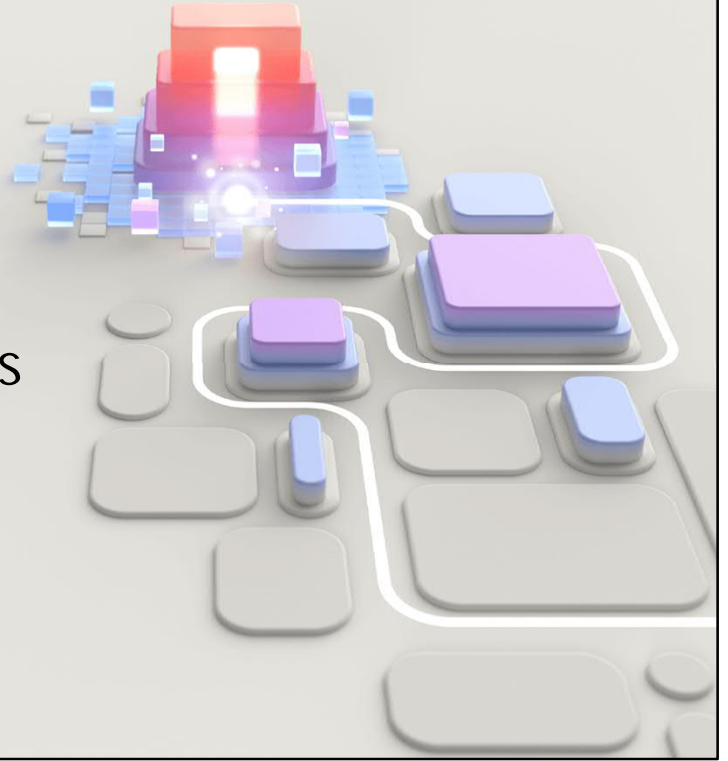


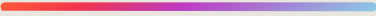
6: Using built-in functions

© Copyright Microsoft Corporation. All rights reserved.



<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/>

Agenda



- Getting started with scalar functions
- Grouping aggregated results

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/1-introduction>

1: Getting started with scalar functions

© Copyright Microsoft Corporation. All rights reserved.



Introduction to built-in functions

Function category	Description
Scalar	Operate on a single row, return a single value
Logical	Compare multiple values to determine a single output
Ranking	Operate on a partition (set) of rows
Rowset	Return a virtual table that can be used subsequently in a Transact-SQL statement
Aggregate	Take one or more input values, return a single summarizing value

Note: Ranking and Rowset functions are advanced features of SQL Server and will not be covered in this course.

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/2-categorize-functions>

Scalar functions

Operate on elements from a single row as inputs, return a single value as output

- Return a single (scalar) value
- Can be used like an expression in queries
- May be deterministic or non-deterministic

```
SELECT UPPER(ProductName) AS Product,  
       ROUND(ListPrice, 0) AS ApproxPrice,  
       YEAR(SaleStartDate) AS SoldSince  
FROM Production.Product;
```

Scalar function categories

- Configuration
- Conversion
- Cursor
- Date and Time
- Mathematical
- Metadata
- Security
- String
- System
- System Statistical
- Text and Image

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/3-scalar-functions>

Scalar functions (cont...)

```
SELECT AVG(ListPrice) AS AveragePrice,  
       MIN(ListPrice) AS MinimumPrice,  
       MAX(ListPrice) AS MaximumPrice  
FROM Production.Product;
```

```
SELECT AVG(ListPrice) AS AveragePrice,  
       MIN(ListPrice) AS MinimumPrice,  
       MAX(ListPrice) AS MaximumPrice  
FROM Production.Product;  
WHERE ProductCategoryID = 15;
```

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/3-scalar-functions>

Nested Scalar functions (cont...)

