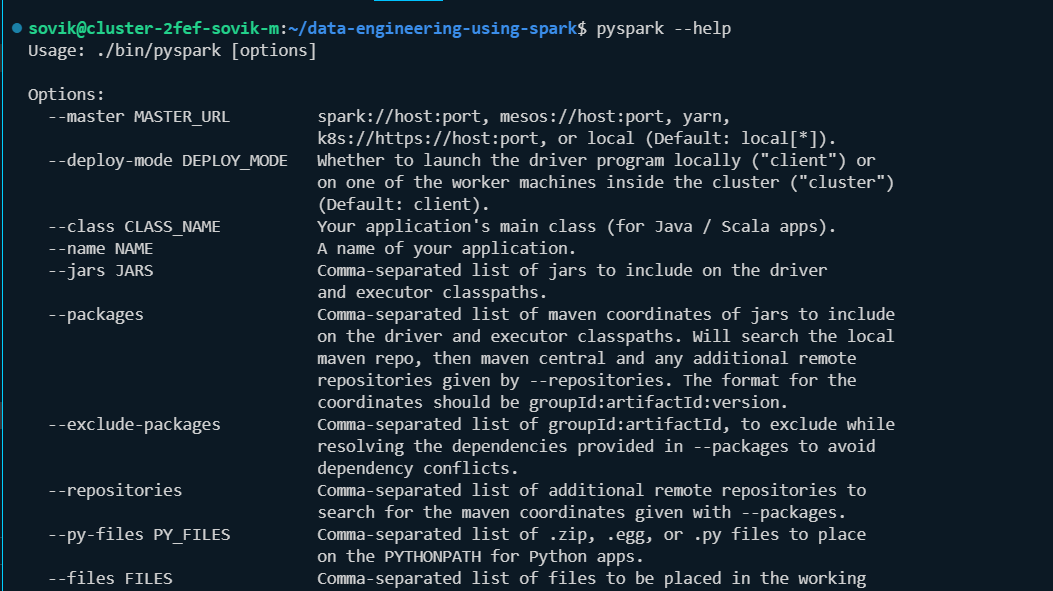
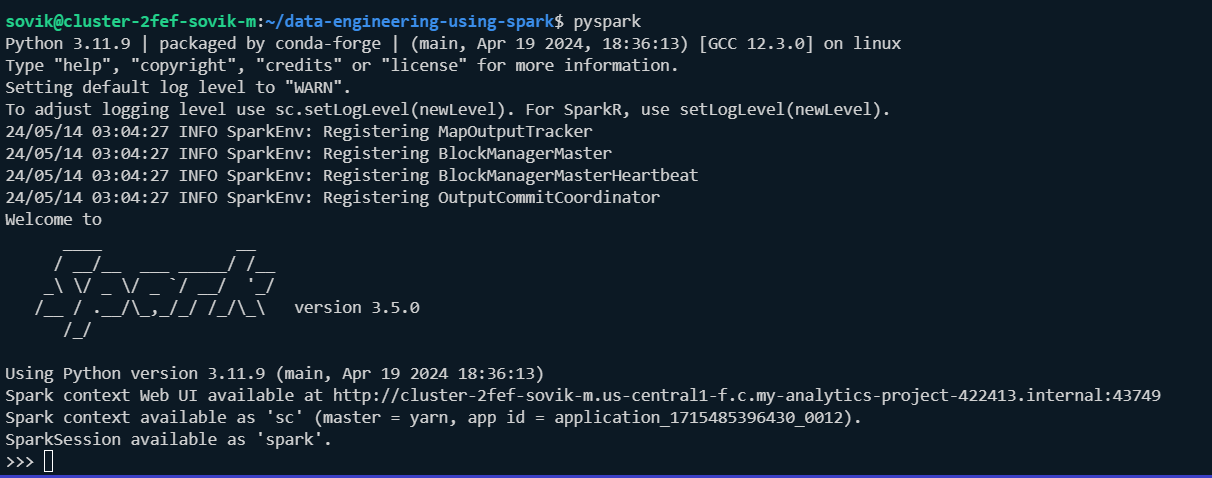
# Pyspark Project

