## **Designing tables in Azure SQL Data Warehouse**

## **Determine table category:**

A star schema organizes data into fact and dimension tables. Some tables are used for integration or staging data before it moves to a fact or dimension table. As you design a table, decide whether the table data belongs in a fact, dimension, or integration table. This decision informs the appropriate table structure and distribution.

* **Fact tables** contain quantitative data that are commonly generated in a transactional system, and then loaded into the data warehouse. For example, a retail business generates sales transactions every day, and then loads the data into a data warehouse fact table for analysis.
* **Dimension tables** contain attribute data that might change but usually changes infrequently. For example, a customer's name and address are stored in a dimension table and updated only when the customer's profile changes. To minimize the size of a large fact table, the customer's name and address do not need to be in every row of a fact table. Instead, the fact table and the dimension table can share a customer ID. A query can join the two tables to associate a customer's profile and transactions.
* **Integration tables** provide a place for integrating or staging data. You can create an integration table as a regular table, an external table, or a temporary table. For example, you can load data to a staging table, perform transformations on the data in staging, and then insert the data into a production table.

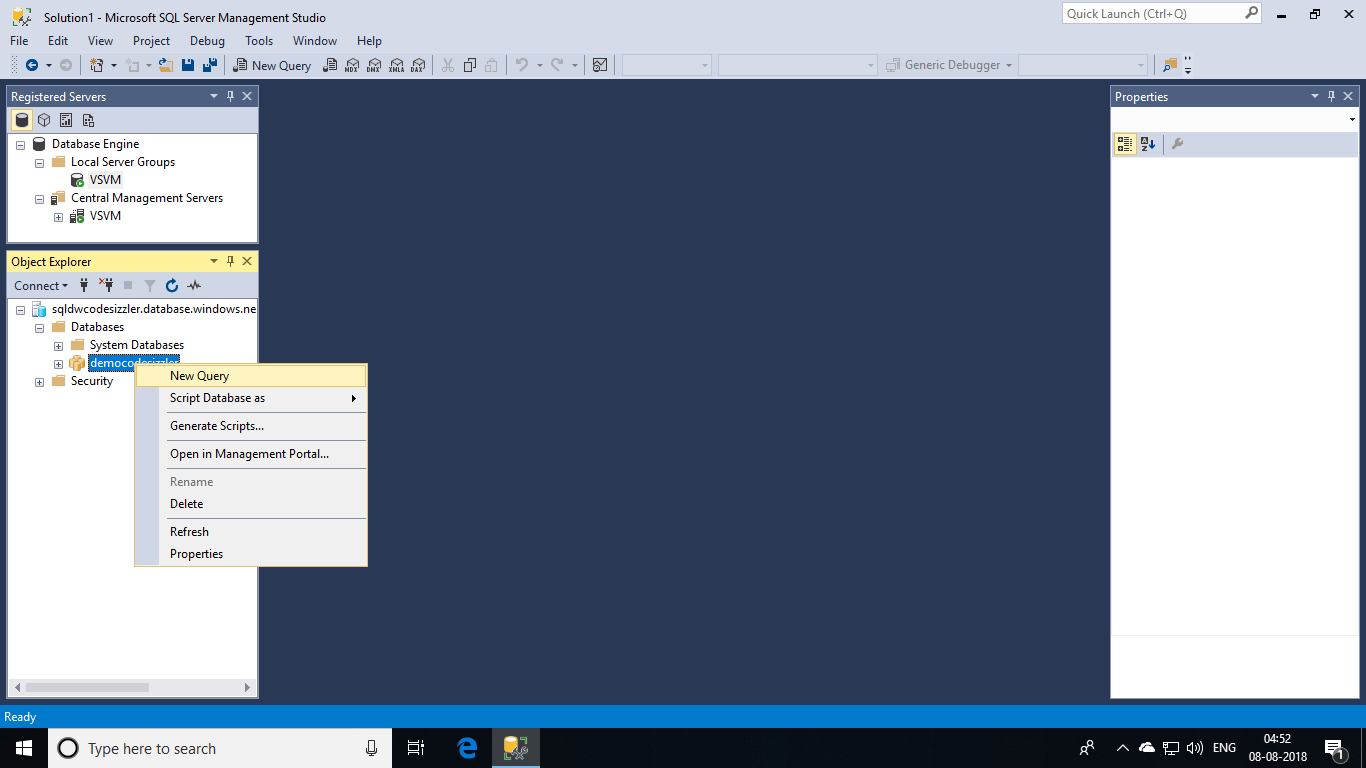
## **Schema and table names:**

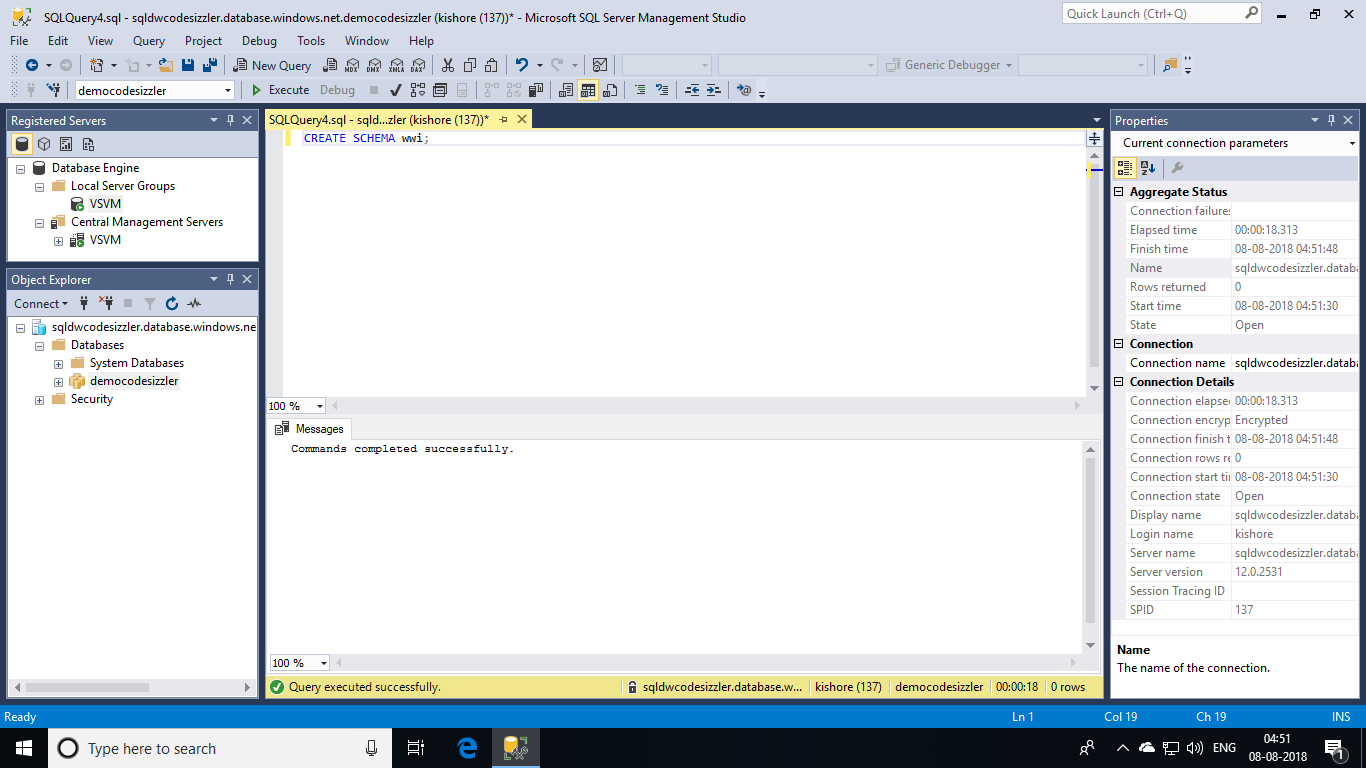
In SQL Data Warehouse, a data warehouse is a type of database. All of the tables in the data warehouse are contained within the same database. You cannot join tables across multiple data warehouses. This behaviour is different from SQL Server, which supports cross-database joins.

