Check for skewed data and space usage

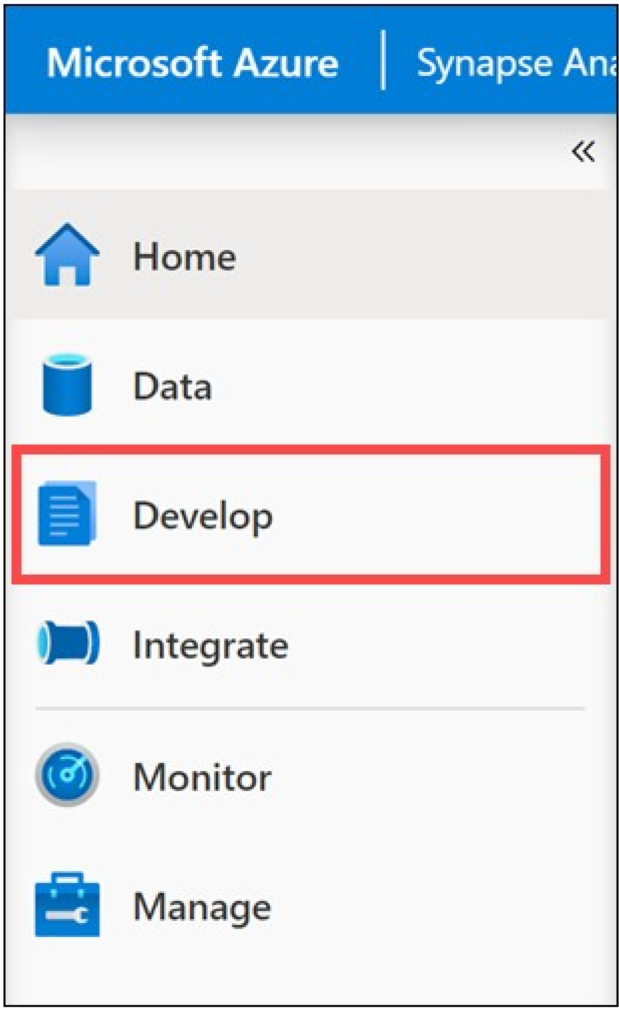
Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

Analyze the space used by tables

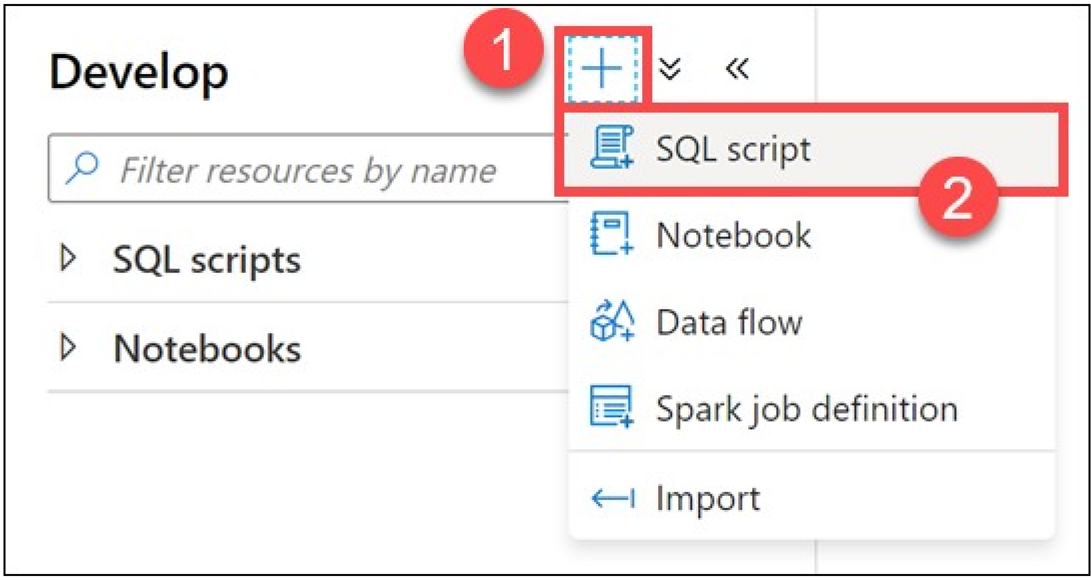
1. Open Synapse Studio.