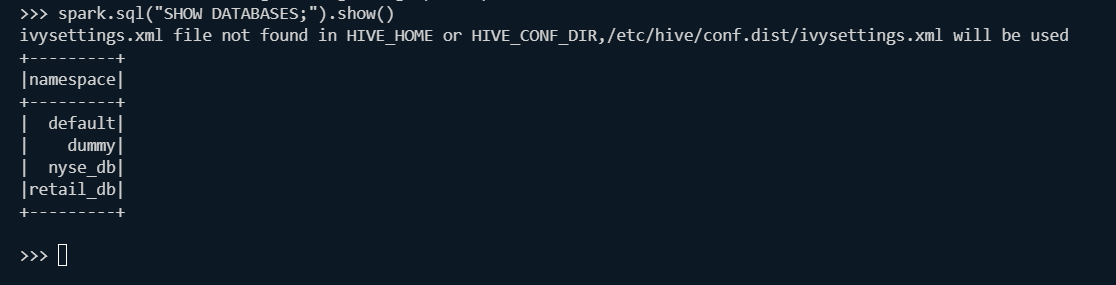
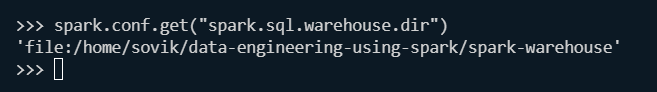
## Introduction

Pyspark –help



Pyspark



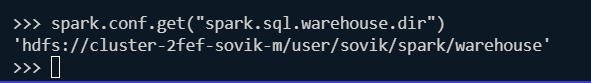


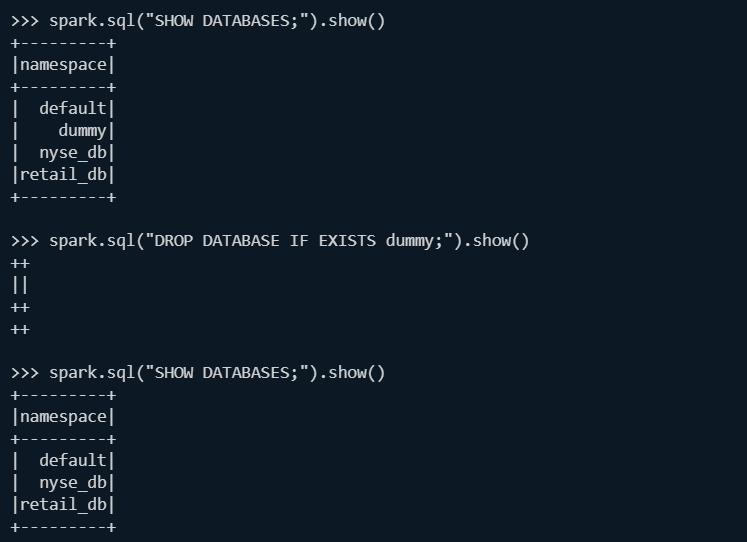
To create we need to launch the pyspark with correct location where we have the privilege of creating database:

pyspark --conf spark.sql.warehouse.dir=/user/`whoami`/spark/warehouse

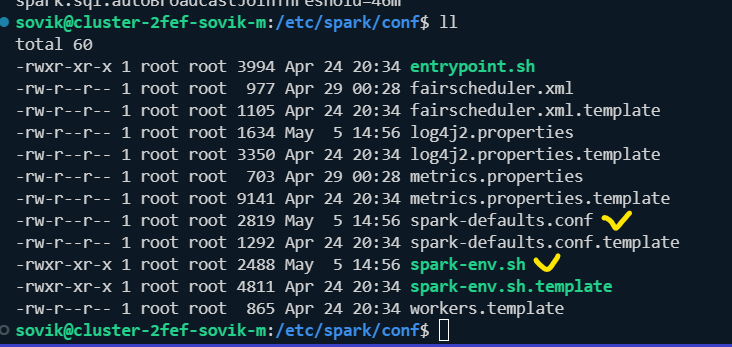


Runtime spark sql warehouse directory;



