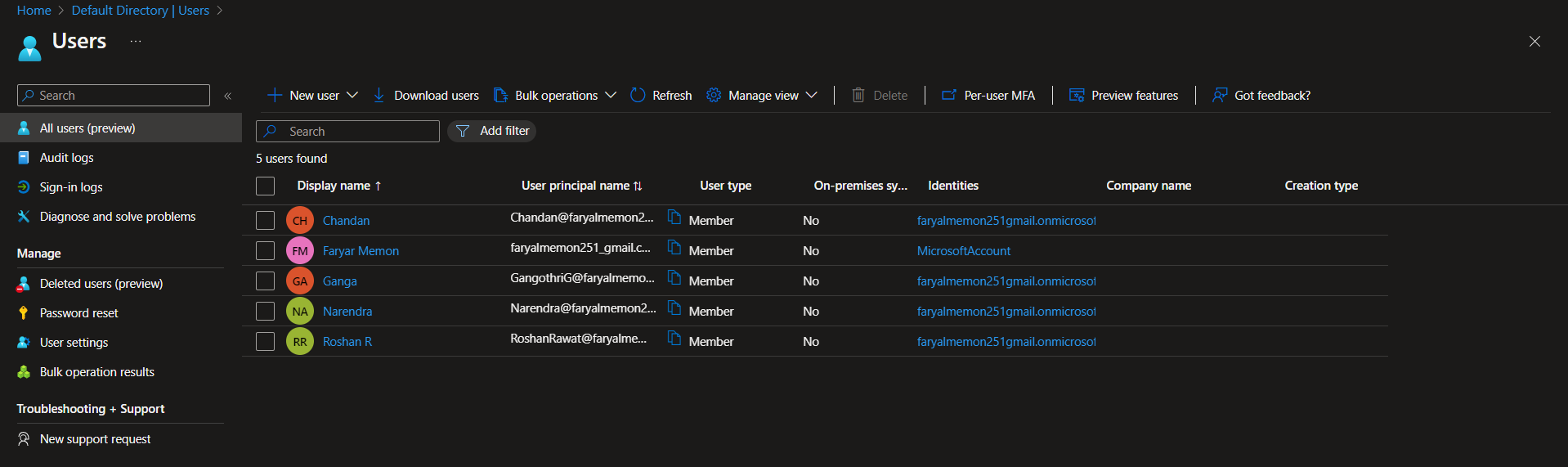
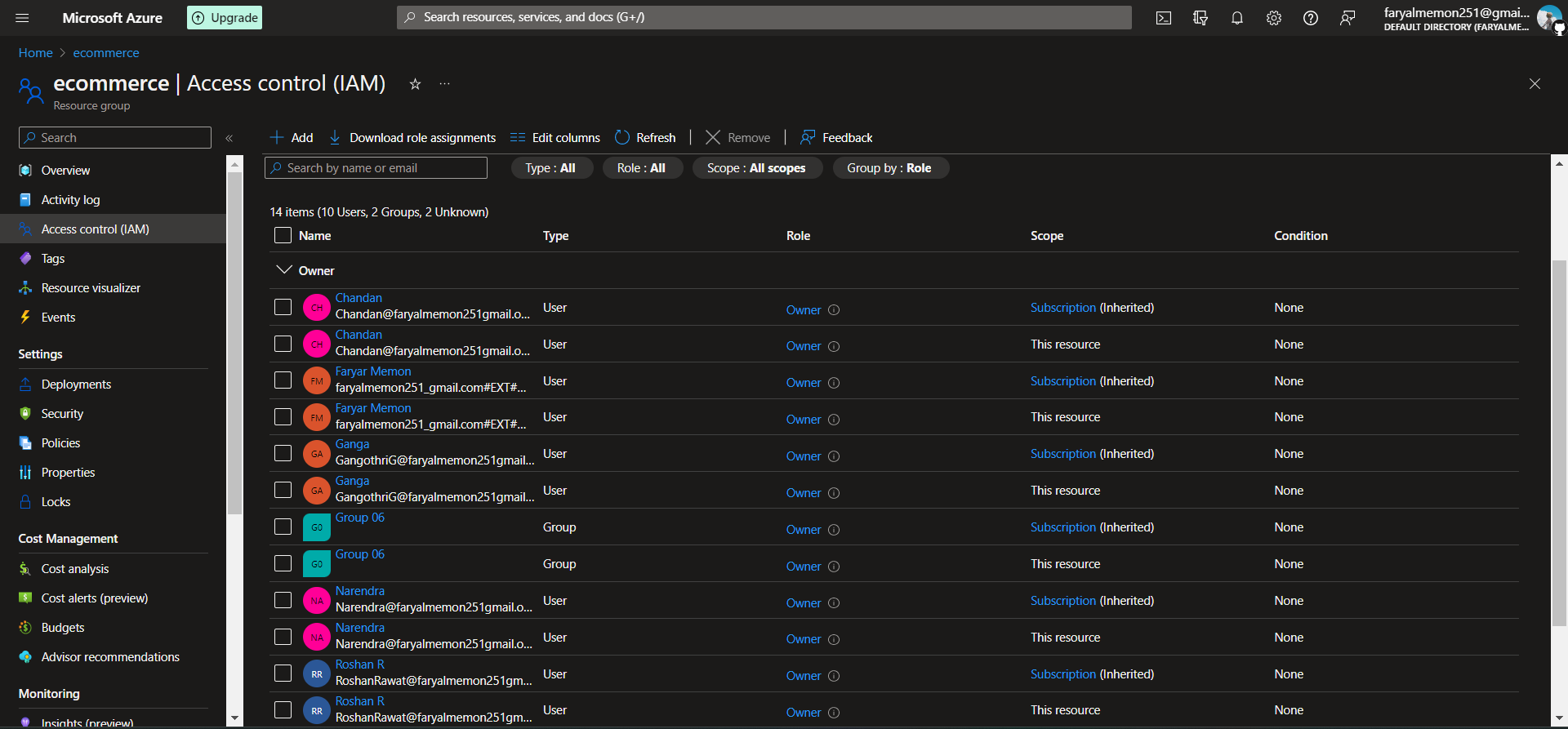
**RETAIL SALES ANALYSIS**

