The background of the slide features a collage of business and technology-related icons. On the left, there is a large, glowing cylindrical database icon. To its right, a list of business terms is displayed: Innovation, Branding, Solution, Marketing, Analysis, Ideas, Success, and Management. Further right, a hand is shown drawing a lightbulb and a flowchart. Below these, there are various icons including a person, a world map, a network diagram, a puzzle piece, and a pie chart. The entire background is overlaid with a large, semi-transparent watermark that reads 'For Intech Centre Use Only'.

# Intelligent Data Management with SQL Server

**Session: 9**

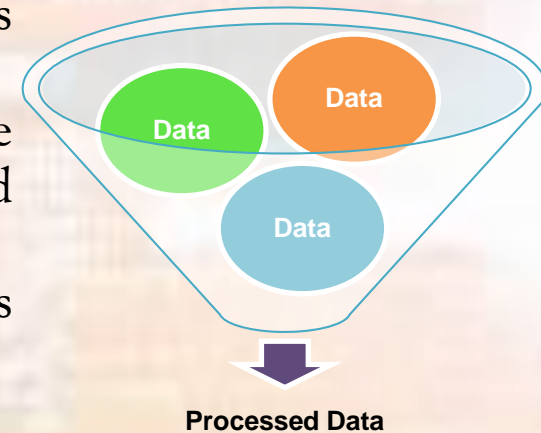
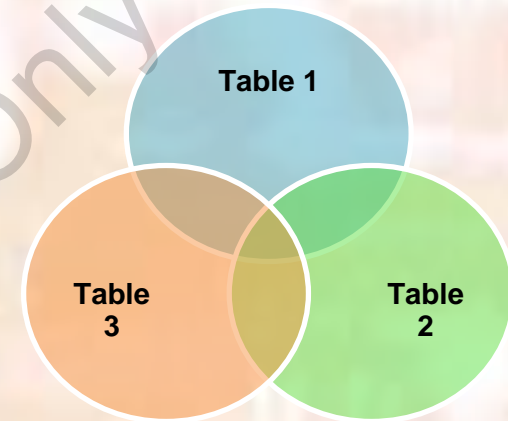
## *Advanced Queries and Joins*

# Objectives

- Explain grouping and aggregating data
- Describe subqueries
- Describe table expressions
- Explain Joins
- Describe various types of Joins
- Explain the use of various set operators to combine data
- Describe pivoting and grouping set operations

# Introduction

- SQL Server 2019 includes several powerful query features that help to retrieve data efficiently and quickly.
- Data can be grouped and/or aggregated together in order to present summarized information.
- Joins help to combine column data from two or more tables based on a logical relationship between the tables.
- Set operators such as UNION and INTERSECT combine row data from two or more tables.
- PIVOT and UNPIVOT operators are used to transform the orientation of data from column-oriented to row-oriented and vice versa.
- GROUPING SET subclause of the GROUP BY clause helps to specify multiple groupings in a single query.



# Grouping Data 1-5

This clause partitions the resultset into one or more subsets and each subset has values and expressions in common.

- It is followed by a list of columns, known as grouped columns.
- It restricts the number of rows of the resultset.

	WorkOrderID	TotalHoursPerWorkOrder
1	13	17.6000
2	14	17.6000
3	15	4.0000
4	16	4.0000
5	17	4.0000
6	18	4.0000
7	19	4.0000
8	20	4.0000
9	21	4.0000
10	22	4.0000

Using GROUP BY Clause

# Grouping Data 2-5

## GROUP BY with WHERE

- To restrict the rows for grouping.
- The rows satisfy the search condition are considered for grouping.
- The rows that do not meet the conditions in the WHERE clause are eliminated before any grouping is done.

	WorkOrderID	TotalHoursPerWorkOrder
1	13	17.6000
2	14	17.6000
3	15	4.0000
4	16	4.0000
5	17	4.0000
6	18	4.0000
7	19	4.0000
8	20	4.0000

Output of GROUP BY with WHERE



# Grouping Data 3-5

## GROUP BY with NULL

- If the grouping column contains a NULL value, that row becomes a separate group in the resultset.
- If the grouping column contains more than one NULL value, the NULL values are put into a single row.

	Class	AverageListPrice
1	NULL	16.314
2	H	1679.4964
3	L	370.6887
4	M	635.5816

Output of GROUP BY with NULL

# Grouping Data 4-5

## GROUP BY with ALL

- The ALL keyword can also be used with the GROUP BY clause.
- It includes all the groups that the GROUP BY clause produces and even those which do not meet the search conditions.