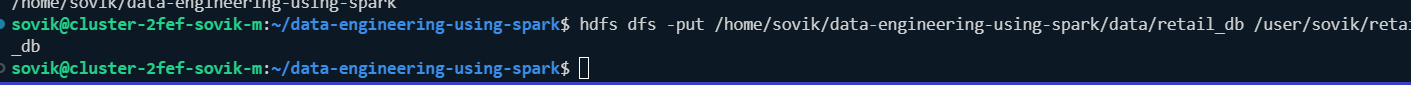


Important spark sql files:

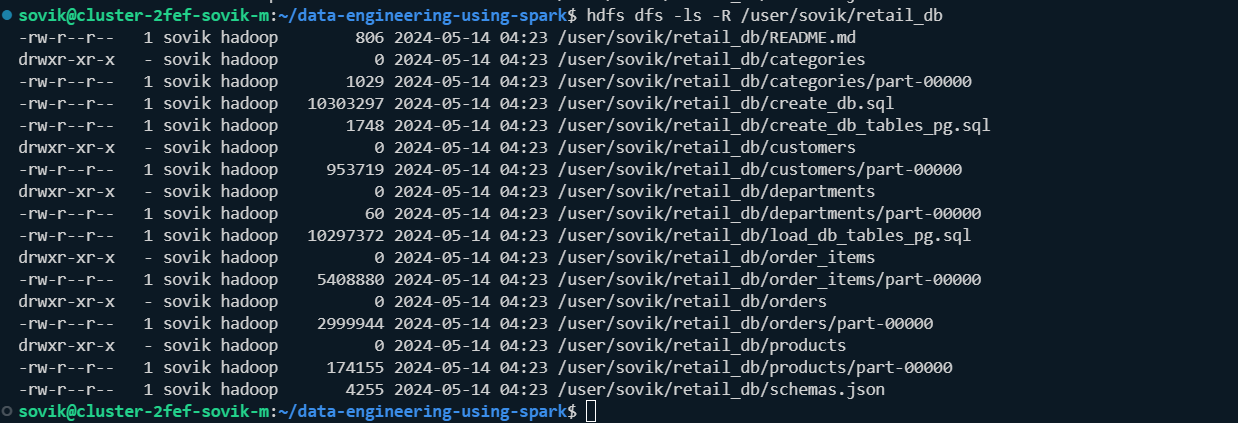


We load the data from local into the hdfs:

hdfs dfs -put /home/sovik/data-engineering-using-spark/data/retail\_db /user/sovik/retail\_db



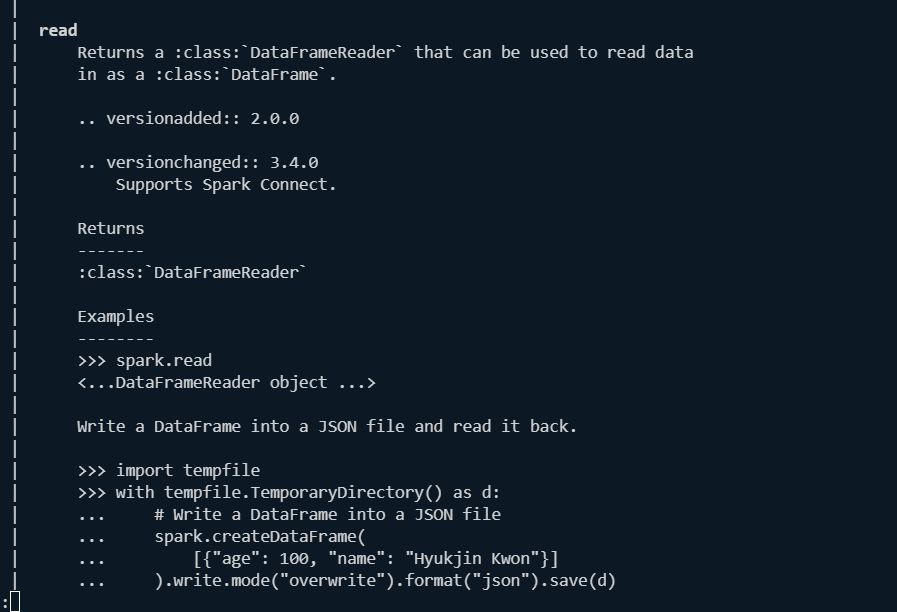
Check:

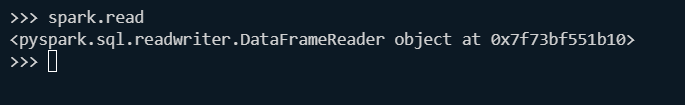


## Getting started with departments example

Login to spark

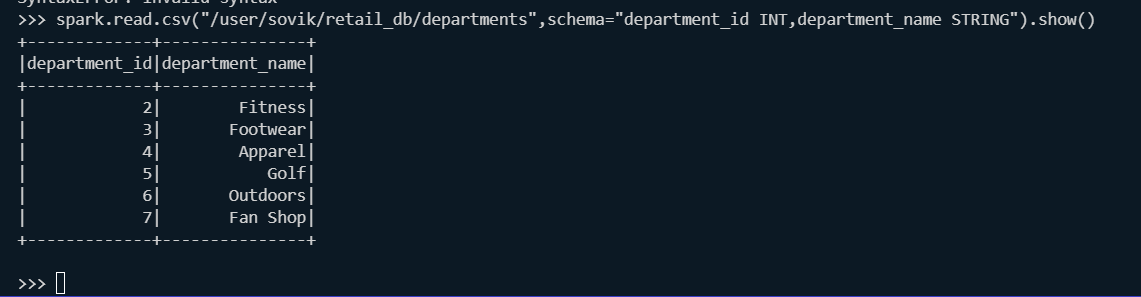
Show doc about spark.read





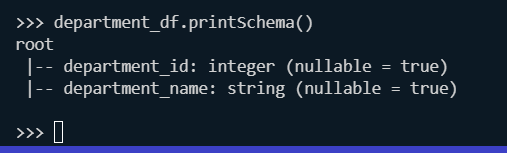
Check the data frames for departments:

spark.read.csv("/user/sovik/retail\_db/departments",schema="department\_id INT,department\_name STRING").show()

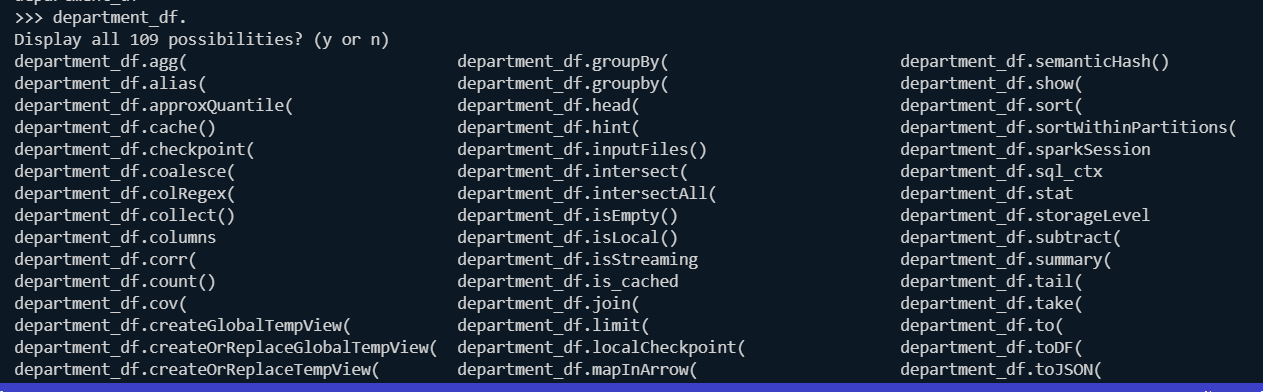


Create department\_df:

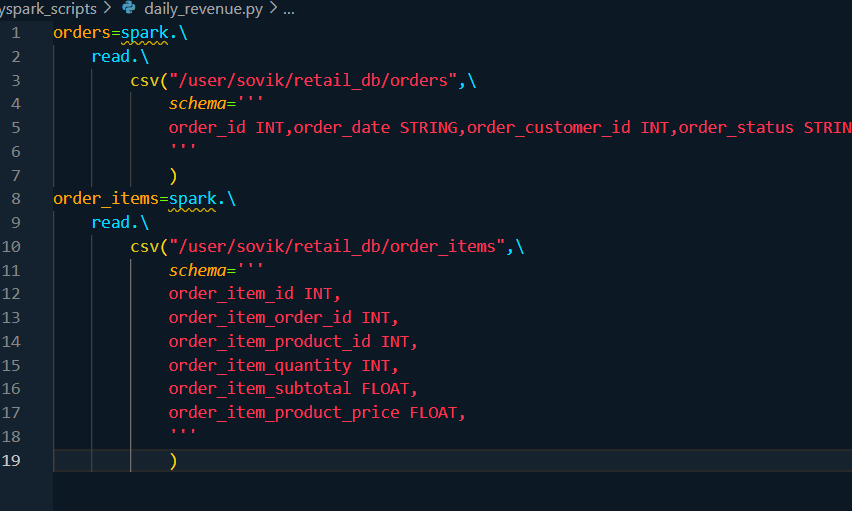
Department\_df= spark.read.csv("/user/sovik/retail\_db/departments",schema="department\_id INT,department\_name STRING")