**Group 06:** Roshan Rawat, Gangothri Gadige, Chandan Kumar, Narendra Reddy, Faryar Memon

### Create multiple users to work under the same resource group



**Give privileges to the users in Resource Group and Subscriptions**



### Create Storage Account and Database

## Client Source Azure SQL Database config

create table retail(region varchar(1000), country varchar(1000), item\_type varchar(1000), sales\_channel varchar(1000), order\_priority varchar(1000), order\_date varchar(2000), order\_id int, ship\_date varchar(1000), units\_sold int, unit\_price float, unit\_cost float, total\_revenue float, total\_cost float, total\_profit float)

CREATE MASTER KEY ENCRYPTION BY PASSWORD ='Roshan@3003'

create database scoped credential MyAzureBlobStorageCredential

WITH IDENTITY ='SHARED ACCESS SIGNATURE',

SECRET = ‘<SAS token>’

CREATE EXTERNAL DATA SOURCE MyAzureBlobStorage

WITH ( TYPE =BLOB\_STORAGE,

LOCATION = 'https://ecomfiles1.blob.core.windows.net/csvfiles/',

credential = MyAzureBlobStorageCredential)

BULK INSERT dbo.retail from 'sales\_data.csv'

with (check\_constraints,

DATA\_SOURCE = 'MyAzureBlobStorage',

datafiletype ='char',

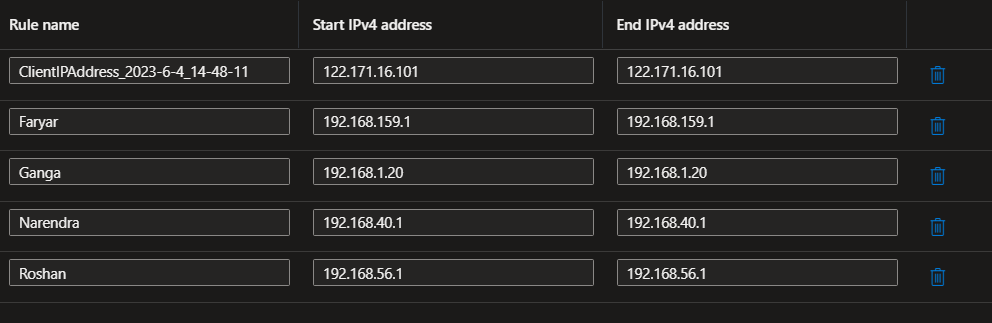
fieldterminator =',',

rowterminator='0x0a',

firstrow=2,

keepidentity,

tablock)