	Group	TotalSales
1	Europe	13590506.0212
2	North America	33182889.0168
3	Pacific	NULL

Output of GROUP BY with ALL

# Grouping Data 5-5

## GROUP BY with HAVING

- HAVING clause is used only with SELECT statement to specify a search condition for a group.
- The HAVING clause acts as a WHERE clause in places where the WHERE clause cannot be used against aggregate functions such as SUM().

### Syntax:

```
SELECT <column_name> FROM <table_name> GROUP BY <column_name> HAVING  
<search_condition>
```



# Summarizing Data 1-4

➤ GROUP BY clause also uses operators such as CUBE and ROLLUP to return summarized data.

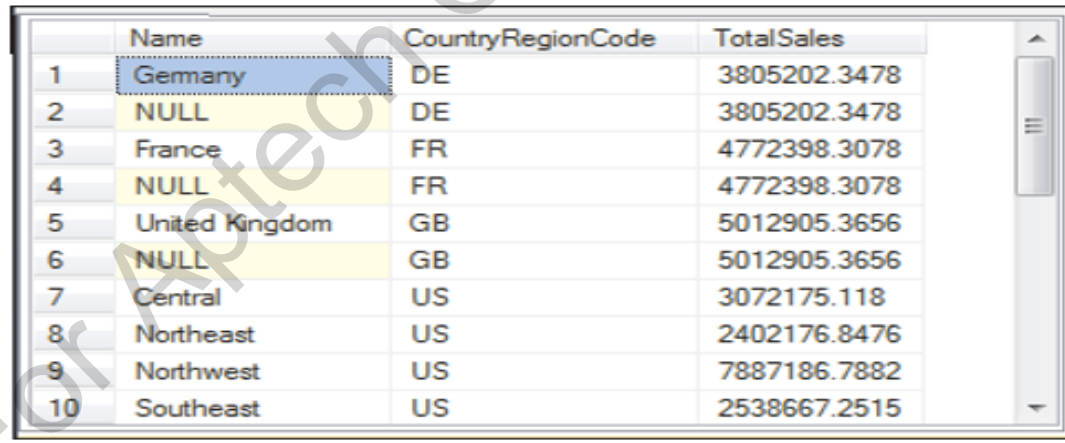
➤ Number of columns in the GROUP BY clause determines number of summary rows in the resultset.

# Summarizing Data 2-4

## CUBE:

CUBE is an aggregate operator that produces a super-aggregate row.

In addition to usual rows provided by the GROUP BY, it also provides the summary of rows that the GROUP BY clause generates.



	Name	CountryRegionCode	TotalSales
1	Germany	DE	3805202.3478
2	NULL	DE	3805202.3478
3	France	FR	4772398.3078
4	NULL	FR	4772398.3078
5	United Kingdom	GB	5012905.3656
6	NULL	GB	5012905.3656
7	Central	US	3072175.118
8	Northeast	US	2402176.8476
9	Northwest	US	7887186.7882
10	Southeast	US	2538667.2515

Using GROUP BY with CUBE

# Summarizing Data 3-4

## ROLLUP:

- It introduces summary rows into the resultset.
  - Generates a resultset that shows groups arranged in a hierarchical order.
  - It arranges the groups from the lowest to the highest.
- 
- ROLLUP assumes a hierarchy among the dimension columns and only generates grouping sets based on this hierarchy.
  - ROLLUP is often used to generate subtotals and totals for reporting purposes.
  - ROLLUP is commonly used to calculate the aggregates of hierarchical data such as sales by year → quarter → month.

# Summarizing Data 4-4

	Name	CountryRegionCode	TotalSales
1	Central	US	3072175.118
2	Central	NULL	3072175.118
3	France	FR	4772398.3078
4	France	NULL	4772398.3078
5	Germany	DE	3805202.3478
6	Germany	NULL	3805202.3478
7	Northeast	US	2402176.8476
8	Northeast	NULL	2402176.8476
9	Northwest	US	7887186.7882
10	Northwest	NULL	7887186.7882
11	Southeast	US	2538667.2515
12	Southeast	NULL	2538667.2515
13	Southwest	US	10510853.8739
14	Southwest	NULL	10510853.8739
15	United Kingdom	GB	5012905.3656
16	United Kingdom	NULL	5012905.3656
17	NULL	NULL	40001565.9004

Using GROUP BY with ROLLUP

# Aggregate Functions 1-3

Developers require to perform analysis across rows, such as counting rows, meeting specific criteria, or summarizing total sales for all orders.

Aggregate functions enable to accomplish it.

Aggregate functions ignore NULLs, except when using COUNT(\*).

Aggregate functions in a SELECT list do not generate a column alias.

Aggregate functions in a SELECT clause operate on all rows passed to the SELECT phase.