SQL

```
SELECT MIN(YEAR(OrderDate)) AS Earliest,  
       MAX(YEAR(OrderDate)) AS Latest  
FROM Sales.SalesOrderHeader;
```

SQL

```
SELECT COUNT(DISTINCT CustomerID) AS UniqueCustomers  
FROM Sales.SalesOrderHeader;
```

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/3-scalar-functions>

Logical functions

Output is determined by comparative logic

IIF

- Evaluate logical expression, return first value if true and second value if false

```
SELECT AddressType,  
       IIF(AddressType = 'Main Office', 'Billing', 'Mailing') AS UseFor  
FROM Sales.CustomerAddress;
```

CHOOSE

- Return value based ordinal position of expression in 1-based list

```
SELECT SalesOrderID, Status,  
       CHOOSE(Status, 'Ordered', 'Shipped', 'Delivered') AS OrderStatus  
FROM Sales.SalesOrderHeader;
```

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/3-scalar-functions>

Aggregate functions

Functions that operate on sets, or rows of data

- Summarize input rows
- Without GROUP BY clause, all rows are arranged as one group

```
SELECT COUNT(*) AS OrderLines,  
       SUM(OrderQty*UnitPrice) AS TotalSales  
FROM Sales.OrderDetail;
```



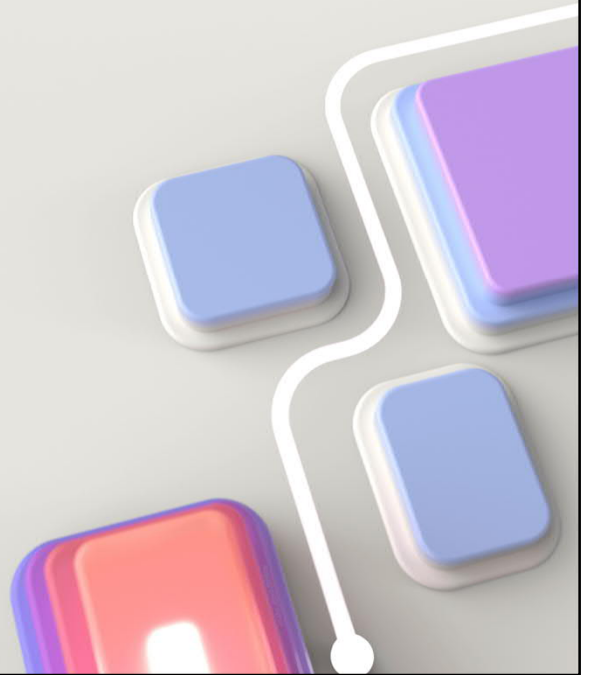
OrderLines	TotalSales
542	714002.9136

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/5-aggregate-functions>

2: Grouping aggregated results

© Copyright Microsoft Corporation. All rights reserved.



Grouping with GROUP BY

- GROUP BY creates groups for output rows, according to unique combination of values specified in the GROUP BY clause
- GROUP BY calculates a summary value for aggregate functions in subsequent phases
- Detail rows are not available after GROUP BY clause is processed

```
SELECT CustomerID, COUNT(*) AS OrderCount
FROM Sales.SalesOrderHeader
GROUP BY CustomerID;
```

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/6-summarize-group-by>

After the GROUP BY clause has been processed and each row has been associated with a group, later phases of the query must aggregate any elements of the source rows that are in the SELECT list but that don't appear in the GROUP BY list. This requirement will have an impact on how you write your SELECT and HAVING clauses.

So, what's the difference between writing the query with a GROUP BY or a DISTINCT? If all you want to know is the distinct values for **CustomerID**, there is no difference. But with GROUP BY, we can add other elements to the SELECT list that are then aggregated for each group.

Filtering groups with HAVING

- HAVING clause provides a search condition that each group must satisfy
- WHERE clause is processed before GROUP BY, HAVING clause is processed after GROUP BY

```
SELECT CustomerID, COUNT(*) AS Orders
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
HAVING COUNT(*) > 10;
```

The clauses in a SELECT statement are applied in the following order:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

© Copyright Microsoft Corporation. All rights reserved.