Faryar 192.168.159.1

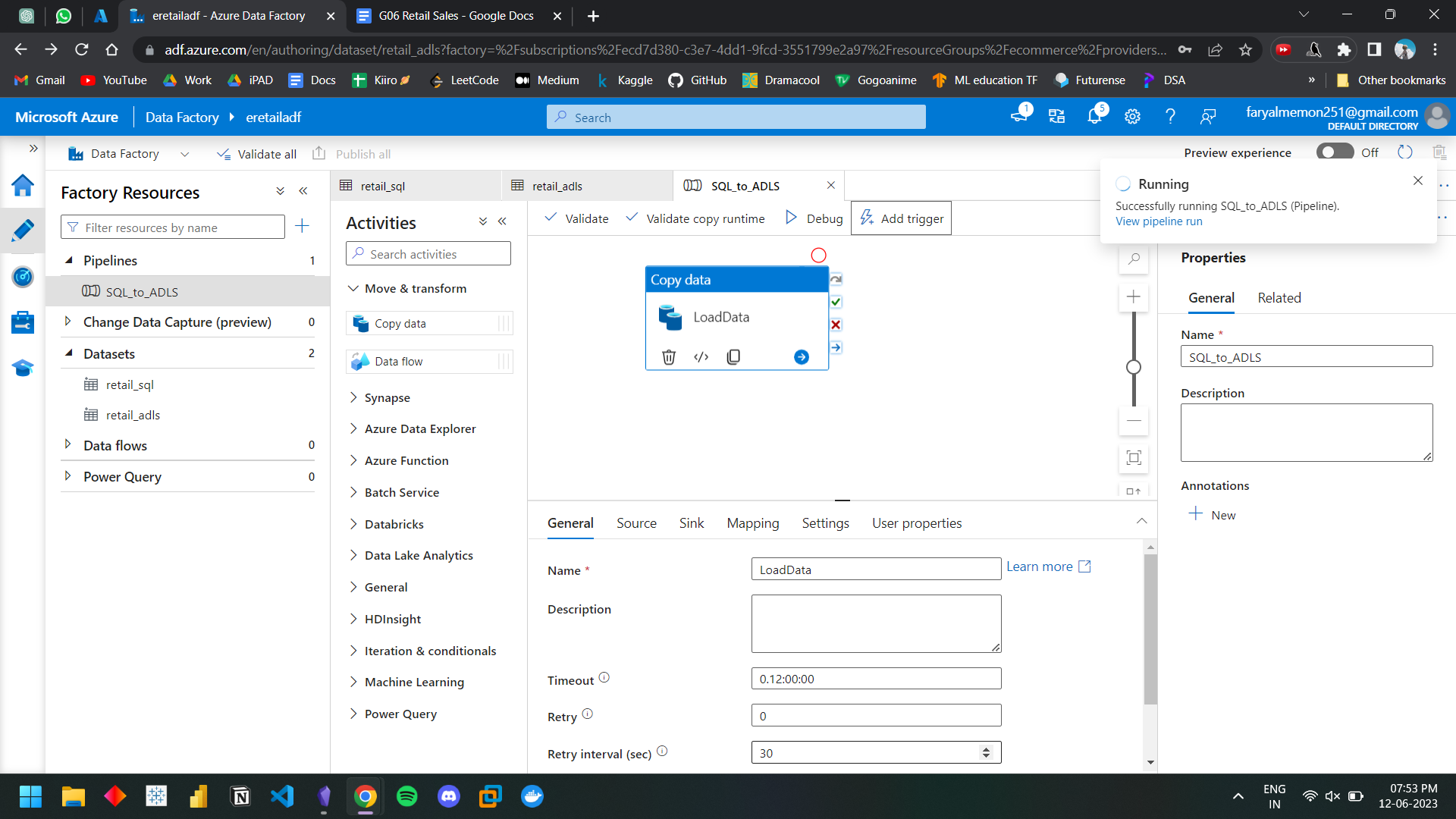
Ganga 192.168.1.20

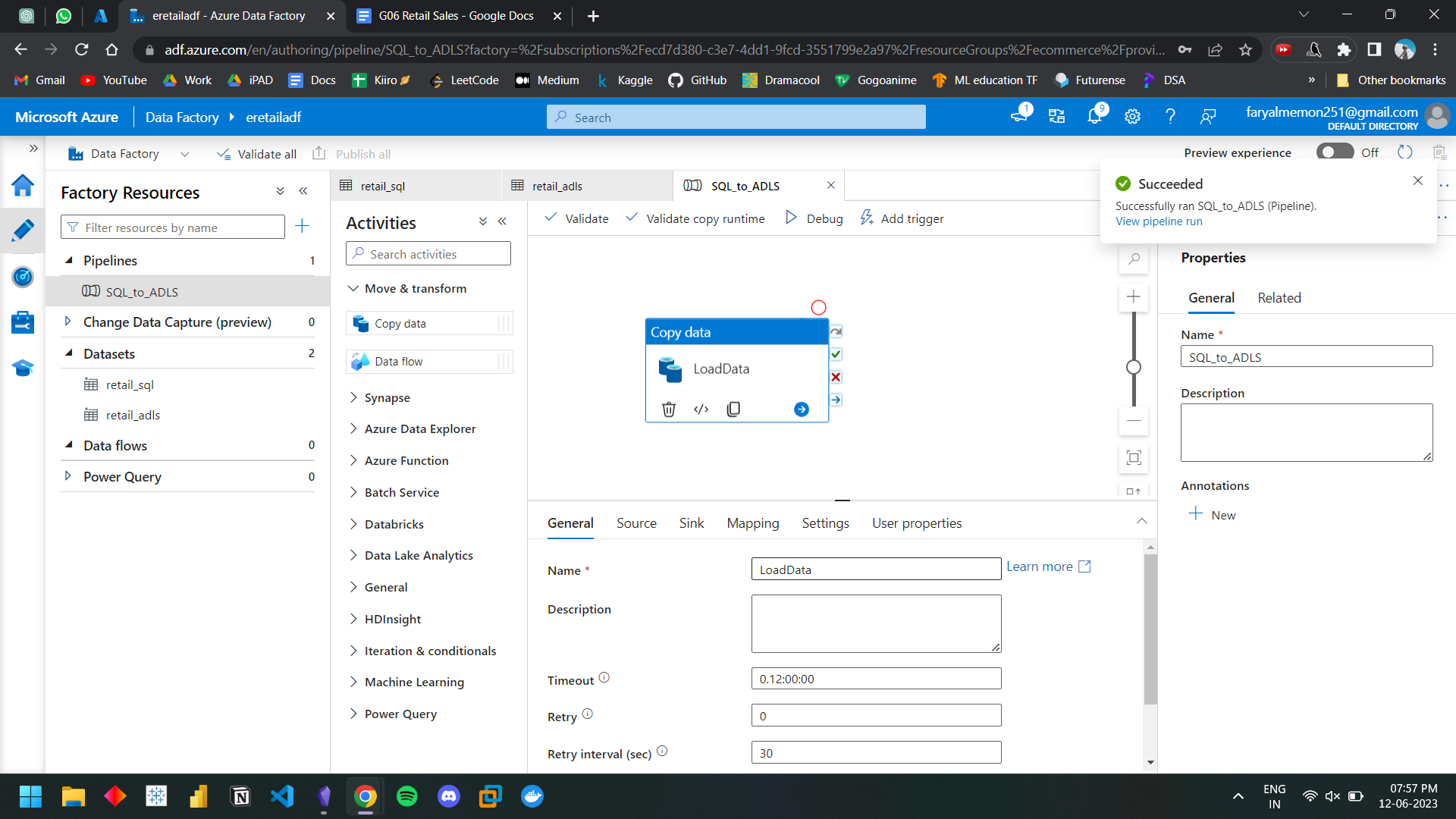
Narendra 192.168.40.1

Roshan 192.168.56.1

Chandan 192.168.187.1

## Transferring Data from Client Source (Azure SQL Database) to ADLS using ADF



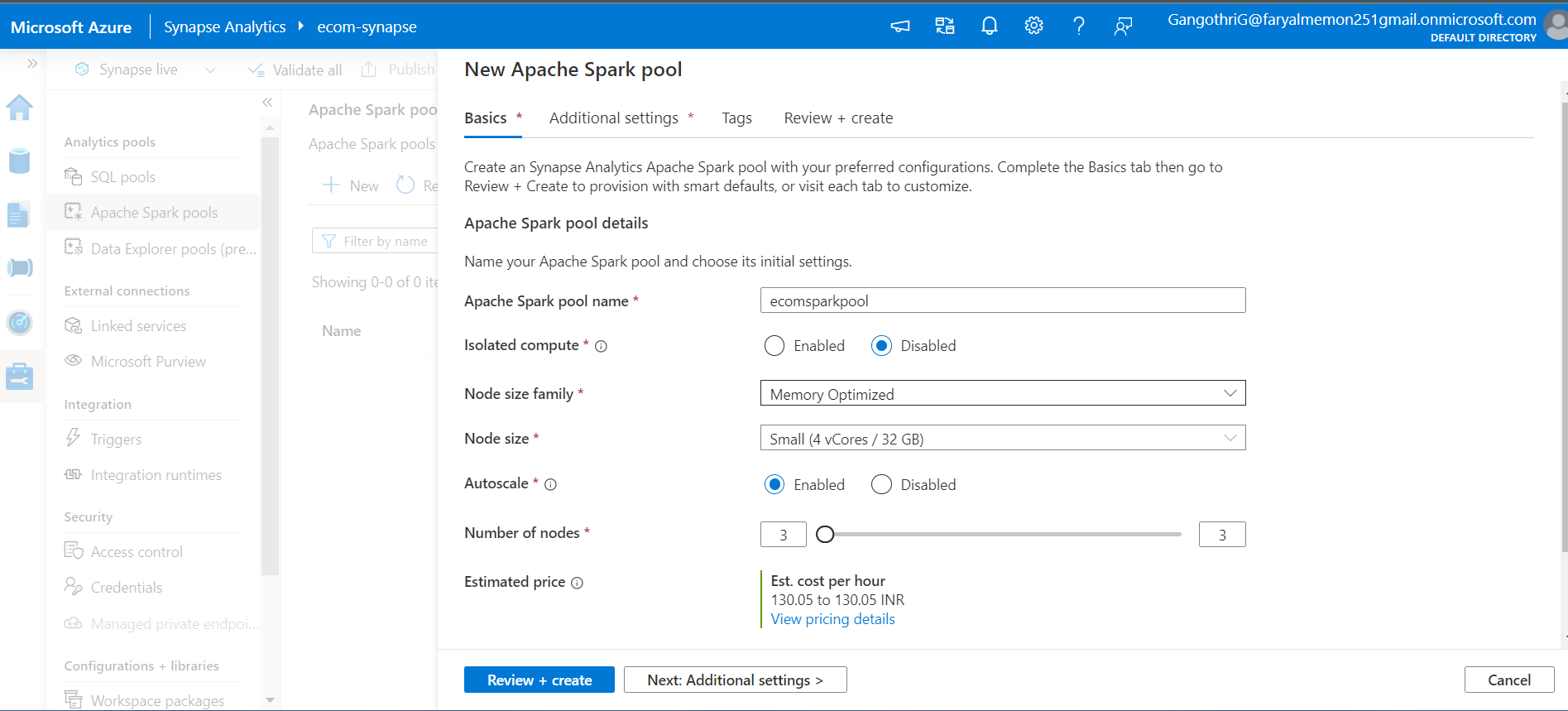


