Create statistics to improve query performance

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.

When queries are submitted, a dedicated SQL pool query optimizer tries to determine which access paths to the data will result in the least amount of effort to retrieve the data required to resolve the query. It is a cost-based optimizer, and compares the cost of various query plans, and then chooses the plan with the lowest cost.

Statistics in dedicated SQL pools

To aid this process, statistics are required that describe the amount of data that is present within ranges of values, and range of rows that may be returned to fulfill a query filter or join. Therefore, after loading data into a dedicated SQL pool, collecting statistics on your data is one of the most important things you can do for query optimization.

When you create a database in a dedicated SQL pool in Azure Synapse Analytics, the automatic creation of statistics is turned on by default. This means that statistics are created when you run the following type of Transact-SQL statements:

- SELECT
- INSERT-SELECT
- CTAS
- UPDATE
- DELETE
- EXPLAIN when containing a join or the presence of a predicate is detected

When executing the above Transact-SQL statements, that the statistics creation is performed on the fly, and as a result, there can be a slight degradation in query performance.

To avoid this, statistics are also created on any index that you create that helps aid the query optimize process. As this is an action that is performed in advance of querying the table on which the index is based, it means that the statistics are created in advance. However, you must consider that as new data is loaded into the table, the statistics may become out of date.

As such, it is important to update the statistics after you load data or update large ranges of data, so that queries can benefit from the updated statistics information.

You can check if your data warehouse has AUTO_CREATE_STATISTICS configured by running the following command:

```
SELECT name, is_auto_create_stats_on
FROM sys.databases
```

If your data warehouse doesn't have AUTO_CREATE_STATISTICS enabled, it is recommended that you enable this property by running the following command:

```
ALTER DATABASE <yourdatawarehousename>
SET AUTO_CREATE_STATISTICS ON
```

Statistics in serverless SQL pools

Statistics in a serverless SQL pool has the same objective of using a cost-based optimizer to choose an execution plan that will execute the fastest. How it creates its statistics is different.

Serverless SQL pool analyses incoming user queries for missing statistics. If statistics are missing, the query optimizer creates statistics on individual columns in the query predicate or join condition to improve cardinality estimates for the query plan. The SELECT statement will trigger automatic creation of statistics. You can also manually create statistics, this is important when working with CSV files, as automatic statistics creation is not enabled for them.

In the following example, a system stored procedure is used to specify the creation of statistics for a specific Transact-SQL statement

```
sys.sp_create_openrowset_statistics [ @stmt = ] N'statement_text'
```

To create statistics for a specific column within a csv file, you can run the following code:

```
/* make sure you have the credentials to access the storage account created
IF EXISTS (SELECT * FROM sys.credentials WHERE name = 'https://
azureopendatastorage.blob.core.windows.net/censusdatacontainer')
DROP CREDENTIAL [https://azureopendatastorage.blob.core.windows.net/
censusdatacontainer]
GO

CREATE CREDENTIAL [https://azureopendatastorage.blob.core.windows.net/
censusdatacontainer]
WITH IDENTITY='SHARED ACCESS SIGNATURE',
SECRET = ''
GO
*/

/*
The following code will create statistics on a column named year, from a file na
med population.csv
*/

EXEC sys.sp_create_openrowset_statistics N'SELECT year
FROM OPENROWSET(
        BULK ''https://sqlondemandstorage.blob.core.windows.net/csv/population/
population.csv'',
        FORMAT = ''CSV'',
        FIELDTERMINATOR ='','',
        ROWTERMINATOR = ''\n''
    )
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
```

```
    [year] smallint,
    [population] bigint
) AS [r]
'
```

You should also update the statistics when the data in the files change. In fact, Serverless SQL pool automatically recreates statistics if data is changed significantly. Every time statistics are automatically created, the current state of the dataset is also saved: file paths, sizes, last modification dates.

To update statistics for the year column in the dataset, which is based on the population.csv file, you need to drop and then create them, here is the drop statement:

```
EXEC sys.sp_drop_openrowset_statistics N'SELECT year
FROM OPENROWSET(
        BULK ''https://sqlondemandstorage.blob.core.windows.net/csv/population/population.csv'',
        FORMAT = ''CSV'',
        FIELDTERMINATOR ='','',
        ROWTERMINATOR = ''\n''
    )
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS [r]
'
```

To update statistics for a statement, you need to drop and create statistics. The following stored procedure is used to drop statistics against a specific Transact-SQL text:

```
1
sys.sp_drop_openrowset_statistics [ @stmt = ] N'statement_text'
```