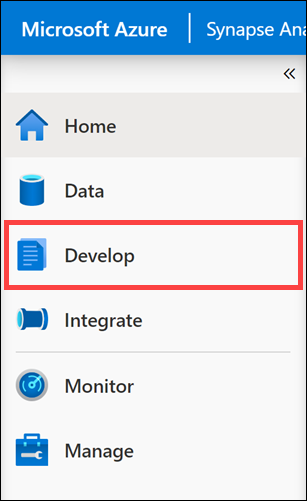
We are going to discuess about below concept:

* Check for skewed data and space usage
* Understand column store storage details
* Study the impact of materialized views
* Explore rules for minimally logged operations

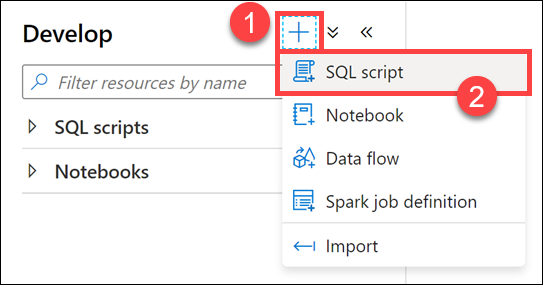
**Check for skewed data and space usage**

**Analyze the space used by tables**

1. Open Synapse Studio.
2. Select the **Develop** hub.



1. From the **Develop** menu, select the **+** button **(1)** and choose **SQL Script (2)** from the context menu.



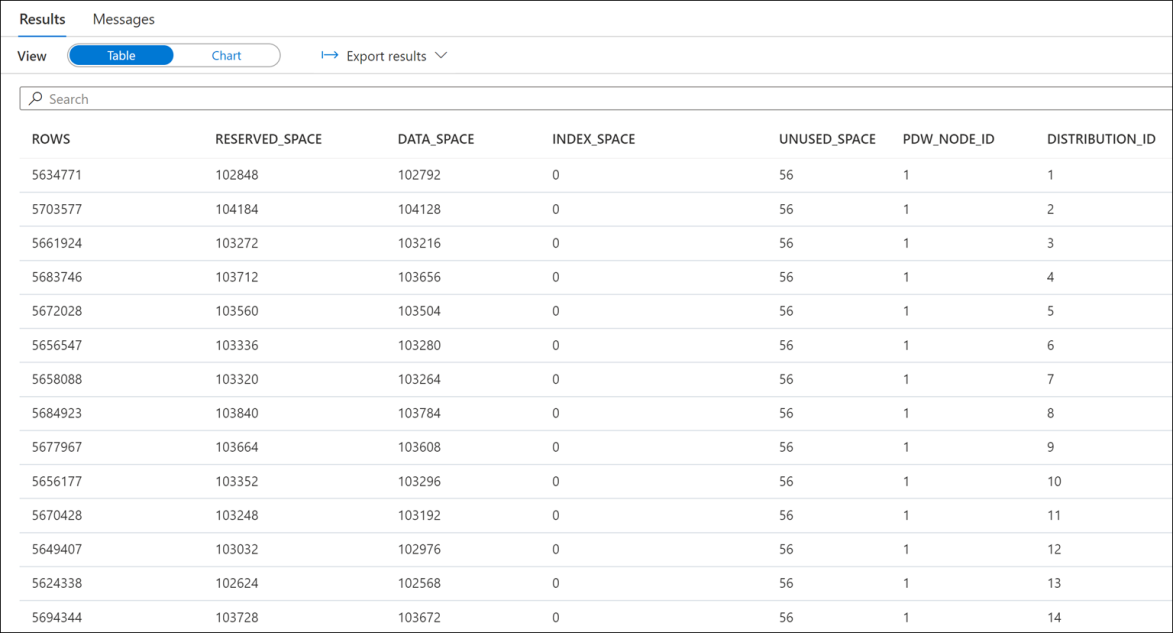
1. In the toolbar menu, connect to the **SQLPool01** database to execute the query.

The connect to option is highlighted in the query toolbar.