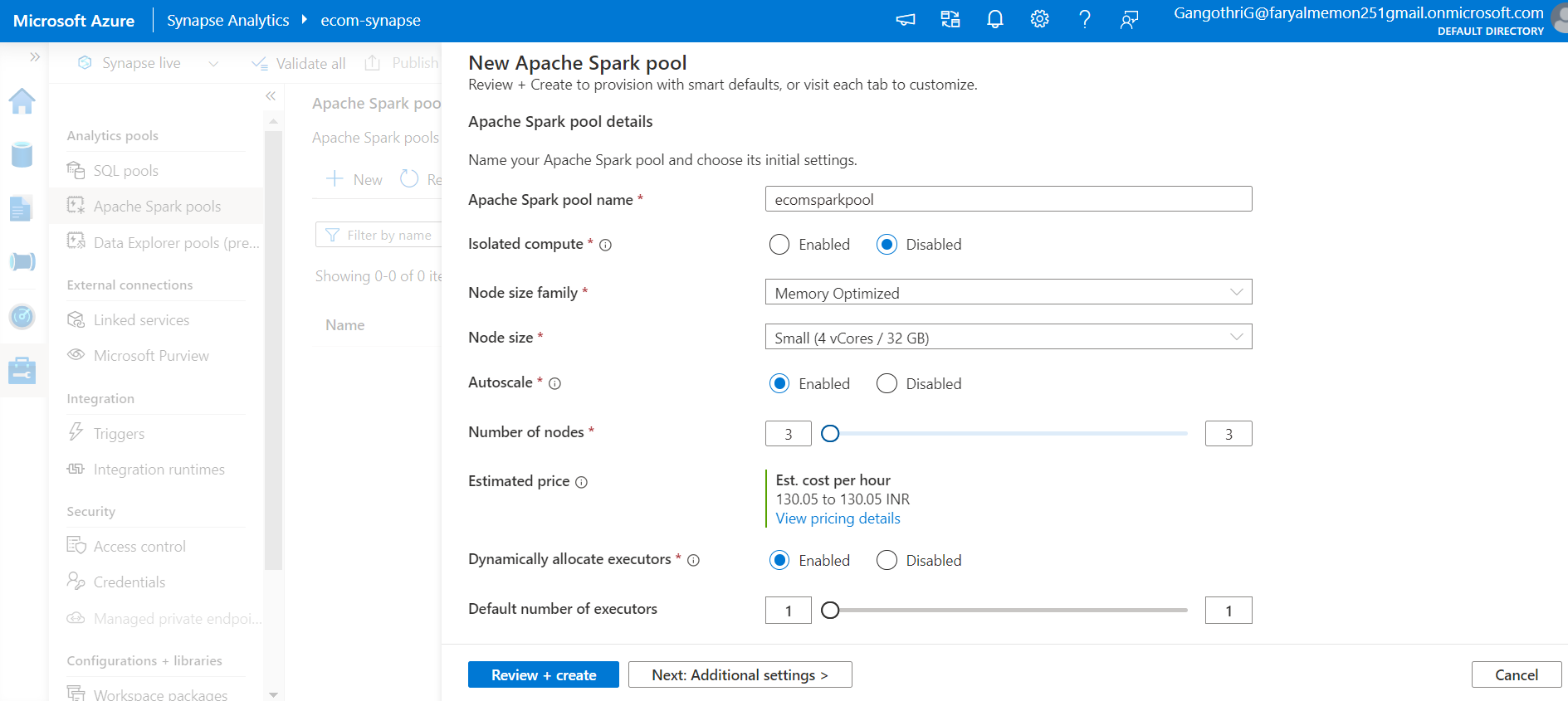


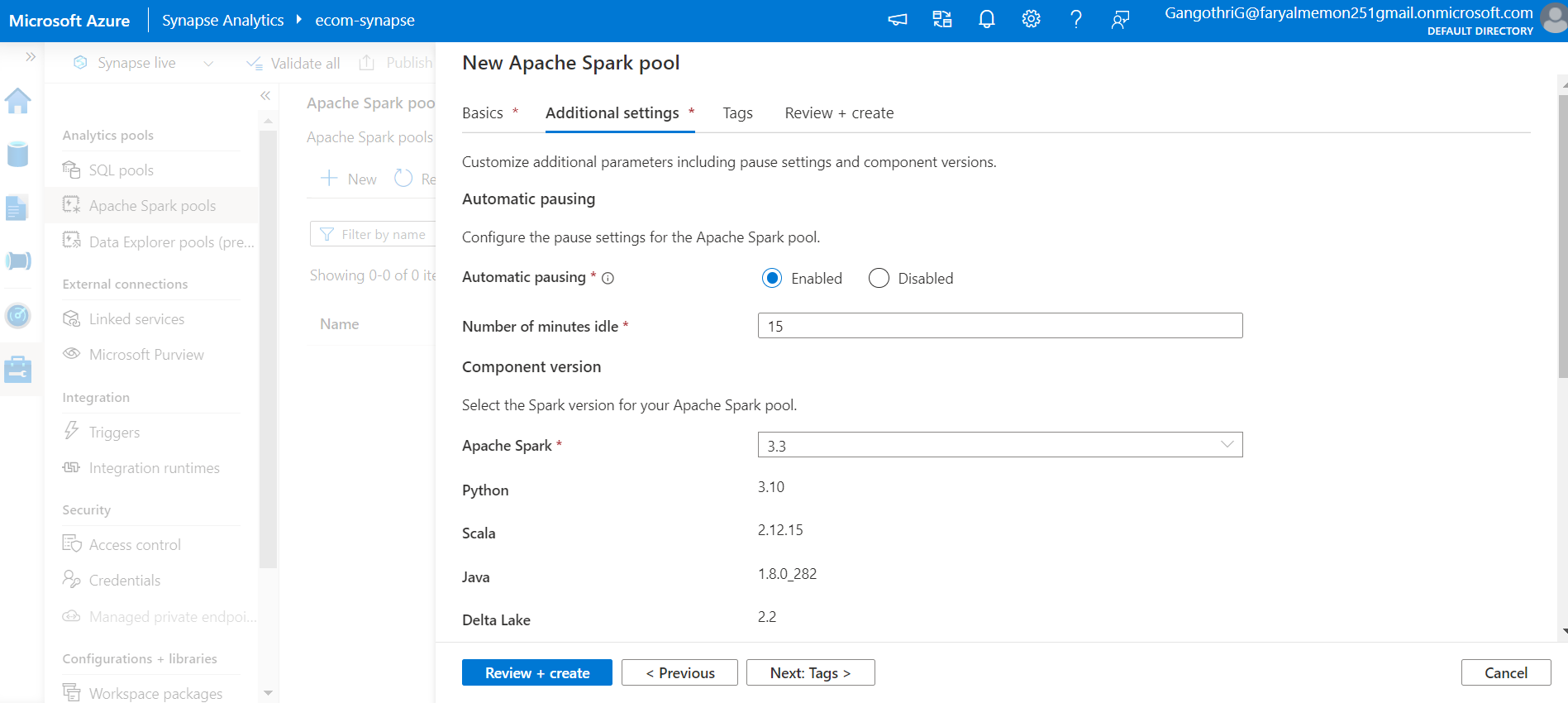
## 

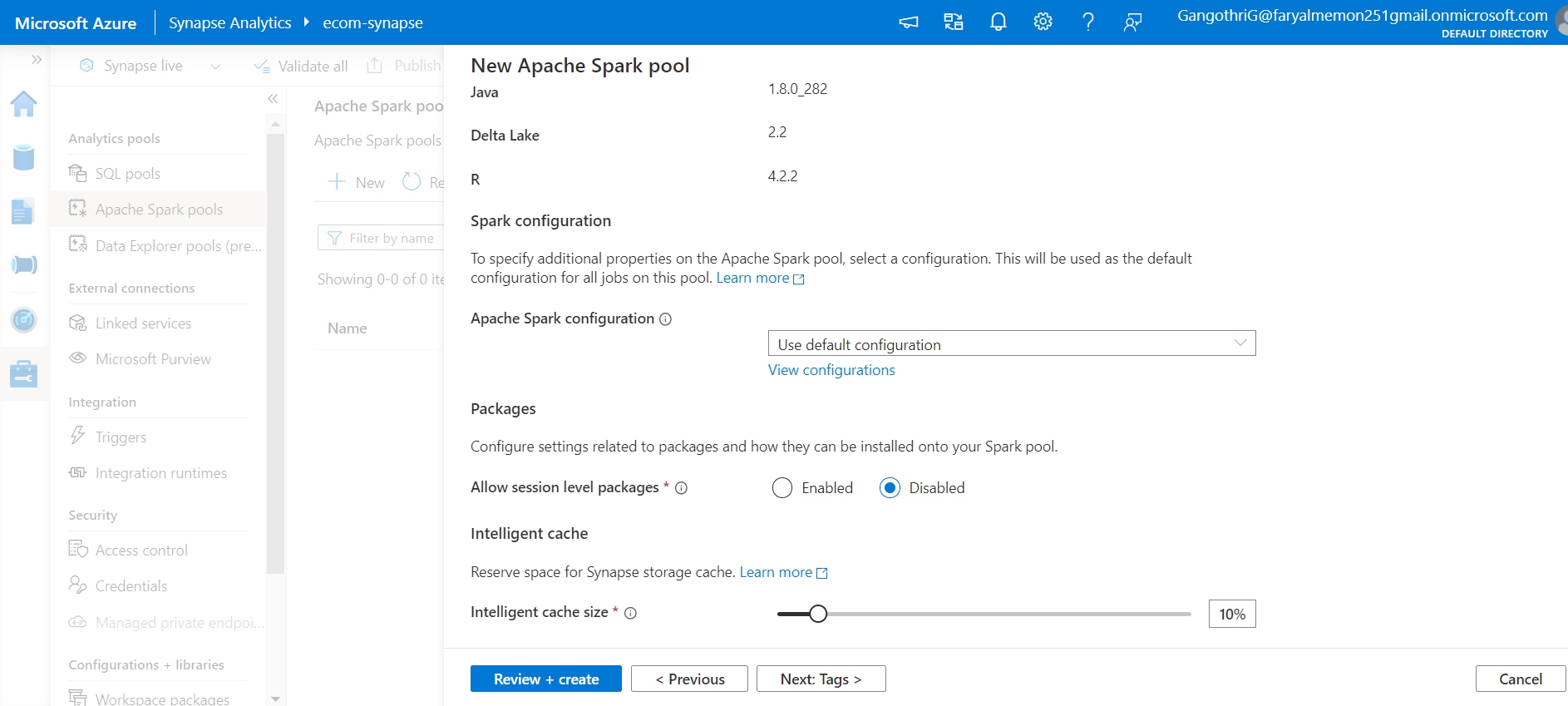
## Analysis in Synapse Studio

### Create Spark Pool and give each user the appropriate permissions of the Synapse Workspace









### Analyzing Queries

**# A.7.Display the number of sales year wise .(Using pyspark)**

from operator import add

year\_and\_revenue = rdd\_df.map(lambda l: (l[5].year, l[11]))

yearly\_sales = year\_and\_revenue.reduceByKey(add).sortBy(lambda x: x[0])

yearly\_sales.collect()