# Aggregate Functions 2-3

Function Name	Syntax	Description
AVG	AVG(<expression>)	Calculates the average of all the non-NULL numeric values in a column.
COUNT or COUNT_BIG	COUNT(*) or COUNT(<expression>)	When (*) is used, this function counts all rows, including those with NULL. The function returns count of non- NULL rows for the column when a column is specified as <expression>. The return value of COUNT function is an int. The return value of COUNT_BIG is a big_int.
MAX	MAX(<expression>)	Returns the largest number, latest date/time, or last occurring string.
MIN	MIN(<expression>)	Returns the smallest number, earliest date/time, or first occurring string.
SUM	SUM(<expression>)	Calculates the sum of all the non-NULL numeric values in a column.

# Aggregate Functions 3-3

Results Messages			
	AvgUnitPrice	MinQty	MaxDiscount
1	465.0934	1	0.40

Using Aggregate Functions

Results Messages		
	Earliest	Latest
1	2011-05-31 00:00:00.000	2014-06-30 00:00:00.000

Using Aggregate Functions with Non-Numeric Data

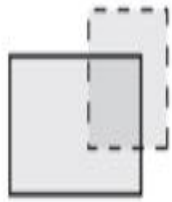
# Spatial Aggregates 1-3

- SQL Server provides several methods that help to aggregate two individual items of geometry or geography data.

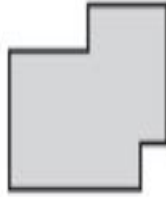
Method	Description
STUnion	Returns an object that represents the union of a geometry/geography instance with another geometry/geography instance.
STIntersection	Returns an object that represents the points where a geometry/geography instance intersects another geometry/geography instance.
STConvexHull	Returns an object representing the convex hull of a geometry/geography instance. A set of points is called convex if for any two points, the entire segment is contained in the set. The convex hull of a set of points is the smallest convex set containing the set. For any given set of points, there is only one convex hull.

Spatial Aggregate Methods

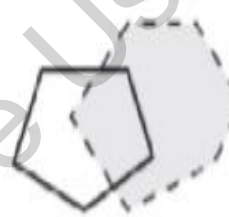
# Spatial Aggregates 2-3



Combining two polygons  
using the `STUnion()` method  
results in a merged polygon.



`STUnion()`



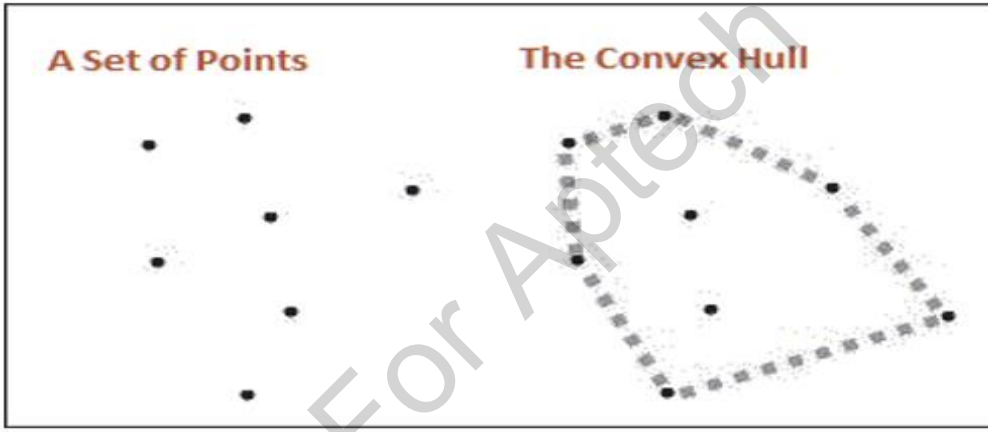
Using `STIntersection()` with  
two polygons can lead to  
another polygon.