1. In the query window, replace the script with the following Database Console Command (DBCC):

SQLCopy

DBCC PDW\_SHOWSPACEUSED('wwi\_perf.Sale\_Hash');



1. Analyze the number of rows in each distribution. Those numbers should be as even as possible. You can see from the results that rows are equally distributed across distributions. Let's dive a bit more into this analysis. Use the following query to get customers with the most sale transaction items:

SQLCopy

SELECT TOP 1000

CustomerId,

count(\*) as TransactionItemsCount

FROM

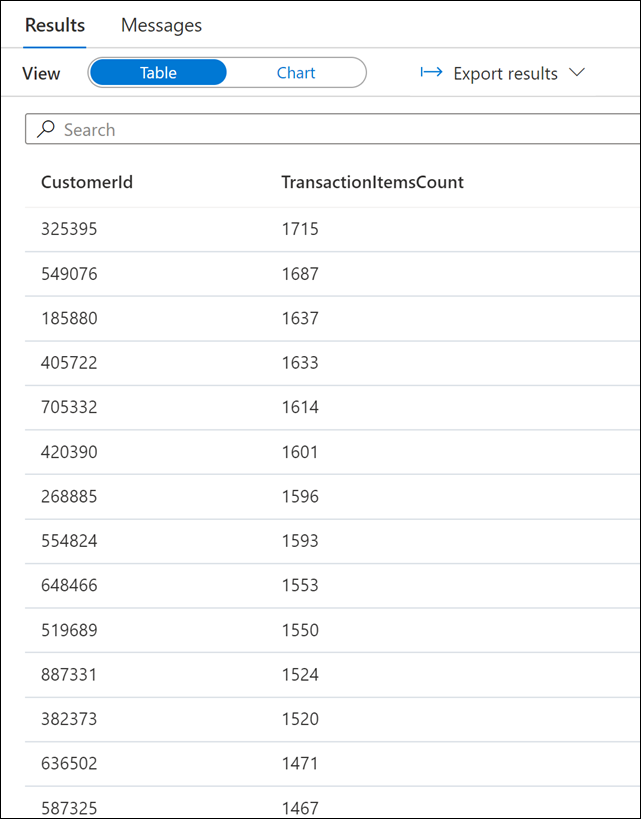
[wwi\_perf].[Sale\_Hash]

GROUP BY

CustomerId

ORDER BY

count(\*) DESC



Now find the customers with the least sale transaction items:

SQLCopy

SELECT TOP 1000

CustomerId,

count(\*) as TransactionItemsCount

FROM

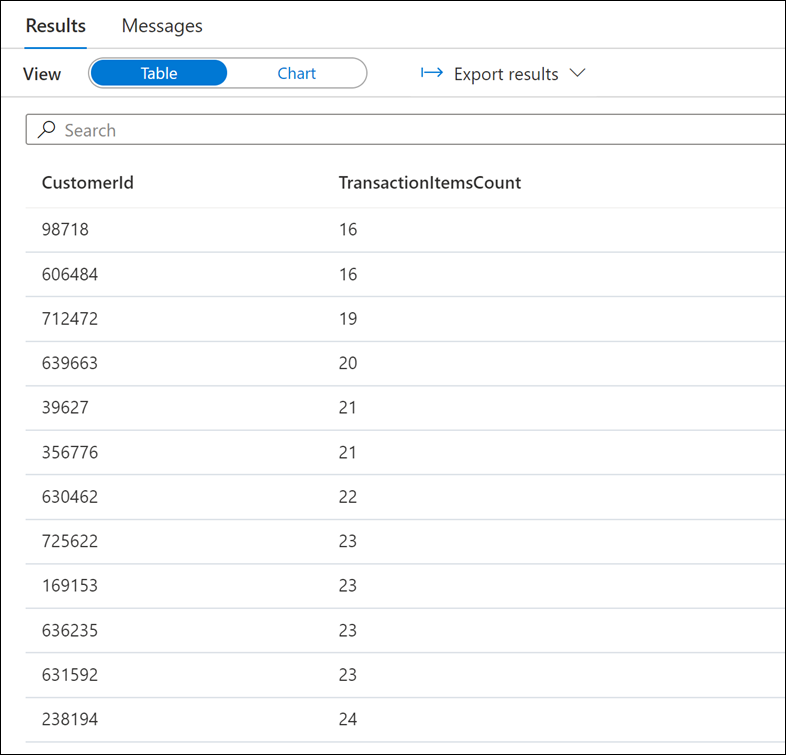
[wwi\_perf].[Sale\_Hash]

GROUP BY

CustomerId

ORDER BY

count(\*) ASC



Notice the largest number of transaction items is 1715 and the smallest is 16.