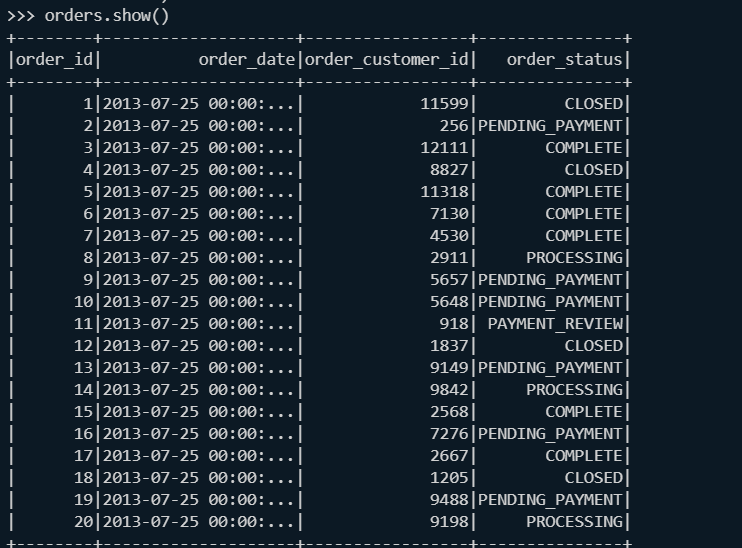
 

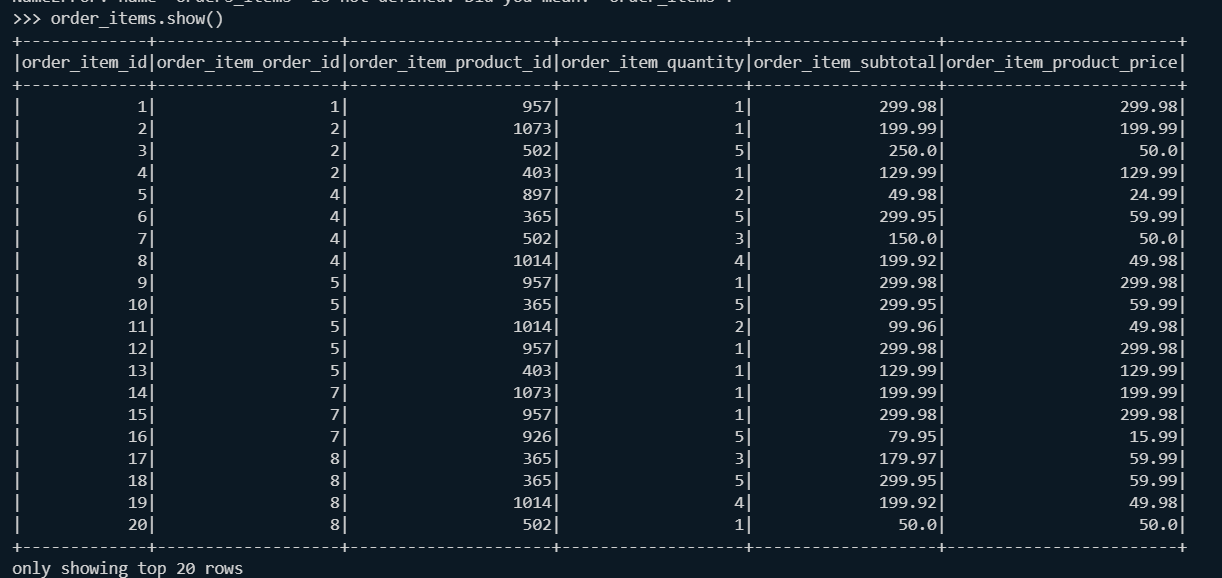
Department\_df.<Press 2\*tab>

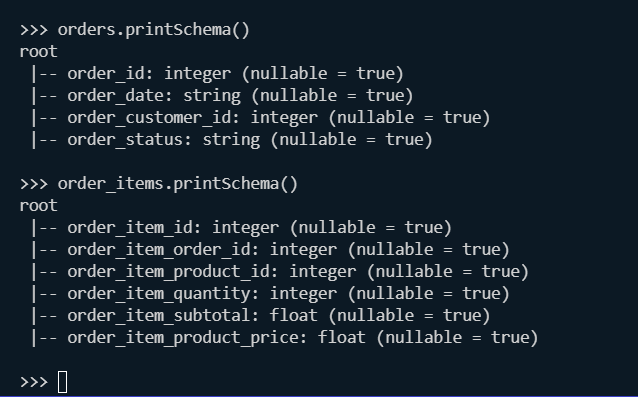


## Create orders and order\_items data frame:









Orders daily:

*from* pyspark.sql.functions *import* sum,round

orders\_daily=orders.filter('''order\_status IN ('COMPLETE','CLOSED')''').\

    join(order\_items,orders['order\_id']==order\_items['order\_item\_id']).\

        groupBy('order\_date').\

            agg(round(sum('order\_item\_subtotal'),2).alias('revenue')).\

                orderBy('order\_date')

>>> from pyspark.sql.functions import sum,round

>>> orders\_daily=orders.filter('''order\_status IN ('COMPLETE','CLOSED')''').\

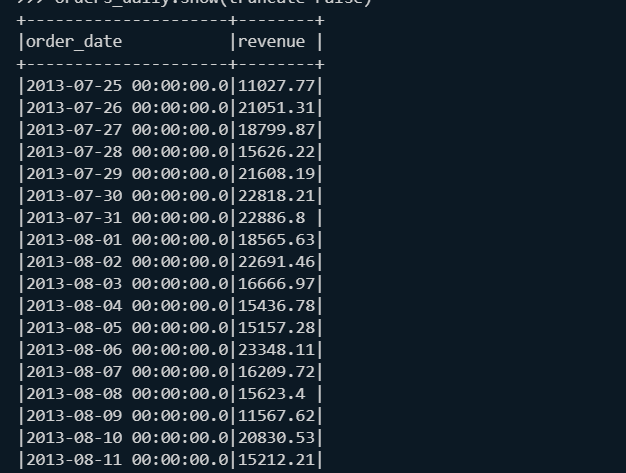
... join(order\_items,orders['order\_id']==order\_items['order\_item\_id']).\

