Write data back to the Azure Cosmos DB transactional store

Note In this reading you can see the steps involved in the process of writing data back to the Azure Cosmos DB transactional store.

Whist the analytical insights that Adventure Works was able to gather (from joining together operational data that was previously siloed in disconnected and hard to correlate data stores) was useful, they have decided that making this information to every user across the organization though their existing application is the right way to achieve ongoing visibility of these key statistics. To do that we need to write this data back to the Azure Cosmos DB transactional stores.

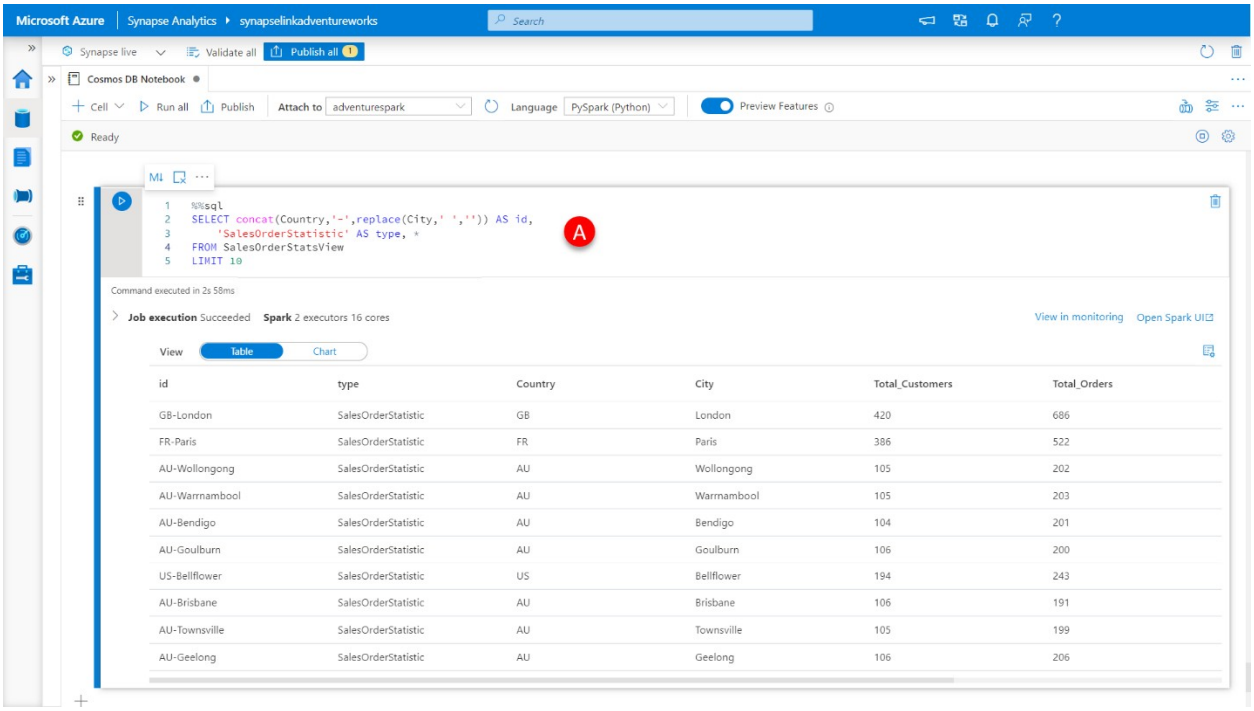
As with all applications the process starts by appropriately modeling the data. Some of the things to consider as we write this data back to Azure Cosmos DB:

- Does the document have an appropriate partition key, such that it will evenly distribute items and queries across all partitions as the data grows?
- Can we provide an appropriate and usable primary key (ID or _id, in the case of Core (SQL) API and API for MongoDB respectively) such that when used together with the partition key can uniquely identify an item for point operations (CRUD)?
- Where we have modeled containers to support multiple entity types that we identify the new data with the appropriate entity type?

For our example, where we are wanting to write the limited number of statistic rows back to the transactional store, we choose to create a new compound ID value that will allow client applications to look up the latest statistics for their city easily by concatenating the country code and city name together to uniquely identify each document. Given that we will be storing this in a container with other entity types, we uniquely identify its type as “SalesOrderStatistic” data.