Let's find now the distribution of per-customer transaction item counts. Run the following query:

SQLCopy

SELECT

T.TransactionItemsCountBucket

,count(\*) as CustomersCount

FROM

(

SELECT

CustomerId,

(count(\*) - 16) / 100 as TransactionItemsCountBucket

FROM

[wwi\_perf].[Sale\_Hash]

GROUP BY

CustomerId

) T

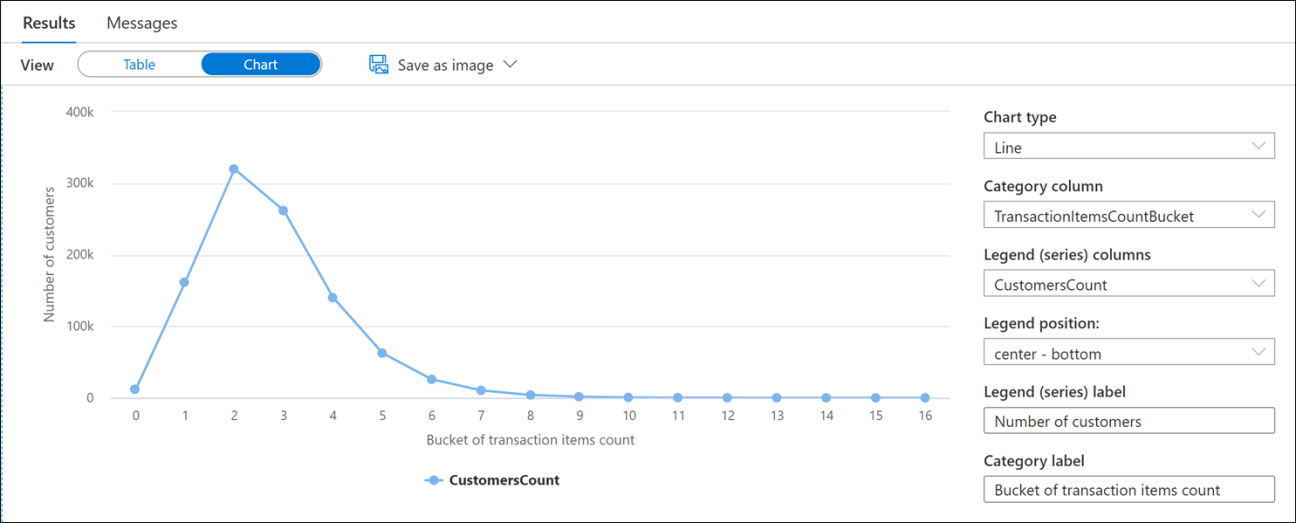
GROUP BY

T.TransactionItemsCountBucket

ORDER BY

T.TransactionItemsCountBucket

In the Results pane, switch to the Chart view and configure it as follows (see the options set on the right side):



Without diving too much into the mathematical and statistical aspects of it, this histogram displays the reason why there is virtually no skew in the data distribution of the Sale\_Hash table. If you haven't figured it out yet, the reason we are talking about is the quasi-normal distribution of the per-customer transaction items counts.

**Use a more advanced approach to understand table space usage**

1. Run the following script to create the vTableSizes view:

CREATE VIEW [wwi\_perf].[vTableSizes]

AS

WITH base

AS

(

SELECT

GETDATE() AS [execution\_time]

, DB\_NAME() AS [database\_name]

, s.name AS [schema\_name]

, t.name AS [table\_name]

, QUOTENAME(s.name)+'.'+QUOTENAME(t.name) AS [two\_part\_name]

, nt.[name] AS [node\_table\_name]

, ROW\_NUMBER() OVER(PARTITION BY nt.[name] ORDER BY (SELECT NULL)) AS [node\_table\_name\_seq]

, tp.[distribution\_policy\_desc] AS [distribution\_policy\_name]

, c.[name] AS [distribution\_column]

, nt.[distribution\_id] AS [distribution\_id]

, i.[type] AS [index\_type]

, i.[type\_desc] AS [index\_type\_desc]

, nt.[pdw\_node\_id] AS [pdw\_node\_id]

, pn.[type] AS [pdw\_node\_type]

, pn.[name] AS [pdw\_node\_name]

, di.name AS [dist\_name]

, di.position AS [dist\_position]

, nps.[partition\_number] AS [partition\_nmbr]

, nps.[reserved\_page\_count] AS [reserved\_space\_page\_count]

, nps.[reserved\_page\_count] - nps.[used\_page\_count] AS [unused\_space\_page\_count]

, nps.[in\_row\_data\_page\_count]