... groupBy('order\_date').\

... agg(round(sum('order\_item\_subtotal'),2).alias('revenue')).\

... orderBy('order\_date')

>>> orders\_daily.show(truncate=False)



Save it in a location;

orders\_daily.\

    toDF('order\_date','order\_revenue').\

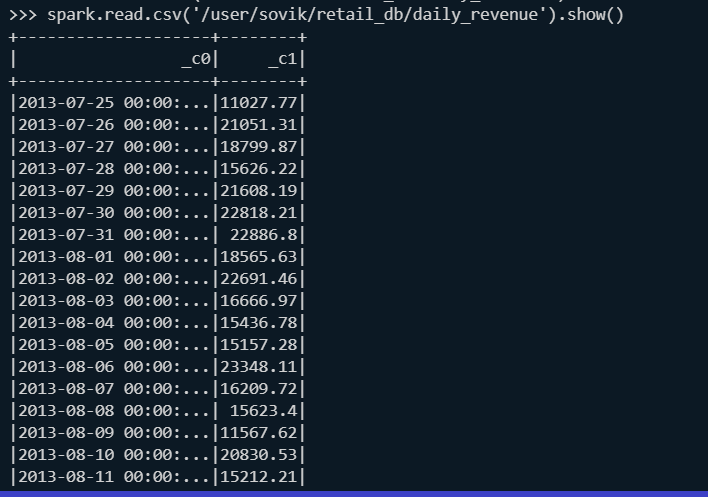
    write.\

        mode('overwrite').\

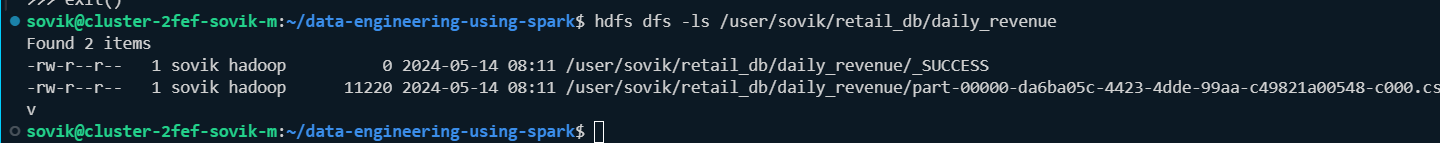
            csv('/user/sovik/retail\_db/daily\_revenue')

Verify:

spark.read.csv('/user/sovik/retail\_db/daily\_revenue').show()