2. Select the **Develop** hub.



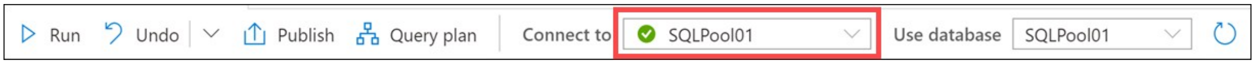
The develop hub is highlighted.

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



The SQL script context menu item is highlighted.

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

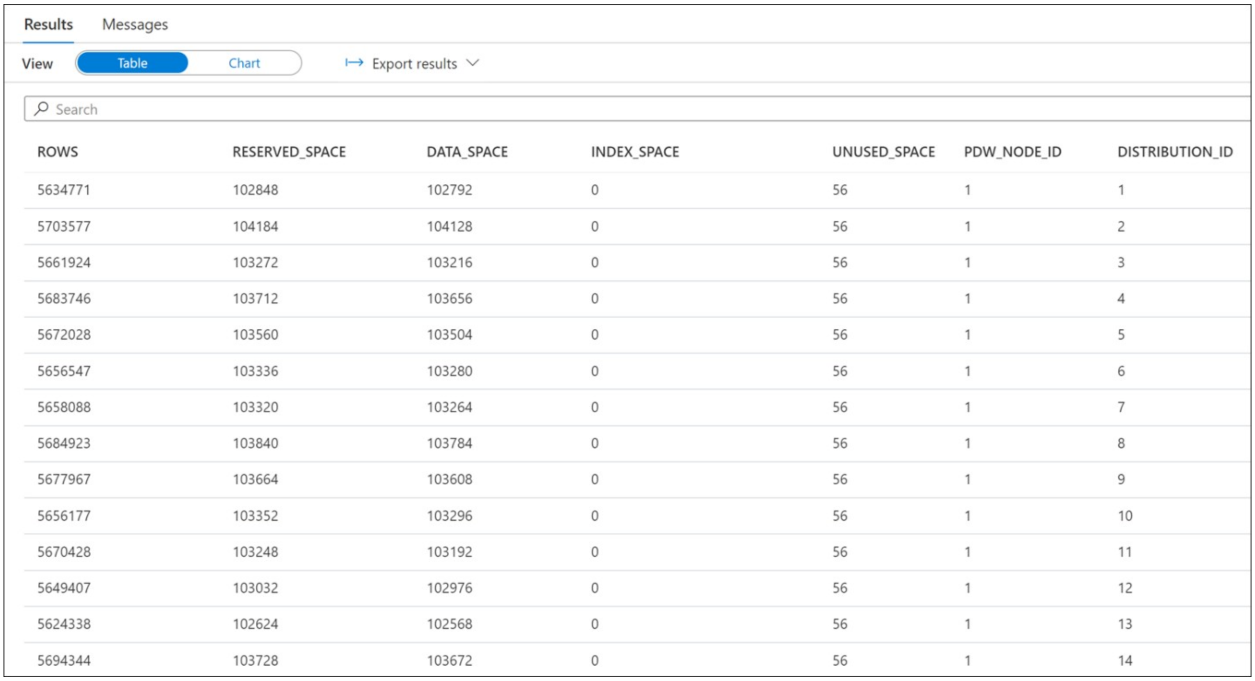


Connect to SQLPool01 Use database SQLPool01

The connect to option is highlighted in the query toolbar.

