
  - Can reduce network traffic and client memory usage

WHERE Clause Syntax

- Filter rows for customers from Spain

```
SELECT contactname, country
FROM Sales.Customers
WHERE country = N'Spain';
```

- Filter rows for orders after July 1, 2007

```
SELECT orderid, orderdate
FROM Sales.Orders
WHERE orderdate > '20070101';
```

- Filter orders within a range of dates

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
WHERE orderdate >= '20070101' AND orderdate < '20080101';
```

Demonstration B with AdventureWorksLT2019 : Filtering Data with Predicates

In this demonstration, you will see how to Filter data in a WHERE clause

- Week7 - Demonstration A - AWLT2019 - Sort with ORDER BY.sql
- Week7 - Demonstration A - TSQL - Sort with ORDER BY.sql
- Week7 - Demonstration B - AAWLT2019 - Filter with WHERE.sql
- Week7 - Demonstration B - TSQL - Filter with WHERE.sql
- Week7 - Demonstration C - AWLT2019 - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration C - TSQL - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration D - AWLT2019 - Handling NULLS.sql
- Week7 - Demonstration D - TSQL- Handling NULLS.sql

Demonstration B with TSQL: Filtering Data with Predicates

In this demonstration, you will see how to Filter data in a WHERE clause

- Week7 - Demonstration A - AWLT2019 - Sort with ORDER BY.sql
- Week7 - Demonstration A - TSQL - Sort with ORDER BY.sql
- Week7 - Demonstration B - AAWLT2019 - Filter with WHERE.sql
- Week7 - Demonstration B - TSQL - Filter with WHERE.sql
- Week7 - Demonstration C - AWLT2019 - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration C - TSQL - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration D - AWLT2019 - Handling NULLS.sql
- Week7 - Demonstration D - TSQL- Handling NULLS.sql

Filtering Data with TOP and OFFSET-FETCH

- Filtering in the SELECT Clause Using the TOP Option
- Filtering in the ORDER BY Clause Using OFFSET-FETCH
- OFFSET-FETCH Syntax
- Demonstration: Filtering Data with TOP and OFFSET-FETCH

Filtering in the SELECT Clause Using the TOP Option

- TOP allows you to limit the number or percentage of rows returned by a query
- Works with ORDER BY clause to limit rows by sort order:
  - If ORDER BY list is not unique, results are not deterministic (no single correct result set)
  - Modify ORDER BY list to ensure uniqueness, or use TOP WITH TIES
- Added to SELECT clause:
  - SELECT TOP (N) | TOP (N) Percent
    - With percent, number of rows rounded up (nondeterministic)
  - SELECT TOP (N) WITH TIES
    - Retrieve duplicates where applicable (deterministic)
- TOP is proprietary to Microsoft SQL Server

Results | Messages

	ProductName	Price
1	Bicycle 1	258.2
2	Bicycle 2	265.3
3	Bicycle 3	267.8
4	Bicycle 5	267.9
5	Bicycle 6	267.9

} TOP 4

} WITH TIES

Filtering in the ORDER BY Clause Using OFFSET-FETCH

OFFSET-FETCH is an extension to the ORDER BY clause:

- Allows filtering a requested range of rows
  - Dependent on ORDER BY clause
- Provides a mechanism for paging through results
- Specify number of rows to skip, number of rows to retrieve:

```
ORDER BY <order_by_list>
OFFSET <offset_value> ROW(S)
FETCH FIRST|NEXT <fetch_value> ROW(S) ONLY
```

- Available in SQL Server 2012, 2014, and 2019
  - Provides more compatibility than TOP

OFFSET-FETCH Syntax

- OFFSET value must be supplied
  - May be zero if no skipping is required
- The optional FETCH clause allows all rows following the OFFSET value to be returned
- Natural Language approach to code:
  - ROW and ROWS interchangeable
  - FIRST and NEXT interchangeable
  - ONLY optional—makes meaning clearer to human reader
- OFFSET value and FETCH value may be constants or expressions, including variables and parameters

```
OFFSET <offset_value> ROW|ROWS
FETCH FIRST|NEXT <fetch_value> ROW|ROWS [ONLY]
```

Demonstration C with AdventureWorksLT2019: Filtering Data with TOP and OFFSET-FETCH

In this demonstration, you will see how to Filter data using TOP and OFFSET-FETCH

- Week7 - Demonstration A - AWLT2019 - Sort with ORDER BY.sql
- Week7 - Demonstration A - TSQL - Sort with ORDER BY.sql
- Week7 - Demonstration B - AAWLT2019 - Filter with WHERE.sql
- Week7 - Demonstration B - TSQL - Filter with WHERE.sql
- Week7 - Demonstration C - AWLT2019 - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration C - TSQL - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration D - AWLT2019 - Handling NULLS.sql
- Week7 - Demonstration D - TSQL- Handling NULLS.sql

Demonstration C TSQL:
Filtering Data with TOP and OFFSET-FETCH

In this demonstration, you will see how to Filter data using TOP and OFFSET-FETCH

- Week7 - Demonstration A - AWLT2019 - Sort with ORDER BY.sql
- Week7 - Demonstration A - TSQL - Sort with ORDER BY.sql
- Week7 - Demonstration B - AAWLT2019 - Filter with WHERE.sql
- Week7 - Demonstration B - TSQL - Filter with WHERE.sql
- Week7 - Demonstration C - AWLT2019 - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration C - TSQL - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration D - AWLT2019 - Handling NULLS.sql
- Week7 - Demonstration D - TSQL- Handling NULLS.sql

Working with Unknown Values

- Three-Valued Logic
- Handling NULL in Queries
- Demonstration: Working with NULL

Three-Valued Logic

- SQL Server uses NULLs to mark missing values
  - NULL can be "missing but applicable" or "missing but inapplicable"
    - Customer middle name: Not supplied, or doesn't have one?
- With no missing values, predicate outputs are TRUE or FALSE only (5 > 2, 1=1)
- With missing values, outputs can be TRUE, FALSE or UNKNOWN (NULL > 99, NULL = NULL)
- Predicates return UNKNOWN when comparing missing value to another value, including another missing value

Handling NULL in Queries

- Different components of SQL Server handle NULL differently
  - Query filters (ON, WHERE, HAVING) filter out UNKNOWNs
  - CHECK constraints accept UNKNOWNS
  - ORDER BY, DISTINCT treat NULLs as equals
- Testing for NULL
  - Use **IS NULL** or **IS NOT NULL** rather than = NULL or <> NULL