1. Paste the code below into a **new cell (A)**, and click the **run cell** button.

```
%%sql
SELECT concat(Country,'-',replace(City,' ','')) AS id,
    'SalesOrderStatistic' AS type, *
FROM SalesOrderStatsView
LIMIT 10
```

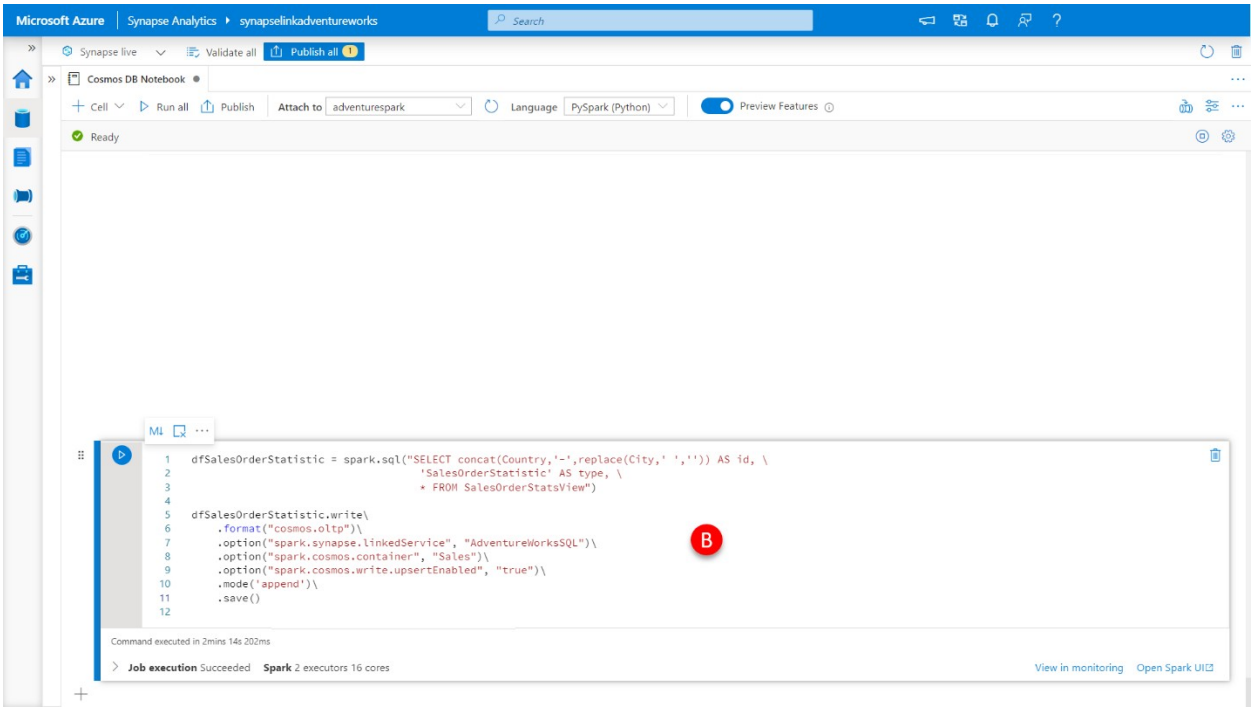


querying the sales order stats view
As you can see, we now have an appropriately shaped result set, ready to write back to the transactional store.