`STIntersection()`

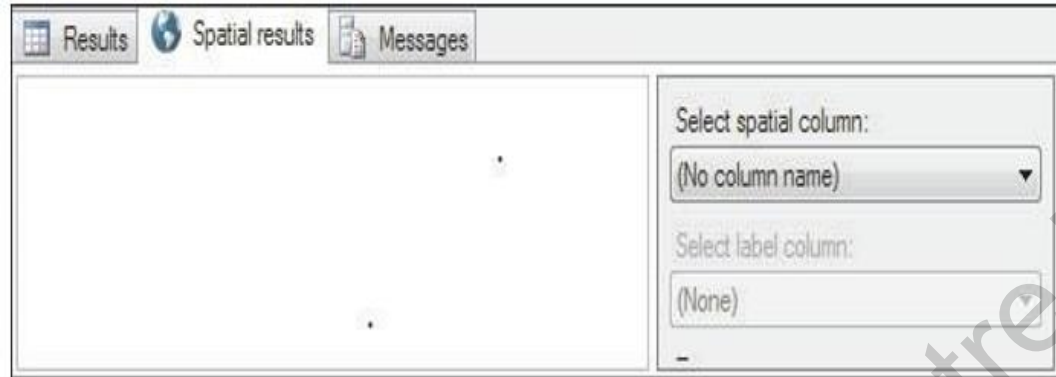
A Set of Points

The Convex Hull

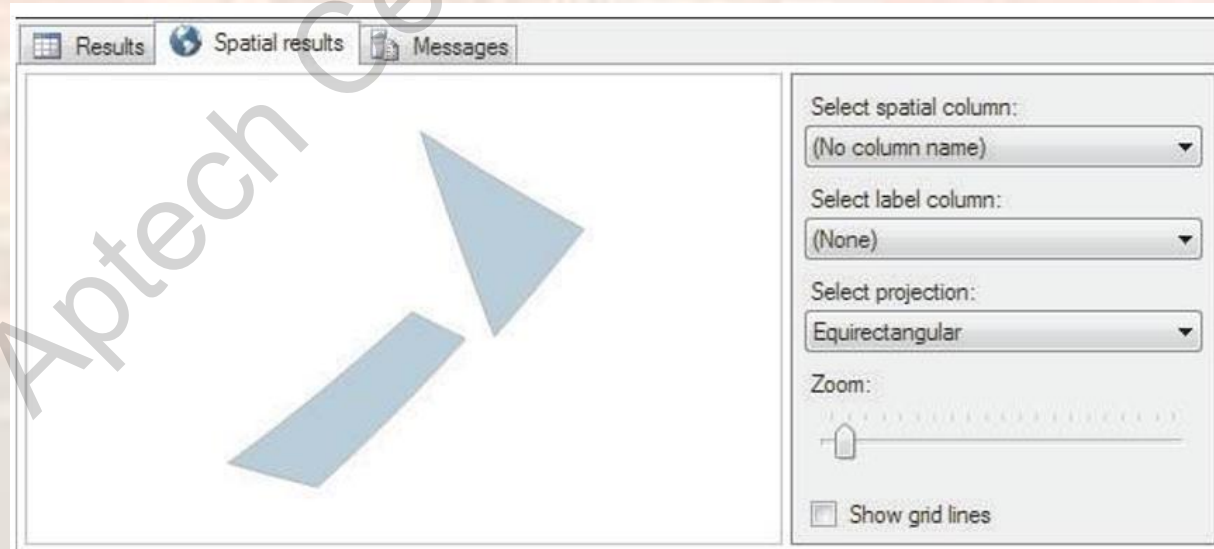


`STConvexHull()`

# Spatial Aggregates 3-3



Using STUnion() with a geometry Type



Using STUnion() with a geography Type



# More Spatial Aggregates 1-5

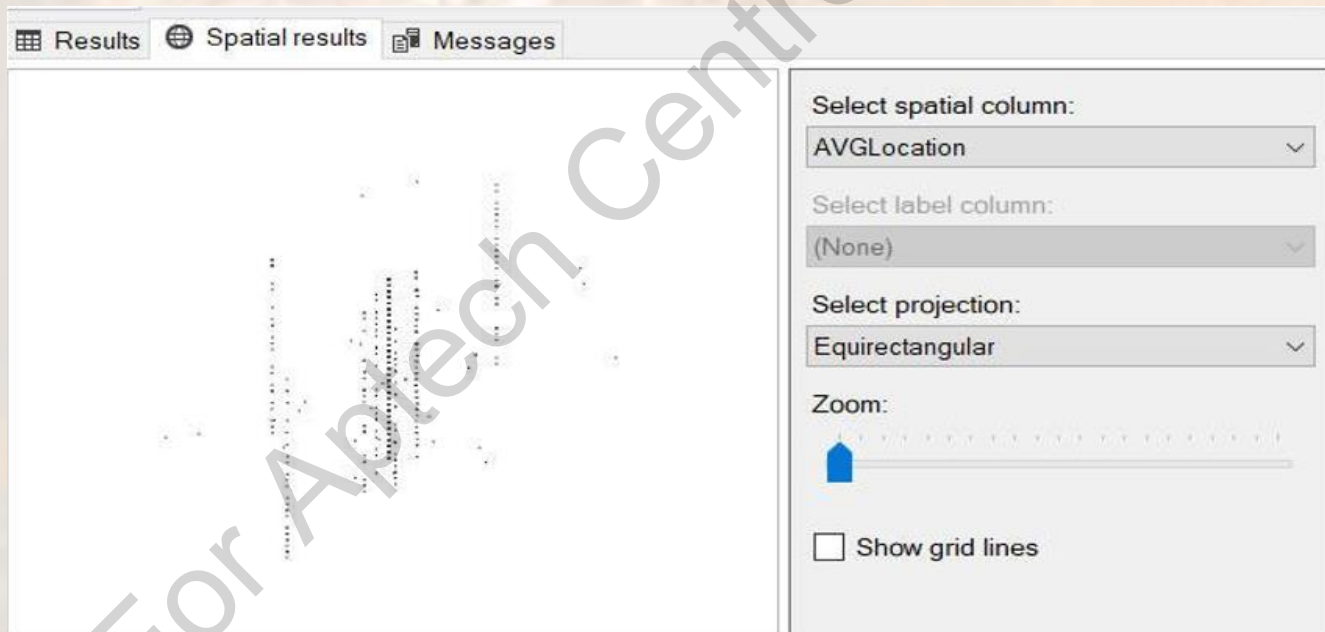


- These aggregates are implemented as static methods, which work for either geography or geometry data types.

# More Spatial Aggregates 2-5

## Union Aggregate

- It performs a union operation on a set of geometry objects.