**# A.8. Display the number of orders for each item.(Using pyspark)**

from operator import add

each\_item = rdd\_df.map(lambda l: (l[2], 1))

item\_ocount = each\_item.reduceByKey(add).sortBy(lambda x: x[1], ascending=False)

item\_ocount.collect()

**# A.9. Display the country with highest sale (Using pyspark)**

from operator import add

country\_order = rdd\_df.map(lambda l: (l[1], l[11]))

country\_revenue = country\_order.reduceByKey(add).sortBy(lambda x: x[1], ascending=False)

country\_revenue.take(10)

**# B.4 Yearly sum for specific country**

from operator import add

iyear = 2013

year\_and\_revenue = rdd\_df.map(lambda l: (l[5].year, l[11]))

yearly\_sales = year\_and\_revenue.reduceByKey(add).sortBy(lambda x: x[0])

yearly\_sales.filter(lambda l: l[0]==iyear).map(lambda x: x[1]).first()

**# B.6. monthly sales report of a particular year**

from operator import add

iyear = 2014

each\_month = rdd\_df.map(lambda l: ((l[5].year, l[5].month), l[11]))

yearlymonth\_sales = each\_month.reduceByKey(add).sortBy(lambda x: x[0])

yearlymonth\_sales.filter(lambda l: l[0][0]==iyear).map(lambda x: (x[0][1],x[1])).collect()

**# B. 7) Quarterly sales report of a particular year**

import math

def quarter(date):

return math.ceil(date.month/3)

iyear = 2014

each\_quarterly = rdd\_df.map(lambda l: ((l[5].year, quarter(l[5])), l[11]))

yearlyquarter\_sales = each\_quarterly.reduceByKey(add).sortBy(lambda x: x[0])

yearlyquarter\_sales.filter(lambda l: l[0][0]==iyear).map(lambda x: (x[0][1], x[1])).collect()

**# B. 8, 11) For each quarter report for each year Write udf function for that. And for all year**

import math

def quarter(date):

return math.ceil(date.month/3)

each\_quarter = rdd\_df.map(lambda l: ((l[5].year, quarter(l[5])), l[11]))

yearquarter\_sales = each\_quarter.reduceByKey(add).sortBy(lambda x: x[0])

yearquarter\_sales.collect()

**# B.15. Compare the quarterly sales of each country of the particular year.**

import math

def quarter(date):

return math.ceil(date.month/3)

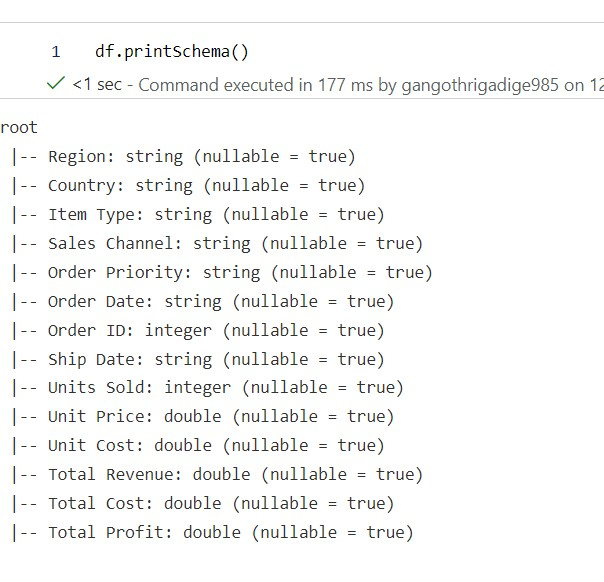
iyear = 2014

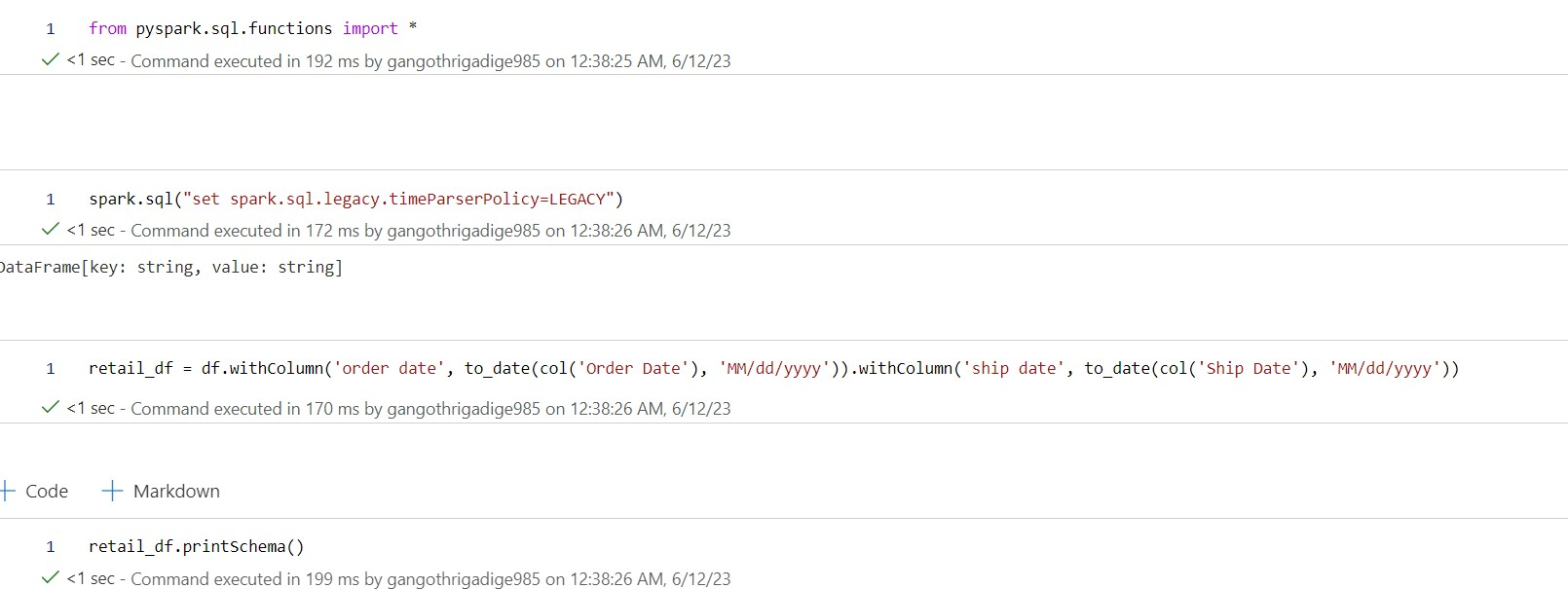
country\_quarterly = rdd\_df.map(lambda l: ((l[5].year, l[1], quarter(l[5])), l[11]))