+ nps.[row\_overflow\_used\_page\_count]

+ nps.[lob\_used\_page\_count] AS [data\_space\_page\_count]

, nps.[reserved\_page\_count]

- (nps.[reserved\_page\_count] - nps.[used\_page\_count])

- ([in\_row\_data\_page\_count]

+ [row\_overflow\_used\_page\_count]+[lob\_used\_page\_count]) AS [index\_space\_page\_count]

, nps.[row\_count] AS [row\_count]

FROM

sys.schemas s

INNER JOIN sys.tables t

ON s.[schema\_id] = t.[schema\_id]

INNER JOIN sys.indexes i

ON t.[object\_id] = i.[object\_id]

AND i.[index\_id] <= 1

INNER JOIN sys.pdw\_table\_distribution\_properties tp

ON t.[object\_id] = tp.[object\_id]

INNER JOIN sys.pdw\_table\_mappings tm

ON t.[object\_id] = tm.[object\_id]

INNER JOIN sys.pdw\_nodes\_tables nt

ON tm.[physical\_name] = nt.[name]

INNER JOIN sys.dm\_pdw\_nodes pn

ON nt.[pdw\_node\_id] = pn.[pdw\_node\_id]

INNER JOIN sys.pdw\_distributions di

ON nt.[distribution\_id] = di.[distribution\_id]

INNER JOIN sys.dm\_pdw\_nodes\_db\_partition\_stats nps

ON nt.[object\_id] = nps.[object\_id]

AND nt.[pdw\_node\_id] = nps.[pdw\_node\_id]

AND nt.[distribution\_id] = nps.[distribution\_id]

LEFT OUTER JOIN (select \* from sys.pdw\_column\_distribution\_properties where distribution\_ordinal = 1) cdp

ON t.[object\_id] = cdp.[object\_id]

LEFT OUTER JOIN sys.columns c

ON cdp.[object\_id] = c.[object\_id]

AND cdp.[column\_id] = c.[column\_id]

WHERE pn.[type] = 'COMPUTE'

)

, size

AS

(

SELECT

[execution\_time]

, [database\_name]

, [schema\_name]

, [table\_name]

, [two\_part\_name]

, [node\_table\_name]

, [node\_table\_name\_seq]

, [distribution\_policy\_name]

, [distribution\_column]

, [distribution\_id]

, [index\_type]

, [index\_type\_desc]

, [pdw\_node\_id]

, [pdw\_node\_type]

, [pdw\_node\_name]

, [dist\_name]

, [dist\_position]

, [partition\_nmbr]

, [reserved\_space\_page\_count]

, [unused\_space\_page\_count]

, [data\_space\_page\_count]

, [index\_space\_page\_count]

, [row\_count]

, ([reserved\_space\_page\_count] \* 8.0) AS [reserved\_space\_KB]

, ([reserved\_space\_page\_count] \* 8.0)/1000 AS [reserved\_space\_MB]

, ([reserved\_space\_page\_count] \* 8.0)/1000000 AS [reserved\_space\_GB]

, ([reserved\_space\_page\_count] \* 8.0)/1000000000 AS [reserved\_space\_TB]

, ([unused\_space\_page\_count] \* 8.0) AS [unused\_space\_KB]

, ([unused\_space\_page\_count] \* 8.0)/1000 AS [unused\_space\_MB]

, ([unused\_space\_page\_count] \* 8.0)/1000000 AS [unused\_space\_GB]

, ([unused\_space\_page\_count] \* 8.0)/1000000000 AS [unused\_space\_TB]

, ([data\_space\_page\_count] \* 8.0) AS [data\_space\_KB]

, ([data\_space\_page\_count] \* 8.0)/1000 AS [data\_space\_MB]

, ([data\_space\_page\_count] \* 8.0)/1000000 AS [data\_space\_GB]

, ([data\_space\_page\_count] \* 8.0)/1000000000 AS [data\_space\_TB]

, ([index\_space\_page\_count] \* 8.0) AS [index\_space\_KB]

, ([index\_space\_page\_count] \* 8.0)/1000 AS [index\_space\_MB]

, ([index\_space\_page\_count] \* 8.0)/1000000 AS [index\_space\_GB]

, ([index\_space\_page\_count] \* 8.0)/1000000000 AS [index\_space\_TB]

FROM base

)