- It combines multiple spatial objects into a single spatial object, removing interior boundaries, where applicable.

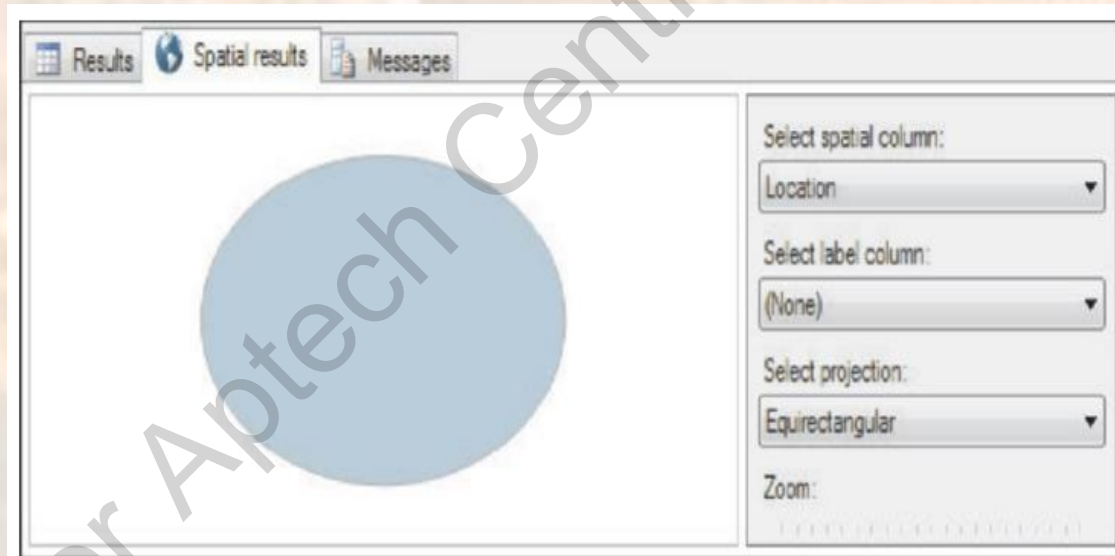


Viewing Spatial Results

# More Spatial Aggregates 3-5

## Envelope Aggregate

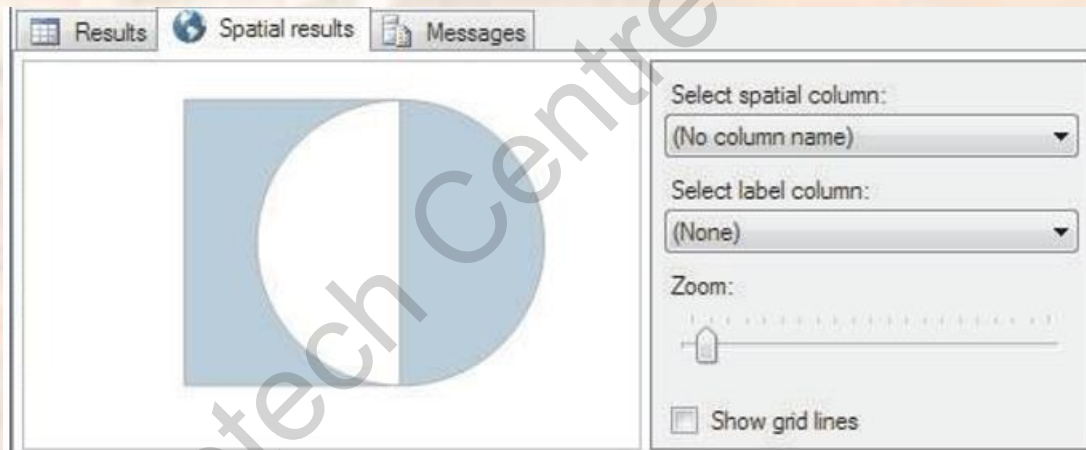
- The Envelope Aggregate returns a bounding area for a given set of geometry or geography objects.
- It exhibits different behaviors for geography and geometry types.