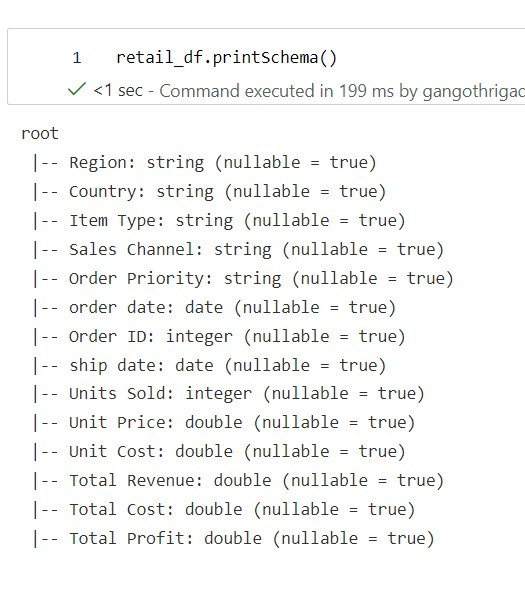
cyearlyquarter\_sales = country\_quarterly.reduceByKey(add).sortBy(lambda x: x[0])

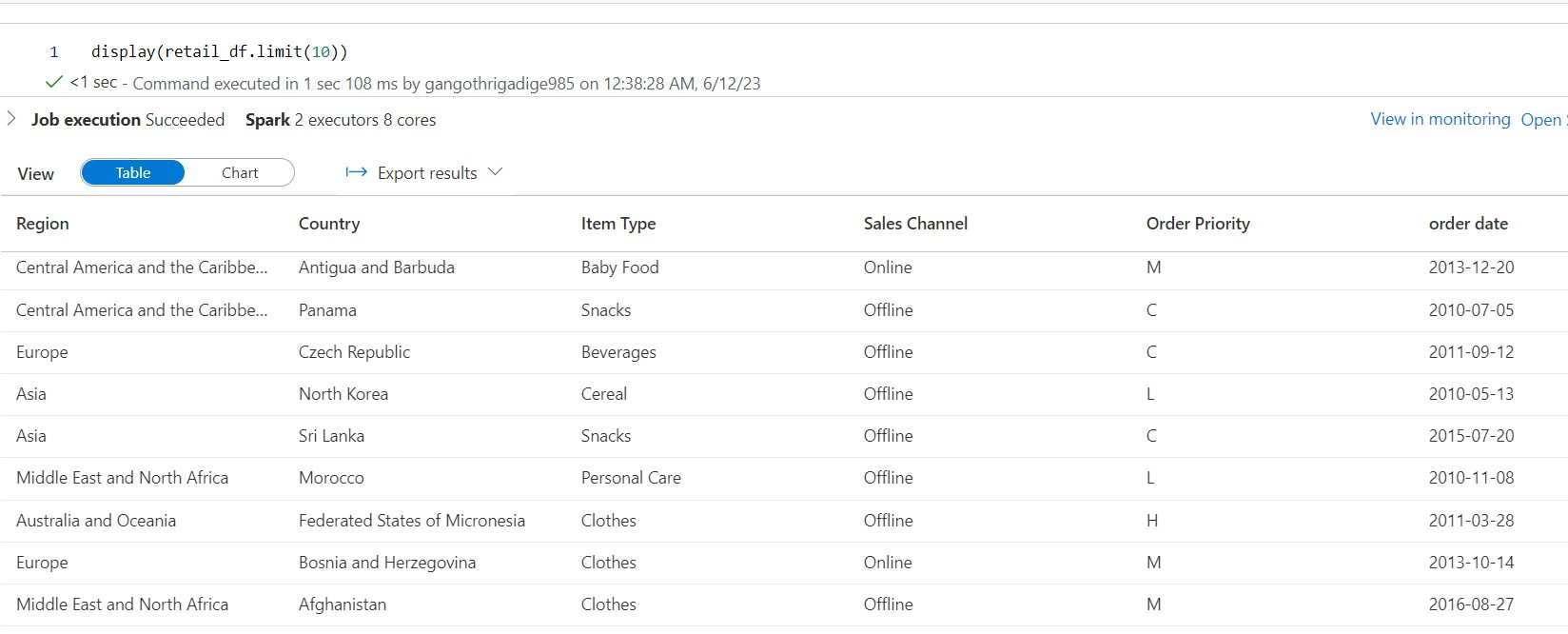
cyearlyquarter\_sales.filter(lambda l: l[0][0]==iyear).map(lambda x: ((x[0][1], x[0][2]), x[1])).take(20)