```
SELECT custid, city, region, country
FROM Sales.Customers
WHERE region IS NOT NULL;
```

"N" prefix stands for National Language

- The "N" prefix stands for National Language in the SQL-92 standard,
- You may see it in old TSQL and must be uppercase.
- If you do not prefix a Unicode string constant with N,
- SQL Server will convert it to the non-Unicode code page of the Current database before it uses the string.
- It is of no relevance, but you may come across the convention in Industry

```
SELECT CustId, ContactTitle, ContactName, companyname, Region
FROM Sales.Customers
WHERE region <> N'A.'
ORDER BY ContactName;
```

Demonstration C with AdventureWorksLT2019: Working with NULL

In this demonstration, you will see how to Test for NULL

- Week7 - Demonstration A - AWLT2019 - Sort with ORDER BY.sql
- Week7 - Demonstration A - TSQL - Sort with ORDER BY.sql
- Week7 - Demonstration B - AAWLT2019 - Filter with WHERE.sql
- Week7 - Demonstration B - TSQL - Filter with WHERE.sql
- Week7 - Demonstration C - AWLT2019 - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration C - TSQL - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration D - AWLT2019 - Handling NULLS.sql
- Week7 - Demonstration D - TSQL- Handling NULLS.sql

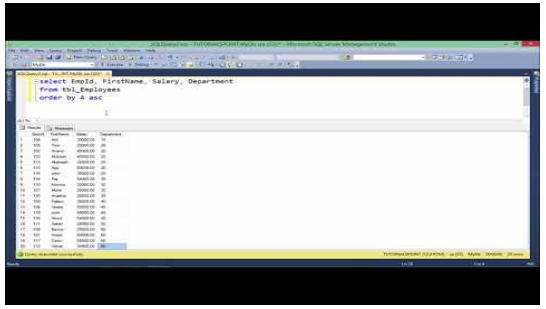
Demonstration C with TSQL: Working with NULL

In this demonstration, you will see how to Test for NULL

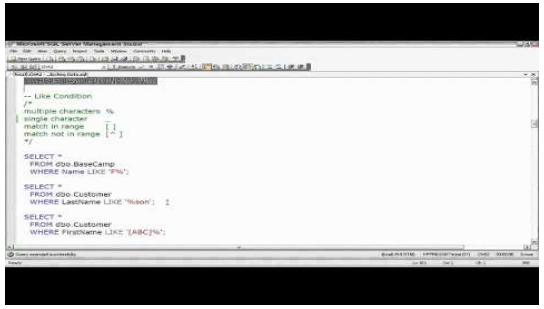
- Week7 - Demonstration A - AWLT2019 - Sort with ORDER BY.sql
- Week7 - Demonstration A - TSQL - Sort with ORDER BY.sql
- Week7 - Demonstration B - AAWLT2019 - Filter with WHERE.sql
- Week7 - Demonstration B - TSQL - Filter with WHERE.sql
- Week7 - Demonstration C - AWLT2019 - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration C - TSQL - Filtering with Top OffsetFetch.sql
- Week7 - Demonstration D - AWLT2019 - Handling NULLS.sql
- Week7 - Demonstration D - TSQL- Handling NULLS.sql

SQL Sort & Filter Video link:

- T-SQL - Sorting Data



Sql Server Filtering Data with the Where Clause



- SQL Order By - Sorting

SQL Filter

Supporting Material 1/3

Sorting Data

- ORDER BY – sort the result set based on values in a specified list of columns

Filtering Data

- DISTINCT – select distinct values in one or more columns of a table.
- WHERE – filter rows in the output of a query based on one or more conditions.
- AND – combine two Boolean expressions and return true if all expressions are true.
- OR– combine two Boolean expressions and return true if either of conditions is true.
- IN – check whether a value matches any value in a list or a subquery.
- BETWEEN – test if a value is between a range of values.
- LIKE – check if a character string matches a specified pattern.
- Column & table aliases – show you how to use column aliases to change the heading of the query output and table alias to improve the readability of a query.

Supporting Material 2/3

- Microsoft Doc | SELECT - ORDER BY Clause (Transact-SQL), WHERE (Transact-SQL)
- W3Schools | SQL ORDER BY Keyword, SQL WHERE Clause
- SQL Server Tutorial.net | SQL Server ORDER BY, SQL WHERE Clause
- Tutorialpoints | SQL - SORTING, SQL WHERE Clause
- JavaTpoint | SQL ORDER BY Clause, SQL WHERE

Supporting Material 3/3

- Supporting online resources
- MSDN SQL Server – Select From Where Group By - Order By Statement
- MSDN SQL Server – Update Statement
- MSDN SQL Server Date & Time Functions
- MSDN SQL Server – Set Operators
- `-- date support: https://www.w3schools.com/sqL/func_sqlserver_dateadd.asp`
- `-- update support: https://www.w3schools.com/SQL/sql_update.asp`
- `-- set Operators %: https://www.w3schools.com/sql/sql_operators.asp`