<https://learn.microsoft.com/en-us/training/modules/use-built-functions-transact-sql/7-filter-groups>

Column Aliases in Aggregated Queries

- Column aliases are assigned in the SELECT clause, which occurs *after* the GROUP BY clause but *before* the ORDER BY clause.
- You can reference a column alias in the ORDER BY clause, but not in the GROUP BY clause.

© Copyright Microsoft Corporation. All rights reserved.

SQL

```
SELECT CustomerID AS Customer,  
       COUNT(*) AS OrderCount  
FROM Sales.SalesOrderHeader  
GROUP BY Customer  
ORDER BY Customer;
```

```
SELECT CustomerID AS Customer,  
       COUNT(*) AS OrderCount  
FROM Sales.SalesOrderHeader  
GROUP BY CustomerID  
ORDER BY Customer;
```

```
SELECT CustomerID AS Customer, COUNT(*) AS OrderCount FROM Sales.SalesOrderHeader GROUP BY Customer  
ORDER BY Customer;
```

```
SELECT CustomerID AS Customer, COUNT(*) AS OrderCount FROM Sales.SalesOrderHeader GROUP BY CustomerID  
ORDER BY Customer;
```

Troubleshooting GROUP BY errors

Msg 8120, Level 16, State 1,
Line 2 Column
<column_name> is invalid in
the select list because it is not
contained in either an
aggregate function or the
GROUP BY clause.

© Copyright Microsoft Corporation. All rights reserved.

```
SELECT CustomerID,  
       COUNT(*) AS OrderCount  
FROM Sales.SalesOrderHeader  
GROUP BY CustomerID;
```

```
SELECT CustomerID,  
       PurchaseOrderNumber,  
       COUNT(*) AS OrderCount  
FROM Sales.SalesOrderHeader  
GROUP BY CustomerID;
```

ERROR! PurchaseOrderNumber isn't
part of the GROUP BY, and it isn't used
with an aggregate function.

Troubleshooting GROUP BY errors

```
SELECT CustomerID,  
       PurchaseOrderNumber,  
       COUNT(*) AS OrderCount  
FROM Sales.SalesOrderHeader  
GROUP BY CustomerID,  
         PurchaseOrderNumber;
```

© Copyright Microsoft Corporation. All rights reserved.

Lab: Using built-in functions



- <https://microsoftlearning.github.io/dp-080-Transact-SQL/Instructions/Labs/04-built-in-functions.html>
- Use scalar functions
- Use logical functions
- Use aggregate functions
- Group aggregated results with GROUP BY clause
- Filter groups with the HAVING clause

© Copyright Microsoft Corporation. All rights reserved.

<https://microsoftlearning.github.io/dp-080-Transact-SQL/Instructions/Labs/04-built-in-functions.html>

Review



- 1 Which OrderState value does this query return for rows with a Status value of 2:

```
SELECT OrderNo, CHOOSE(Status, 'Ordered', 'Shipped', 'Delivered') AS OrderState FROM Sales.Order;
```

 - ☒ Shipped
 - ☐ Delivered
 - ☐ NULL

- 2 Which query returns the number of customers in each city?
 - ☐ SELECT City, COUNT(*) AS CustomerCount FROM Sales.Customer;
 - ☒ SELECT City, COUNT(*) AS CustomerCount FROM Sales.Customer GROUP BY City;
 - ☐ SELECT City, COUNT(*) AS CustomerCount FROM Sales.Customer ORDER BY City;

- 3 Which query returns a row for each category with an average price over 10.00?
 - ☐ SELECT Category, AVG(Price) FROM Store.Product WHERE AVG(Price) > 10.00;
 - ☐ SELECT Category, AVG(Price) FROM Store.Product GROUP BY Category WHERE AVG(Price) > 10.00;
 - ☒ SELECT Category, AVG(Price) FROM Store.Product GROUP BY Category HAVING AVG(Price) > 10.00;

© Copyright Microsoft Corporation. All rights reserved.

Use the slide animation to reveal the correct answers.



© Copyright Microsoft Corporation. All rights reserved.