# More Spatial Aggregates 4-5

## Collection Aggregate

- It returns a GeometryCollection/GeographyCollection instance with one geometry/geography part for each spatial object(s) in the selection set.

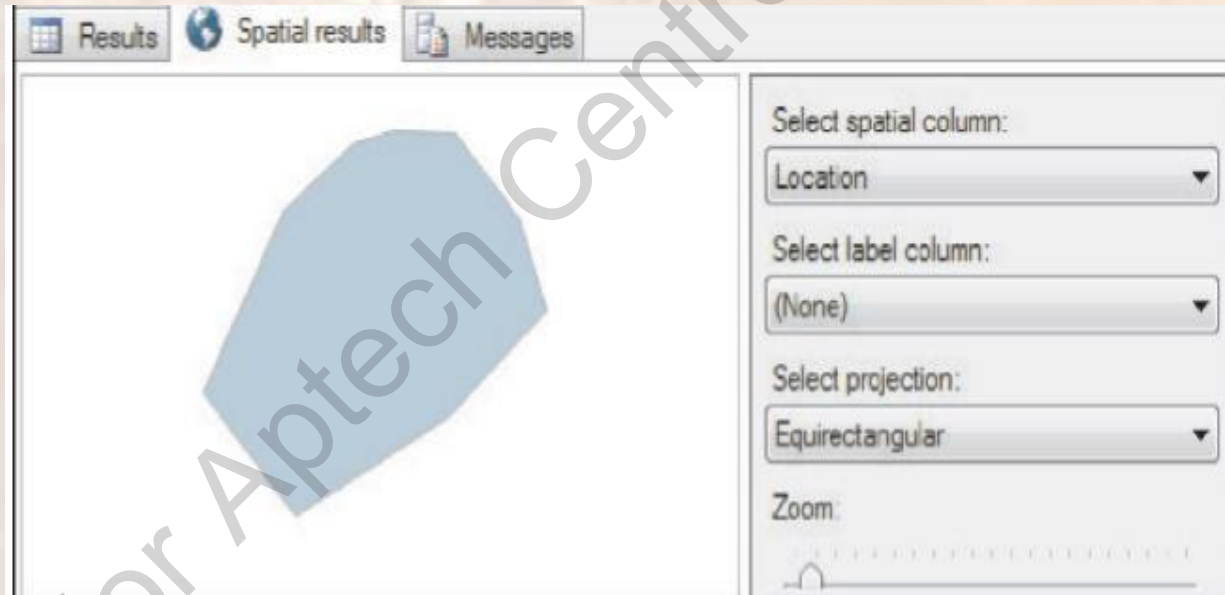


Using CollectionAggregate

# More Spatial Aggregates 5-5

## Convex Hull Aggregate

- It returns a convex hull polygon, which encloses one or more spatial objects for a given set of geometry/geography objects.

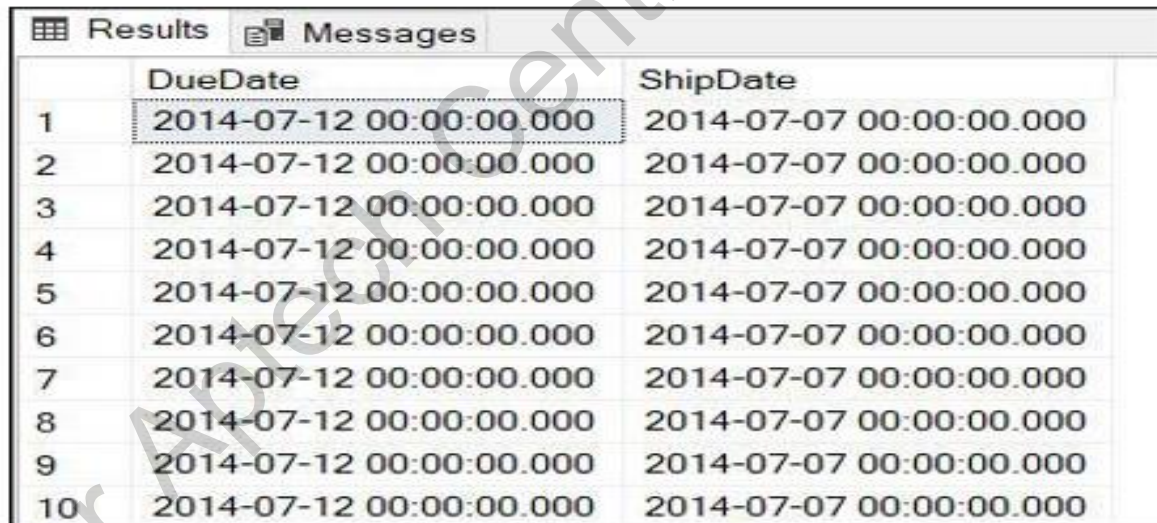


Using ConvexHullAggregate



# Subqueries 1-2

- The outer query is called parent query and the inner query is called a subquery.
- The purpose of a subquery is to return results to the outer query.
- In other words, the inner query statement should return the column or columns used in the criteria of the outer query statement.