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

```
DBCC PDW_SHOWSPACEUSED('wwi_perf.Sale_Hash');
```



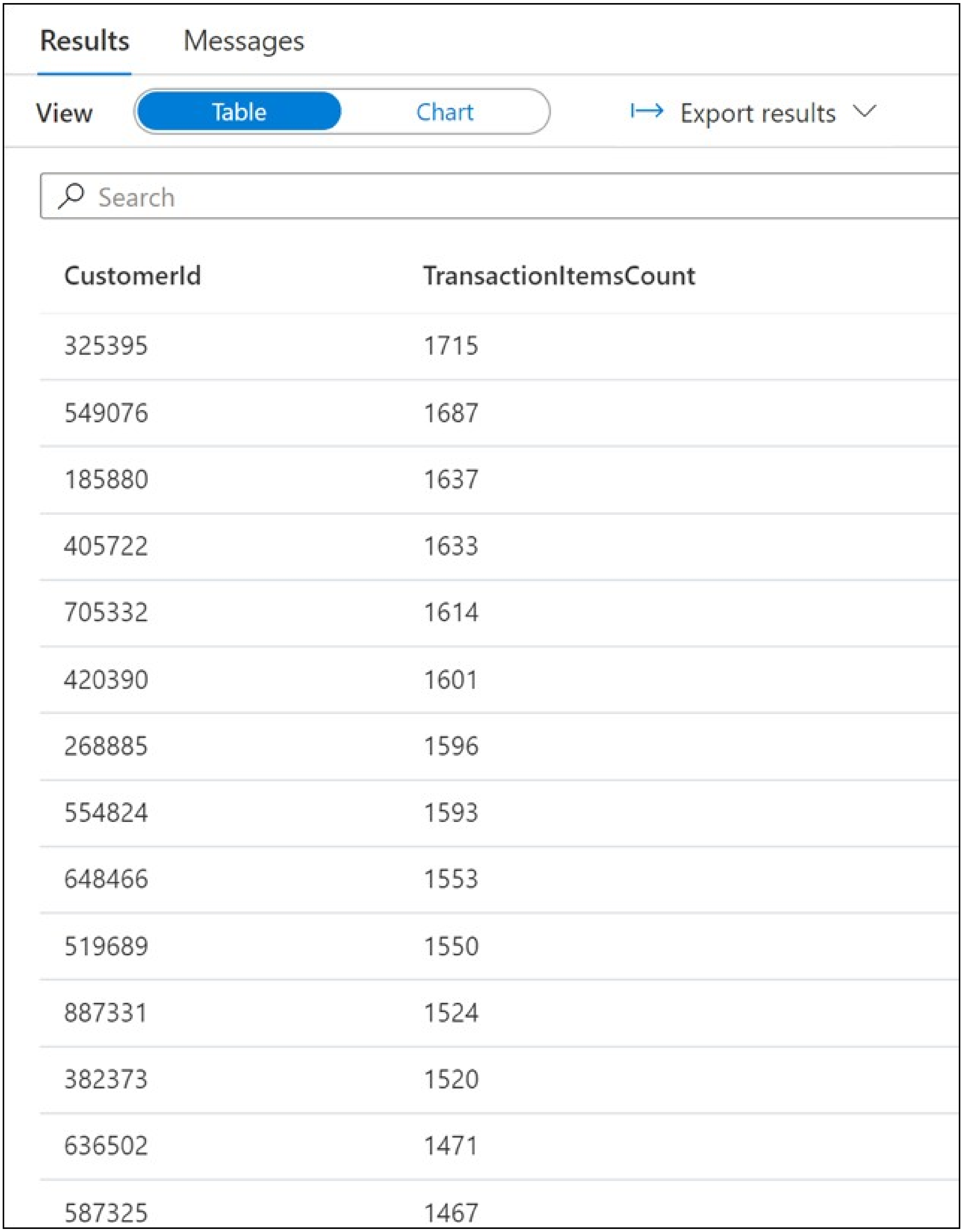
ROWS	RESERVED_SPACE	DATA_SPACE	INDEX_SPACE	UNUSED_SPACE	PDW_NODE_ID	DISTRIBUTION_ID
5634771	102848	102792	0	56	1	1
5703577	104184	104128	0	56	1	2
5661924	103272	103216	0	56	1	3
5683746	103712	103656	0	56	1	4
5672028	103560	103504	0	56	1	5
5656547	103336	103280	0	56	1	6
5658088	103320	103264	0	56	1	7
5684923	103840	103784	0	56	1	8
5677967	103664	103608	0	56	1	9
5656177	103352	103296	0	56	1	10
5670428	103248	103192	0	56	1	11
5649407	103032	102976	0	56	1	12
5624338	102624	102568	0	56	1	13
5694344	103728	103672	0	56	1	14