In a SQL Server database, you might use fact, dim, or integrate for the schema names. If you are migrating a SQL Server database to SQL Data Warehouse, it works best to migrate all of the fact, dimension, and integration tables to one schema in SQL Data Warehouse. For example, you could store all the tables in the [WideWorldImportersDW](https://docs.microsoft.com/en-us/sql/sample/world-wide-importers/database-catalog-wwi-olap) sample data warehouse within one schema called wwi. The following code creates a [user-defined schema](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-schema-transact-sql) called wwi.

Now, open the SSM and login into your SQL Warehouse. Then Right click the Datawarehouse which you would have create and run the query given below by click on **Execute** button. This will be creating a Schema now.

CREATE SCHEMA wwi;

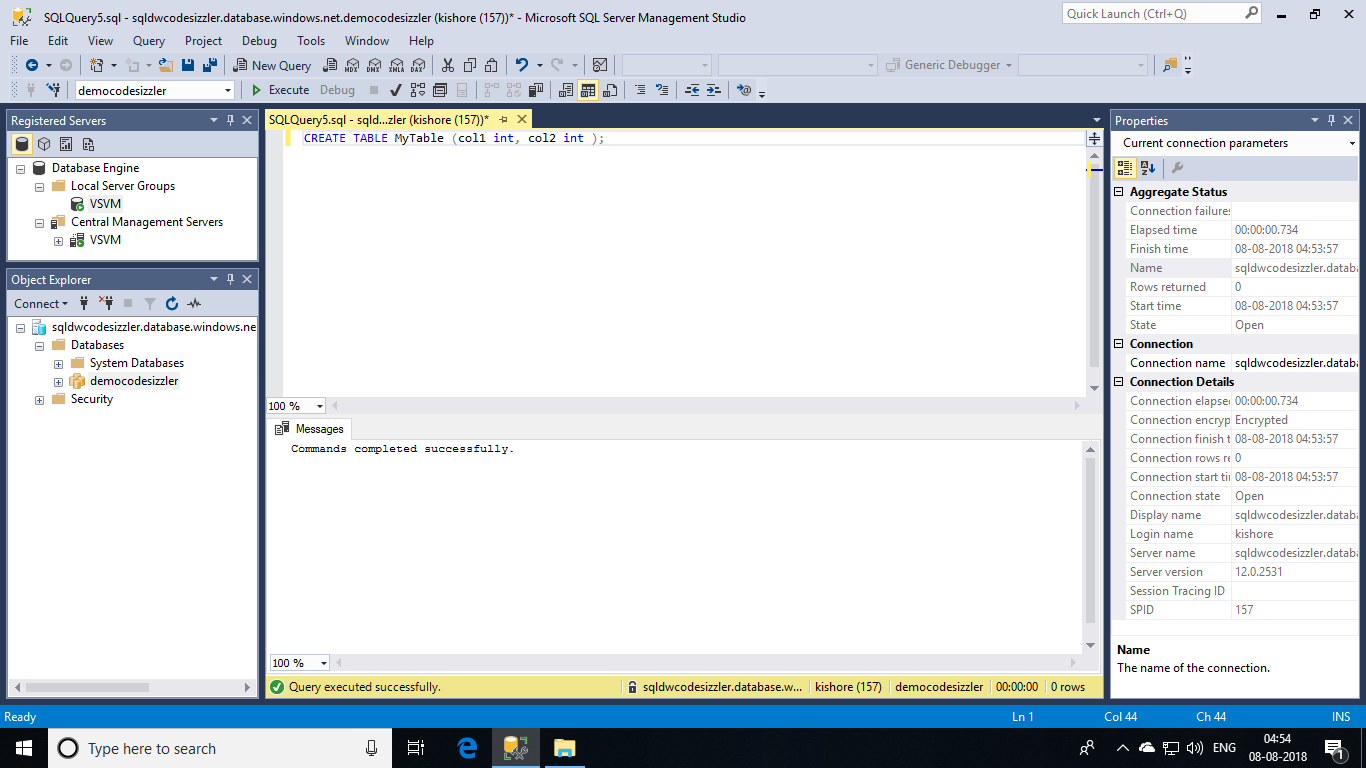




### **Regular table:**

A regular table stores data in Azure Storage as part of the data warehouse. The table and the data persist regardless of whether a session is open. This example creates a regular table with two columns.