2. Paste the code below into a **new cell (B)**, and click the **run cell** button.

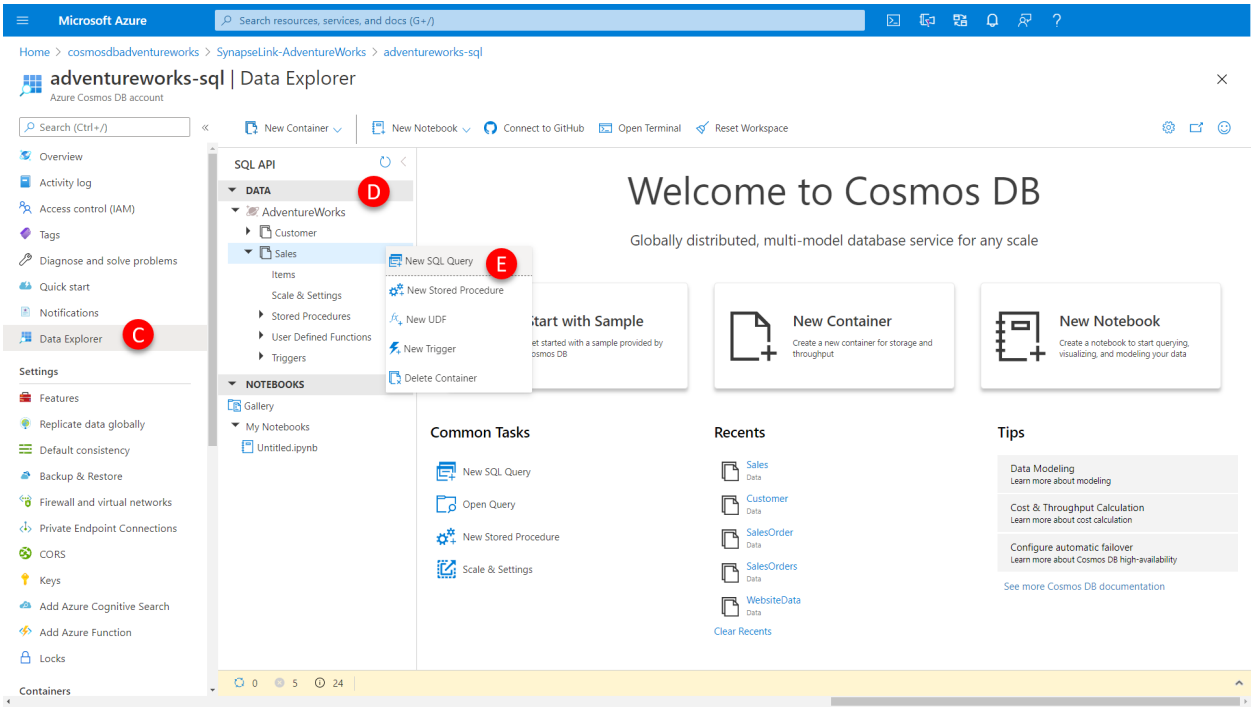
```
dfSalesOrderStatistic = spark.sql("SELECT concat(Country,'-',replace(City,' ','')
) AS id, \
                                  'SalesOrderStatistic' AS type, \
                                  * FROM SalesOrderStatsView")

dfSalesOrderStatistic.write\
    .format("cosmos.oltp")\
    .option("spark.synapse.linkedService", "AdventureWorksSQL")\
    .option("spark.cosmos.container", "Sales")\
    .option("spark.cosmos.write.upsertEnabled", "true")\
```

```
    .mode('append')\
    .save()
```



Write back data to the transactional store

Within a couple of minutes the DataFrame write operation should have finished writing our statistics back to the transactional store.



Creating a query to validate the data

Let's validate that these records are now visible in our Cosmos DB container:

3. Navigate to the Azure portal (https://portal.azure.com) and select the **Azure Cosmos DB account**.

4. Navigate to your previously created **Azure Cosmos DB Core (SQL) API account**.

5. Select **Data Explorer** in the **left-hand menu (C)**.

6. Navigate to the **Sales container** we created earlier by **(D)**:

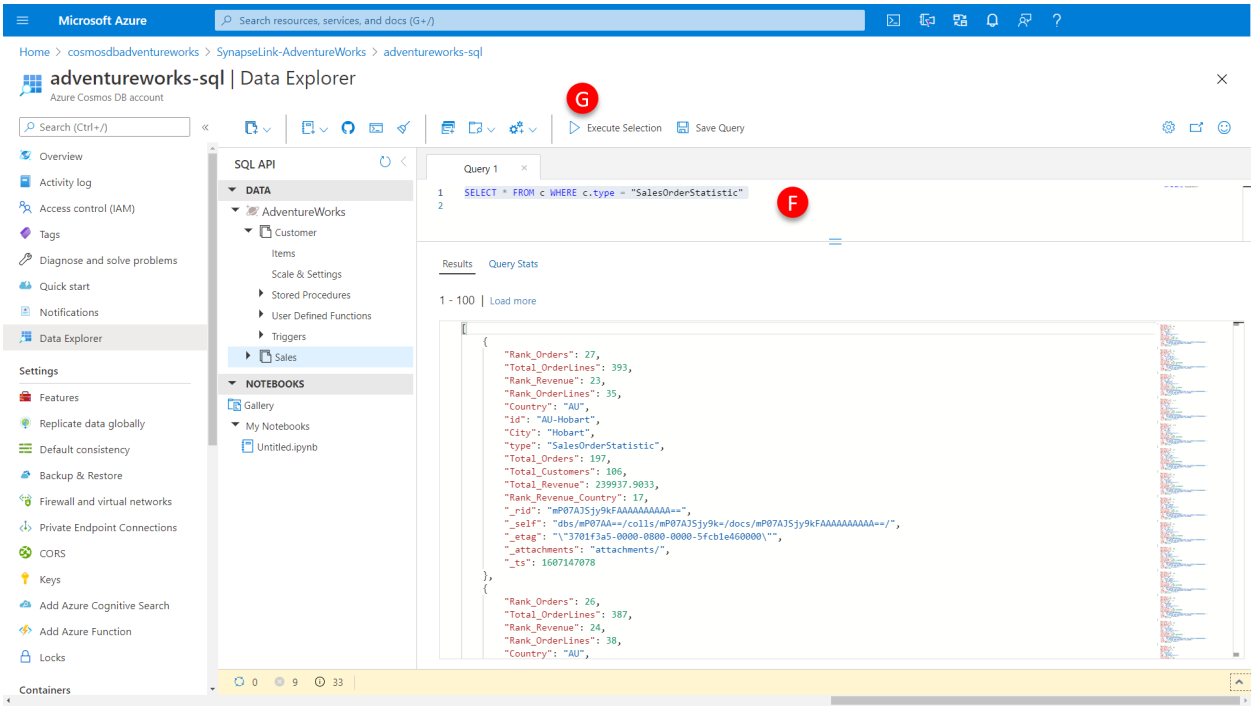
- Expanding AdventureWorks database
- Expanding the Sales Container