	DueDate	ShipDate
1	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
2	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
3	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
4	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
5	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
6	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
7	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
8	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
9	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000
10	2014-07-12 00:00:00.000	2014-07-07 00:00:00.000

Using a Simple Subquery

# Subqueries 2-2

Based on results returned by inner query, a subquery can be classified as a **scalar subquery** or a **multi-valued subquery**:

Scalar subqueries return a single value. Here, the outer query must be written to process a single result.

Multi-valued subqueries return a result similar to a single-column table. Here, the outer query must be written to handle multiple possible results.

# Working with Multi-valued Queries

The ntext, text, and image data types cannot be used in the SELECT list of subqueries.

The SELECT list of a subquery introduced with a comparison operator can have only one expression or column name.

Subqueries that are introduced by a comparison operator not followed by the keyword ANY or ALL cannot include GROUP BY and HAVING clauses.

You cannot use DISTINCT keyword with subqueries that include GROUP BY.

You can specify ORDER BY only when TOP is also specified.

For Aptech  
Centre Use Only

# Nested Subqueries

- A subquery that is defined inside another subquery is called a nested subquery.

For example:

Retrieve and display the names of persons from Canada

Results		Messages
	LastName	FirstName
1	Vargas	Garrett
2	Saraiva	José

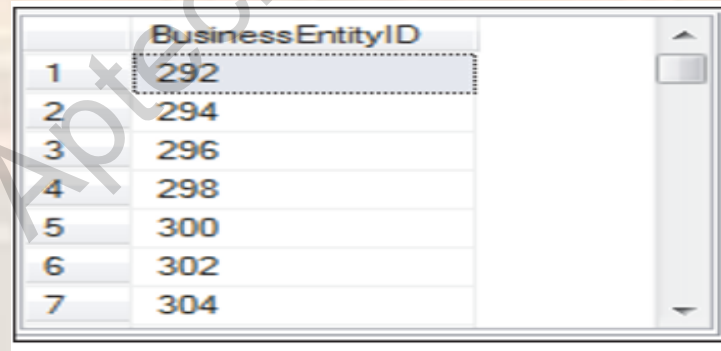
Output of Nested Subqueries

# Correlated Queries

- When a subquery takes parameters from its parent query, it is known as Correlated subquery.

For example:

Retrieve all the business entity ids of persons whose contact information was last modified not earlier than 2019.



A screenshot of a database query result window. It displays a table with two columns: an index from 1 to 7 and a column labeled 'BusinessEntityID'. The values in the 'BusinessEntityID' column are 292, 294, 296, 298, 300, 302, and 304. The first row is highlighted with a dashed border.

	BusinessEntityID
1	292
2	294
3	296
4	298
5	300
6	302
7	304

Output of Correlated Queries



# Joins

- Specifying the column from each table to be used for the join. A typical join specifies a foreign key from one table and its associated key in the other table.

- Specifying a logical operator such as =, <> to be used in comparing values from the columns.

	FirstName	LastName	Job Title
1	Ken	Sánchez	Chief Executive Officer
2	Teri	Duffy	Vice President of Engineering
3	Roberto	Tamburello	Engineering Manager
4	Rob	Walters	Senior Tool Designer
5	Gail	Erickson	Design Engineer
6	Jossef	Goldberg	Design Engineer
7	Dylan	Miller	Research and Development Manager

Output of Join

Three types of joins:

Inner Joins

Outer Joins

Self-Joins

# Inner Join

- An inner join is formed when records from two tables are combined only if the rows from both the tables are matched based on a common column

Following is the syntax of an inner join:

```
SELECT <ColumnName1>, <ColumnName2>...<ColumnNameN> FROM Table_A  
AS Table_Alias_A  
INNER JOIN  
Table_B AS Table_Alias_B ON  
Table_Alias_A.<CommonColumn>=Table_Alias_B.<CommonColumn>
```

# Outer Join

- Outer joins are join statements that return all rows from at least one of the tables specified in the FROM clause, as long as those rows meet any WHERE or HAVING conditions of the SELECT statement.

Two types of commonly used outer joins



# Self-Join

- A self-join is used to find records in a table that are related to other records in the same table.
- A table is joined to itself in a self-join.