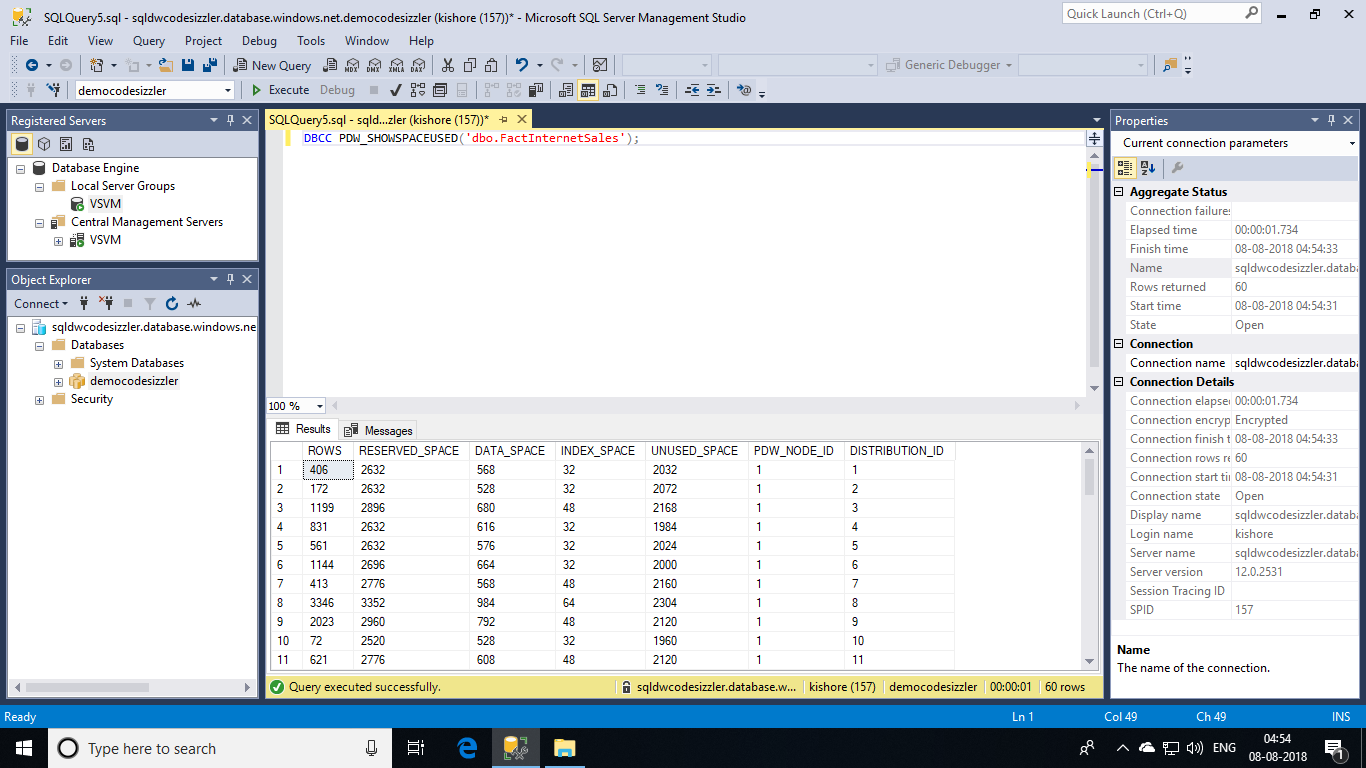
CREATE TABLE MyTable (col1 int, col2 int );



## **Table size queries:**

One simple way to identify space and rows consumed by a table in each of the 60 distributions, is to use DBCC PDW\_SHOWSPACEUSED.

DBCC PDW\_SHOWSPACEUSED('dbo.FactInternetSales');



However, using DBCC commands can be quite limiting. Dynamic management views (DMVs) show more detail than DBCC commands. Start by creating a view by executing the below given query.