Show table space usage

6. 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:

```sql
SELECT TOP 1000
    CustomerId,
    count(*) as TransactionItemsCount
FROM
    [wwi_perf].[Sale_Hash]
GROUP BY
    CustomerId
ORDER BY
    count(*) DESC
```

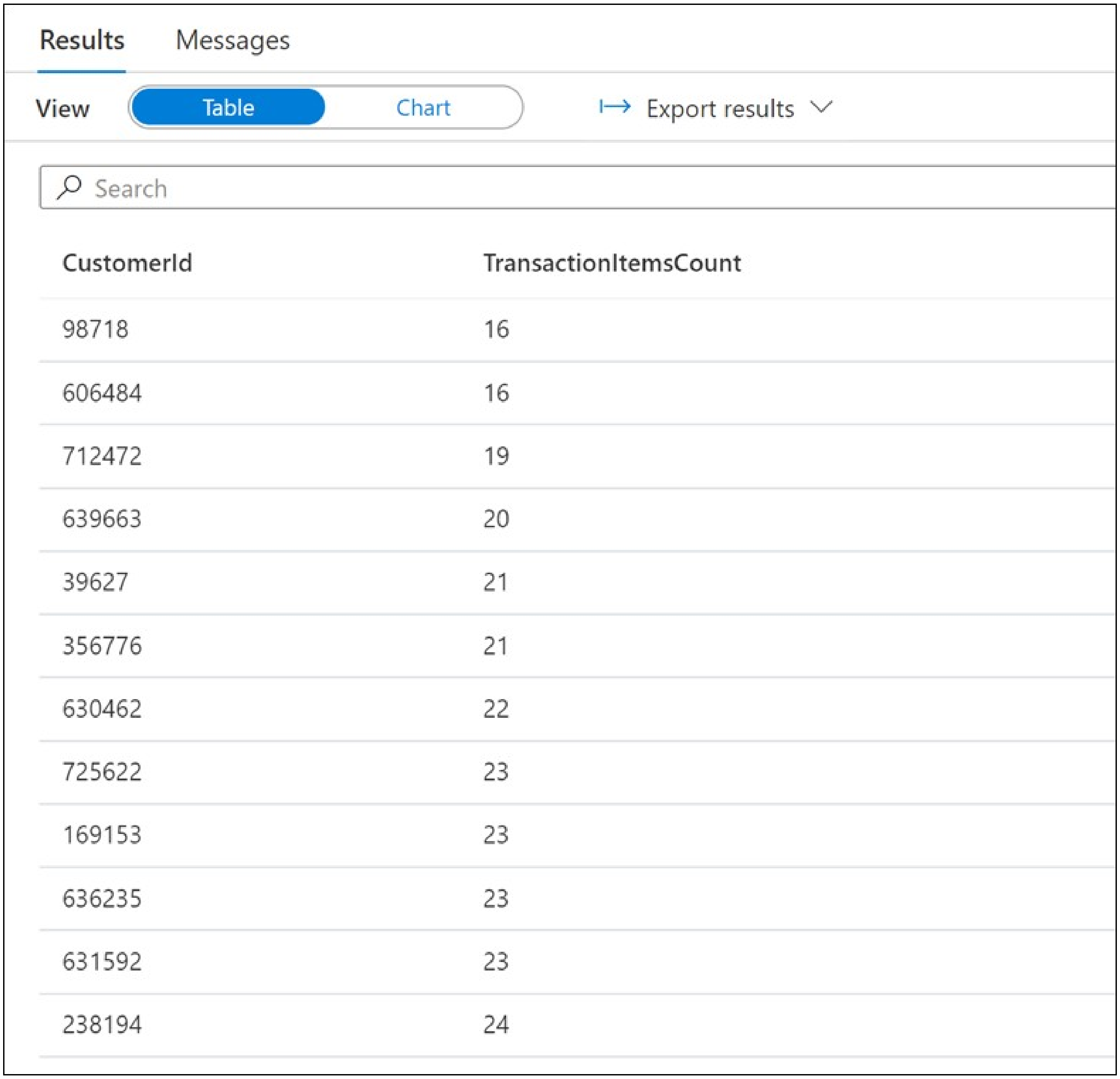


CustomerId	TransactionItemsCount
325395	1715
549076	1687
185880	1637
405722	1633
705332	1614
420390	1601
268885	1596
554824	1593
648466	1553
519689	1550
887331	1524
382373	1520
636502	1471
587325	1467

Initial look at the customers with most sale transaction items

Now find the customers with the least sale transaction items:

```
SELECT TOP 1000
    CustomerId,
    count(*) as TransactionItemsCount
FROM
    [wwi_perf].[Sale_Hash]
GROUP BY
    CustomerId
ORDER BY
    count(*) ASC
```



CustomerId	TransactionItemsCount