Results		Messages		
	ProductID	Color	Name	Name
1	317	Black	LL Crankarm	LL Crankarm
2	317	Black	LL Crankarm	ML Crankarm
3	317	Black	LL Crankarm	HL Crankarm
4	317	Black	LL Crankarm	Chainring
5	317	Black	LL Crankarm	HL Road Frame - Black, 58
6	317	Black	LL Crankarm	Sport-100 Helmet, Black
7	317	Black	LL Crankarm	Road-750 Black, 44
8	317	Black	LL Crankarm	Road-750 Black, 48
9	317	Black	LL Crankarm	Road-750 Black, 52
10	317	Black	LL Crankarm	Road-750 Black, 58

Self Join Example

# MERGE Statement

Insert a new row from the source if the row is missing in the target table

Update a target row if a record already exists in the source table

Delete a target row if the row is missing in the source table

- Compare last and first names of customers from both source and target tables
- Update customer information in target table if the last and first names match
- Insert new records in target table if the last and first names in source table do not exist in target table
- Delete existing records in target table if the last and first names do not match with those of source table

# Common Table Expressions (CTEs)

- A CTE is similar to a temporary resultset defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is a named expression defined in a query.
- A CTE that include references to itself is called a **recursive CTE**.

CTEs are limited in scope to the execution of the outer query. Hence, when the outer query ends, the lifetime of the CTE will end.

You must define a name for a CTE and also, define unique names for each of the columns referenced in the SELECT clause of the CTE.

It is possible to use inline or external aliases for columns in CTEs.

A single CTE can be referenced multiple times in the same query with one definition.



# Combining Data Using SET Operators

- SQL Server 2019 provides certain keywords, also called as operators, to combine data from multiple tables.

These operators are as follows:

- UNION
- INTERSECT
- EXCEPT

# UNION Operator

- Results from two different query statements can be combined into a single resultset using UNION operator.
  - Query statements must have compatible column types and equal number of columns.
- The column names can be different in each statement, but the data types must be compatible.
  - By compatible data types, it means that it should be possible to convert the contents of one of the columns into another.

Following is the syntax of the UNION operator.

```
Query_Statement1 UNION [ALL] Query_Statement2
```

# INTERSECT Operator

- The INTERSECT operator is used with two query statements to return a distinct set of rows that are common to both the query statements.

The basic rules for using INTERSECT are as follows:

- Number of columns and order in which they are given must be same in both queries.
- Data types of the columns being used must be compatible.

# EXCEPT Operator

- The EXCEPT operator returns all of the distinct rows from the query given on left of the EXCEPT operator and removes all rows from the resultset that match rows on right of the EXCEPT operator.

# Pivoting and Grouping Set Operations

➤ The process of transforming data from a row-based orientation to a column-based orientation is called pivoting.

➤ The PIVOT and UNPIVOT operators of SQL Server help to change the orientation of data from column-oriented to row-oriented and vice versa.

# PIVOT Operator

## Grouping

In the `FROM` clause, the input columns must be provided. The `PIVOT` operator uses those columns to determine which column(s) to use for grouping the data for aggregation.

## Spreading

Here, a comma-separated list of values that occur in the source data is provided that will be used as the column headings for the pivoted data.

## Aggregation

An aggregation function, such as `SUM`, to be performed on the grouped rows.

	TotalSalesYTD	Name
1	7887186.7882	Northwest
2	2402176.8476	Northeast
3	3072175.118	Central
4	10510853.8739	Southwest
5	2538667.2515	Southeast

Grouping without PIVOT



# UNPIVOT Operator

Source columns to be unpivoted

A name for the new column that will display the unpivoted values

A name for the column that will display the names of the unpivoted values

Results		Messages
	SalesYear	TotalSales
1	2011	151706131.5475
2	2012	862786335.1754
3	2013	1190722789.0552
4	2014	391255200.8993

Output for UNPIVOT

# EXCEPT Operator

- The EXCEPT operator returns all the distinct rows from the query given on left of the EXCEPT operator and removes all rows from the resultset that match rows on right of the EXCEPT operator.

# Summary

- The GROUP BY clause and aggregate functions enable to group and/or aggregate data together in order to present summarized information.
- Spatial aggregate functions were first introduced in SQL Server 2012 and are supported in SQL Server 2019 as well.
- A subquery allows the resultset of one SELECT statement to be used as criteria for another SELECT statement.
- Joins help you to combine column data from two or more tables based on a logical relationship between the tables.
- Set operators such as UNION and INTERSECT help you to combine row data from two or more tables.
- The PIVOT and UNPIVOT operators help to change the orientation of data from column-oriented to row-oriented and vice versa.
- The GROUPING SET subclause of the GROUP BY clause helps to specify multiple groupings in a single query.