**Synapse:**

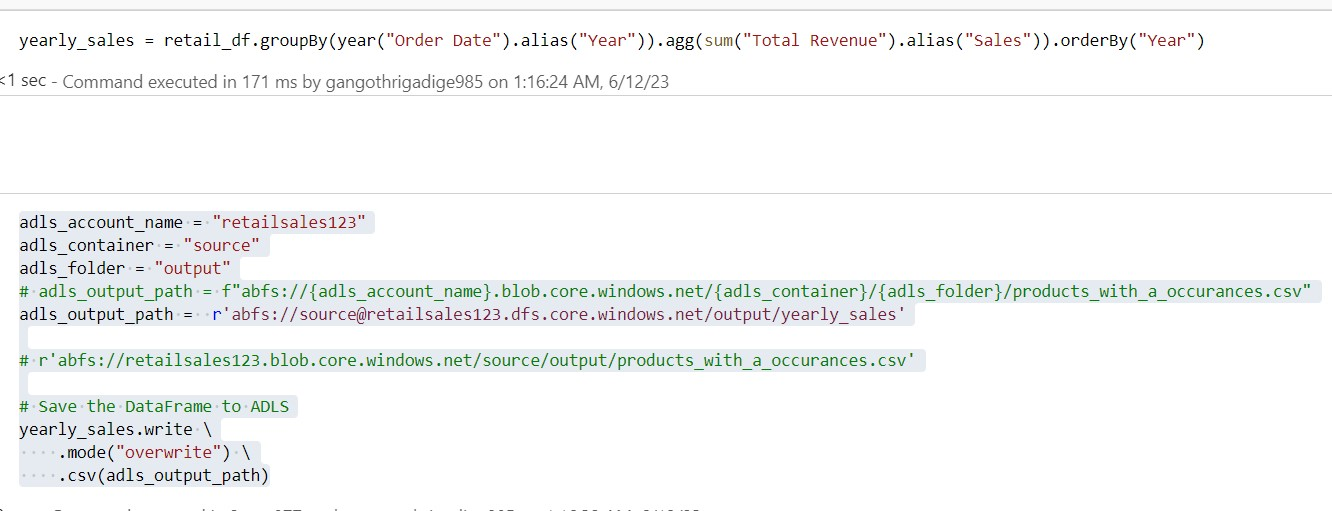
****

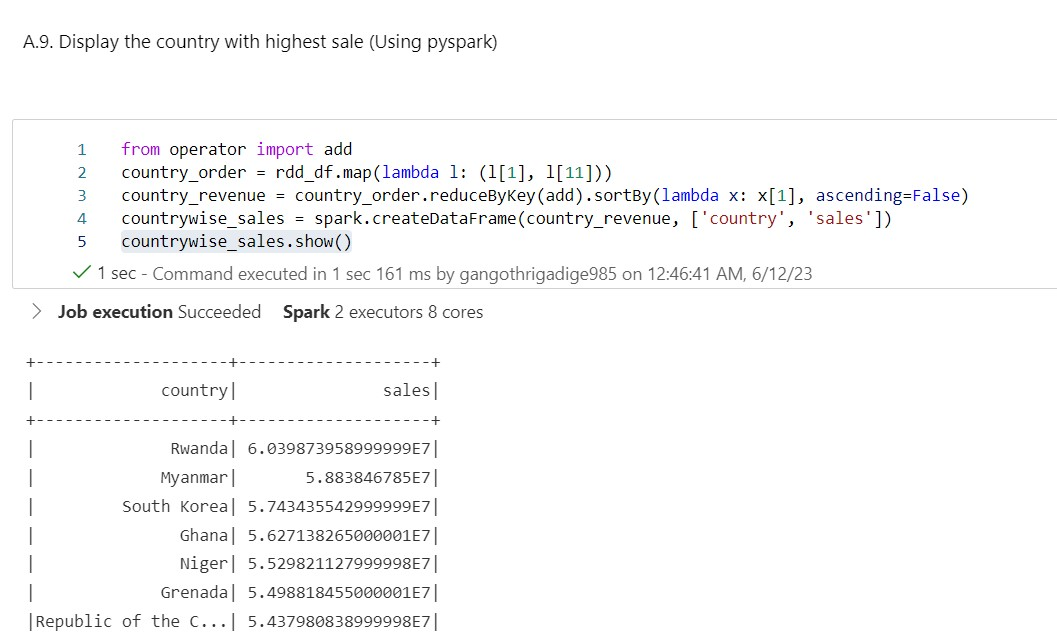
****













%%pyspark

df = spark.read.load('abfss://retaildb@ecomfiles1.dfs.core.windows.net/dbo.retail.csv', format='csv'

## If header exists uncomment line below

, header=True,inferSchema=True

)

display(df.limit(10))

from pyspark.sql.functions import \*

spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")

retail\_df = df.withColumn('order\_date', to\_date(col('order\_date'), 'MM/dd/yyyy')).withColumn('ship\_date', to\_date(col('ship\_date'), 'MM/dd/yyyy'))

retail\_df.printSchema()

root

|-- region: string (nullable = true)

|-- country: string (nullable = true)

|-- item\_type: string (nullable = true)

|-- sales\_channel: string (nullable = true)

|-- order\_priority: string (nullable = true)

|-- order\_date: date (nullable = true)

|-- order\_id: integer (nullable = true)

|-- ship\_date: date (nullable = true)

|-- units\_sold: integer (nullable = true)

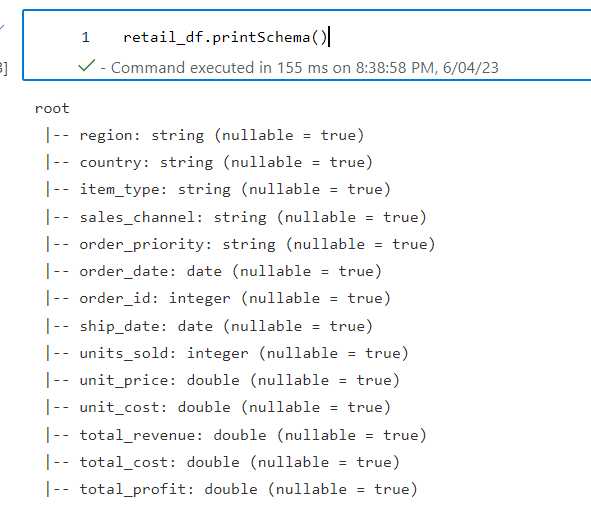
|-- unit\_price: double (nullable = true)

|-- unit\_cost: double (nullable = true)

|-- total\_revenue: double (nullable = true)

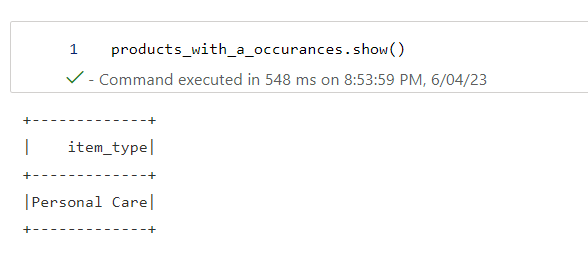
|-- total\_cost: double (nullable = true)

|-- total\_profit: double (nullable = true)



retail\_df.createOrReplaceTempView("retail\_table")

products\_with\_a\_occurances=spark.sql("SELECT distinct item\_type FROM retail\_table WHERE LOWER(item\_type) LIKE '%a%a%' ")



**Databricks**