98718	16
606484	16
712472	19
639663	20
39627	21
356776	21
630462	22
725622	23
169153	23
636235	23
631592	23
238194	24

Customers with most sale transaction items

Notice the largest number of transaction items is 69 and the smallest is 16. Let's find now the distribution of per-customer transaction item counts. Run the following query:

```
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):

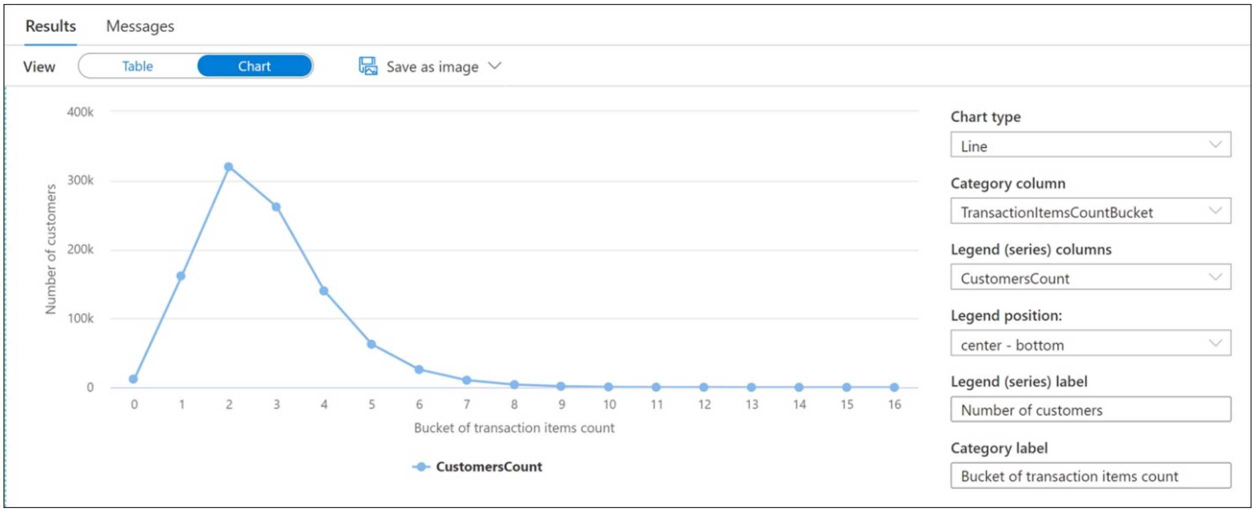


Chart type: Line

Category column: TransactionItemsCountBucket

Legend (series) columns: CustomersCount

Legend position: center - bottom

Legend (series) label: Number of customers

Category label: Bucket of transaction items count

Distribution of per-customer transaction item counts

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]
```