SELECT \*

FROM size

Take a moment to analyze the script above. Some of the tables might already look familiar. Here is a short description of the tables and DMVs involved in the query:

| **Table Name** | **Description** |
| --- | --- |
| sys.schemas | All schemas in the database. |
| sys.tables | All tables in the database. |
| sys.indexes | All indexes in the database. |
| sys.columns | All columns in the database. |
| sys.pdw\_table\_mappings | Maps each table to local tables on physical nodes and distributions. |
| sys.pdw\_nodes\_tables | Contains information on each local table in each distribution. |
| sys.pdw\_table\_distribution\_properties | Holds distribution information for tables (the type of distribution tables have). |
| sys.pdw\_column\_distribution\_properties | Holds distribution information for columns. Filtered to include only columns used to distribute their parent tables (distribution\_ordinal = 1). |
| sys.pdw\_distributions | Holds information about the distributions from the SQL pool. |
| sys.dm\_pdw\_nodes | Holds information about the nodes from the SQL pool. Filtered to include only compute nodes (type = COMPUTE). |
| sys.dm\_pdw\_nodes\_db\_partition\_stats | Returns page and row-count information for every partition in the current database. |

1. Run the following script to view the details about the structure of the tables in the wwi\_perf schema:

SELECT

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

, distribution\_column

, index\_type\_desc

, COUNT(distinct partition\_nmbr) as nbr\_partitions

, SUM(row\_count) as table\_row\_count

, SUM(reserved\_space\_GB) as table\_reserved\_space\_GB

, SUM(data\_space\_GB) as table\_data\_space\_GB

, SUM(index\_space\_GB) as table\_index\_space\_GB

, SUM(unused\_space\_GB) as table\_unused\_space\_GB

FROM

[wwi\_perf].[vTableSizes]

WHERE

schema\_name = 'wwi\_perf'

GROUP BY

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

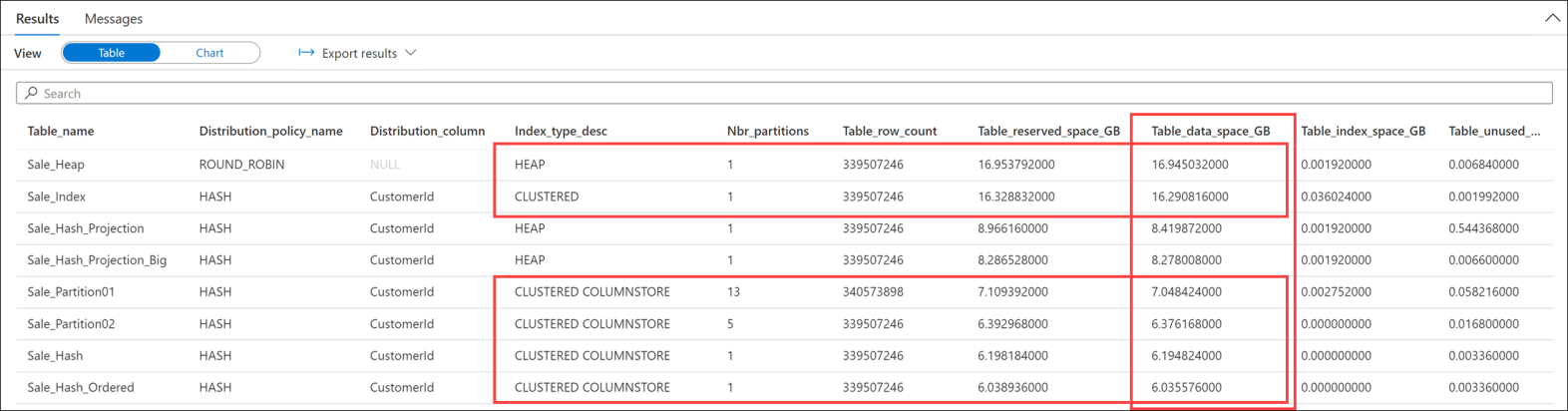
, distribution\_column

, index\_type\_desc

ORDER BY

table\_reserved\_space\_GB desc

Analyze the results:

[](https://docs.microsoft.com/en-us/learn/wwl-data-ai/analyze-optimize-data-warehouse-storage-azure-synapse-analytics/media/table-space.png#lightbox)

Notice the significant difference between the space used by CLUSTERED COLUMNSTORE and HEAP or CLUSTERED tables. This provides a clear indication on the significant advantages columnstore indexes have.

Also notice the slight increase of storage space for ordered clustered columnstore index (CCI) table (Sale\_Hash\_Ordered).