Here is the code which is shown in Figure 18-12 to read the database and container name that you created in the previous section.

import azure.cosmos

from azure.cosmos.partition\_key import PartitionKey

database = cosmos\_client.get\_database\_client('RetailDemo')

print('Database RetailDemo Read')

container = database.get\_container\_client('WebsiteData')

print('Container WebsiteData Read')

Here is the code which is shown in Figure 18-13 to update the throughput of the container to allow for a quicker upload.

old\_throughput = container.read\_offer().offer\_throughput

new\_throughput = container.replace\_throughput(1000).offer\_throughput

print("Container WebsiteData's throughput updated from {} RU/s to {} RU/s".format(old\_throughput, new\_throughput))

Here is the code which is shown in Figure 18-14 that uses the %%upload magic function to insert items into the container.

%%upload --databaseName RetailDemo --containerName WebsiteData --url <https://cosmosnotebooksdata.blob.core.windows.net/notebookdata/websiteData.json>

Here is the code which is shown in Figure 18-15 to scale down the throughput of the container once the data load is complete.

lowered\_throughput = container.replace\_throughput(400).offer\_throughput

print("Container WebsiteData's throughput lowered from {} RU/s to {} RU/s".format(new\_throughput, lowered\_throughput))

After the linked service has been created, run the following code which will be auto created when the data is loaded to a data frame from the previous step. Prior to running the code, remember to create a Synapse Spark pool and attach it to the notebook.

# Read from Cosmos DB analytical store into a Spark DataFrame and display 10 rows from the DataFrame

# To select a preferred list of regions in a multi-region Cosmos DB account, add .option("spark.cosmos.preferredRegions", "<Region1>,<Region2>")

df = spark.read\

    .format("cosmos.olap")\

    .option("spark.synapse.linkedService", "LS\_CosmosDb\_RetailDemo")\

    .option("spark.cosmos.container", "WebsiteData")\

    .load()

display(df.limit(10))

Use the following code to aggregate the dataset to get a sum of the Price column and then display the data frame.

from pyspark.sql.functions import \*

df = df.agg(sum(col('Price')))

df.show()