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:

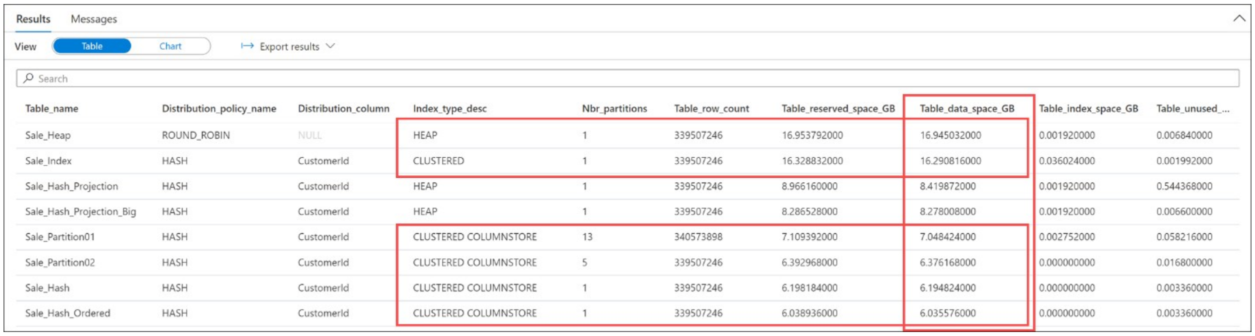
Table Name	Description
sys.schemas	All schemas in the database.
sys.tables	All tables in the database.
sys.indexes	All indexes in the database.
sys.columns	All columns in the database.
sys.pdw_table_mappings	Maps each table to local tables on physical nodes and distributions.
sys.pdw_nodes_tables	Contains 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.

2. 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:



Table_name	Distribution_policy_name	Distribution_column	Index_type_desc	Nbr_partitions	Table_row_count	Table_reserved_space_GB	Table_data_space_GB	Table_index_space_GB	Table_unused_...
Sale_Heap	ROUND_ROBIN	NULL	HEAP	1	339507246	16.953792000	16.945032000	0.001920000	0.006840000
Sale_Index	HASH	CustomerId	CLUSTERED	1	339507246	16.328832000	16.290816000	0.036024000	0.001992000
Sale_Hash_Projection	HASH	CustomerId	HEAP	1	339507246	8.966160000	8.419872000	0.001920000	0.544368000
Sale_Hash_Projection_Big	HASH	CustomerId	HEAP	1	339507246	8.286528000	8.278008000	0.001920000	0.006600000
Sale_Partition01	HASH	CustomerId	CLUSTERED COLUMNSTORE	13	340573898	7.109392000	7.048424000	0.002752000	0.058216000
Sale_Partition02	HASH	CustomerId	CLUSTERED COLUMNSTORE	5	339507246	6.392968000	6.376168000	0.000000000	0.016800000
Sale_Hash	HASH	CustomerId	CLUSTERED COLUMNSTORE	1	339507246	6.198184000	6.194824000	0.000000000	0.003360000
Sale_Hash_Ordered	HASH	CustomerId	CLUSTERED COLUMNSTORE	1	339507246	6.038936000	6.035576000	0.000000000	0.003360000

Detailed table space usage

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`**).