CREATE VIEW dbo.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]

)

, 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]

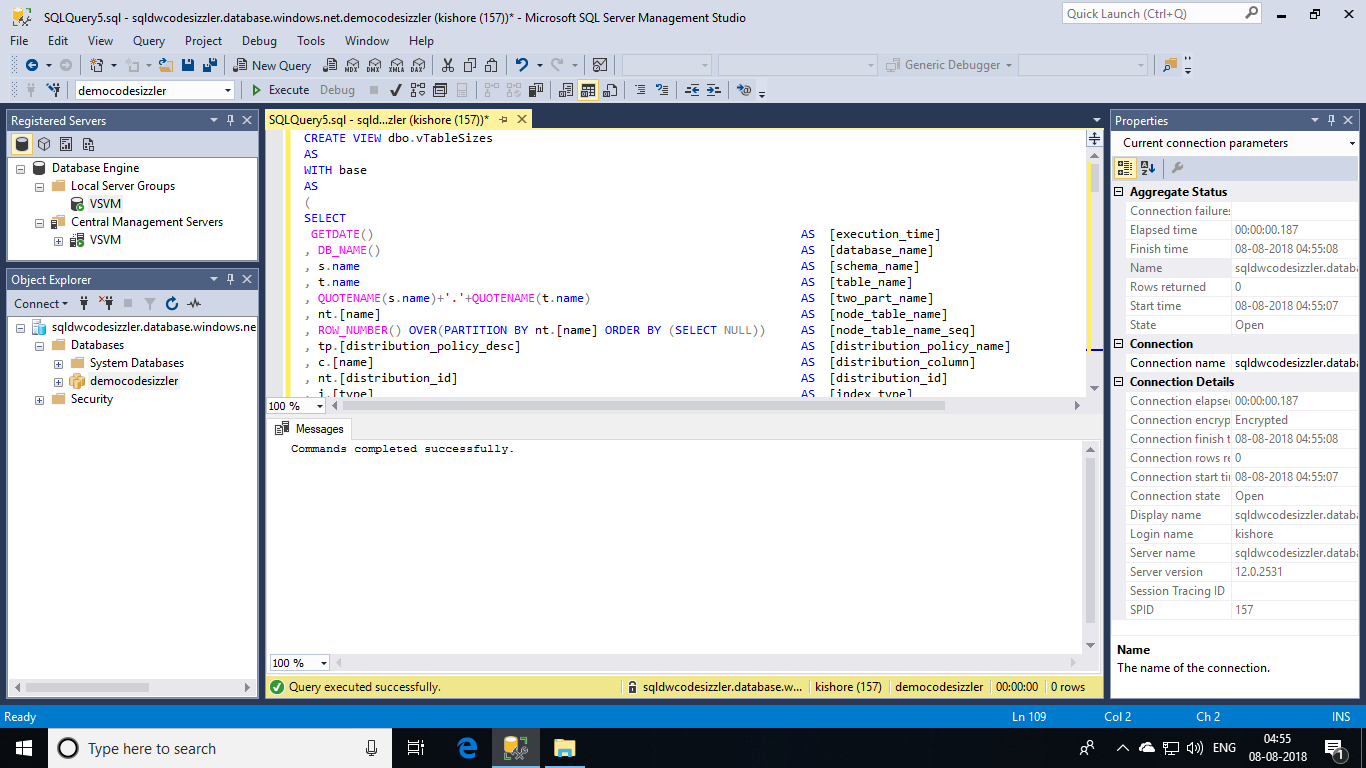
FROM base

)

SELECT \*

FROM size

;



The above execution will be helping you in performing Joins and many other operations over your warehouse table and represent them in a view that is defined in the script. Let us explore few more queries regarding to **Table Size**.

### **Table space summary:**

This query returns the rows and space by table. It allows you to see which tables are your largest tables and whether they are round-robin, replicated, or hash -distributed. For hash-distributed tables, the query shows the distribution column.

Now, execute the script that is shown below.