7. Clicking on the **...** and selecting **New SQL Query** from the menu **(E)**

8. Paste the below code into the query pane **(F)**, and click the **Execute** button **(G)**.

```
SELECT * FROM c WHERE c.type = "SalesOrderStatistic"
```

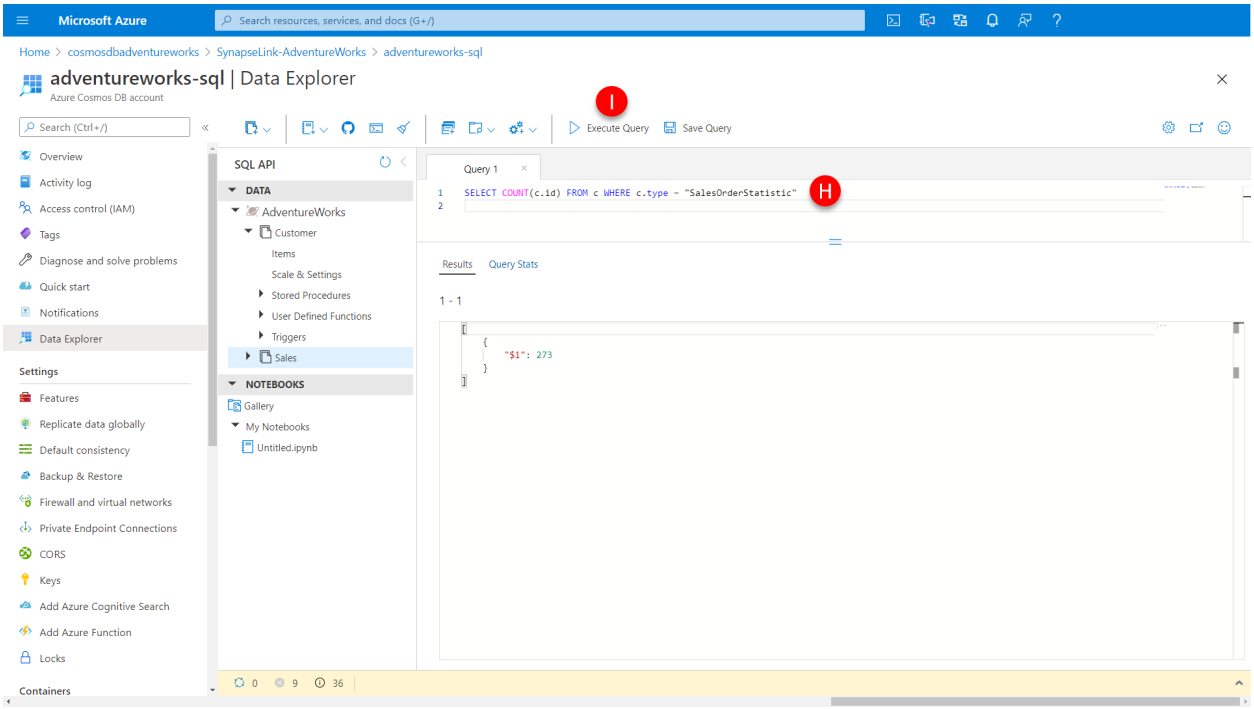
This will immediately return items containing the statistics we inserted from Synapse Analytics.



Viewing query results
We can validate that all the expected 273 items have been inserted:

9. Paste the code below into the query pane **(F)**, and click the **Execute** button **(G)**.

```
SELECT COUNT(c.id) FROM c WHERE c.type = 'SalesOrderStatistic’
```



Writing to the transactional store
This should immediately return a value of 273 as the result.

We have successfully updated the Azure Cosmos DB transactional store from Synapse Analytics, with the results of our analytical processing performed over data that was sourced from Azure Cosmos DB analytical store.