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

dbo.vTableSizes

GROUP BY

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

, distribution\_column

, index\_type\_desc

ORDER BY

table\_reserved\_space\_GB desc

;

### 

### **Table space by distribution type:**

SELECT

distribution\_policy\_name

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

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

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

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

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

FROM dbo.vTableSizes

GROUP BY distribution\_policy\_name

;

### 

### **Table space by index type:**

SELECT

index\_type\_desc

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

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

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

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

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

FROM dbo.vTableSizes

GROUP BY index\_type\_desc

;

### 

### **Distribution space summary:**

SELECT

distribution\_id

, SUM(row\_count) as total\_node\_distribution\_row\_count

, SUM(reserved\_space\_MB) as total\_node\_distribution\_reserved\_space\_MB

, SUM(data\_space\_MB) as total\_node\_distribution\_data\_space\_MB

, SUM(index\_space\_MB) as total\_node\_distribution\_index\_space\_MB

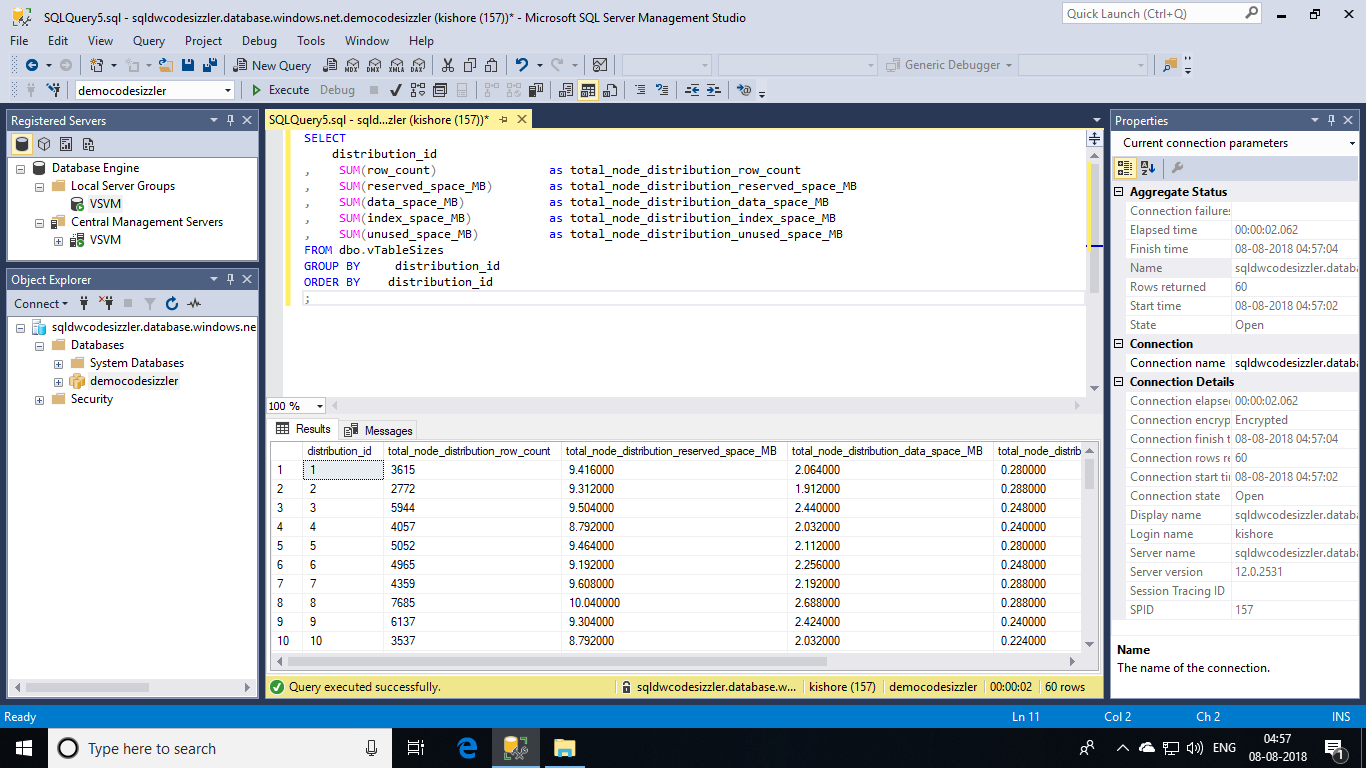
, SUM(unused\_space\_MB) as total\_node\_distribution\_unused\_space\_MB

FROM dbo.vTableSizes

GROUP BY distribution\_id

ORDER BY distribution\_id

;



These are a few set of queries that can be used to work with SQL Datawarehouse tables and perform the ETL tasks with the same.