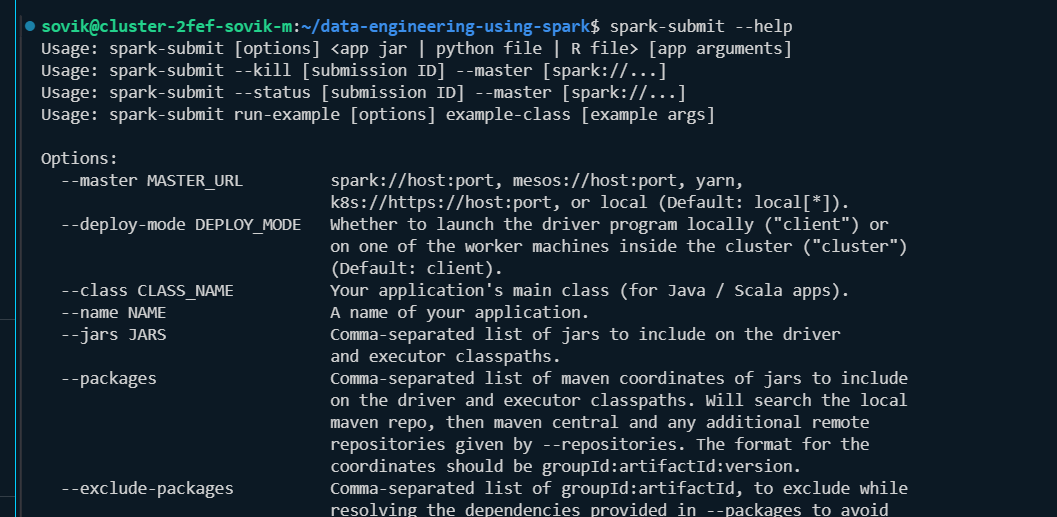


From file system:



## Spark submit

Spark-submit –help



Create the scripts:

One unpartitioned op-version CSV

*from* pyspark.sql *import* SparkSession

*from* pyspark.sql.functions *import* sum, round

*# Initialize Spark Session*

spark = SparkSession. \

    builder. \

    appName('Compute Daily Revenue'). \

    master('yarn'). \

    getOrCreate()

*# Read orders data frame*

orders = spark. \

    read. \

    csv("/user/sovik/retail\_db/orders",

*schema*='''

        order\_id INT, order\_date STRING, order\_customer\_id INT, order\_status STRING

        ''',

*header*=False

    )

*# Read order-items data frame*

order\_items = spark. \

    read. \

    csv("/user/sovik/retail\_db/order\_items",

*schema*='''

        order\_item\_id INT,

        order\_item\_order\_id INT,

        order\_item\_product\_id INT,

        order\_item\_quantity INT,

        order\_item\_subtotal FLOAT,

        order\_item\_product\_price FLOAT

        ''',

*header*=False

    )

*# Filter, join, group and aggregate to get daily revenue*

orders\_daily = orders.filter("order\_status IN ('COMPLETE', 'CLOSED')"). \

    join(order\_items, orders['order\_id'] == order\_items['order\_item\_order\_id']). \

    groupBy('order\_date'). \

    agg(round(sum('order\_item\_subtotal'), 2).alias('revenue')). \

    orderBy('order\_date')

*# Write the result to CSV*

orders\_daily. \

    write. \

    mode('overwrite'). \

    csv('/user/sovik/retail\_db/daily\_revenue')

*# Validate - Display and count rows in the resultant data frame*

orders\_daily.show()

print("Total Rows:", orders\_daily.count())

2nd is partitioned op version parquet:

*from* pyspark.sql *import* SparkSession

*from* pyspark.sql.functions *import* sum,round

spark = SparkSession. \

            builder. \

            appName("Compute Daily Revenue").\

            .master('yarn').\

            getOrCreate()

*#orders data frame-bronze*

orders=spark.\

    read.\

        csv("/user/sovik/retail\_db/orders",\

*schema*='''

            order\_id INT,order\_date STRING,order\_customer\_id INT,order\_status STRING

            '''

            )

*# Extract order\_month and filter orders*

orders = orders.withColumn('order\_month', substr(col('order\_date'), 1, 7))

*#order-items data frame-bronze*

order\_items=spark.\

    read.\

        csv("/user/sovik/retail\_db/order\_items",\

*schema*='''

            order\_item\_id INT,

            order\_item\_order\_id INT,

            order\_item\_product\_id INT,

            order\_item\_quantity INT,

            order\_item\_subtotal FLOAT,

            order\_item\_product\_price FLOAT

            '''

            )

*#order-daily data-frame silver*

orders\_daily=orders.filter('''order\_status IN ('COMPLETE','CLOSED')''').\

    join(order\_items,orders['order\_id']==order\_items['order\_item\_id']).\

        groupBy('order\_date').\

            agg(round(sum('order\_item\_subtotal'),2).alias('revenue')).\

                orderBy('order\_date')

*#daily revenue data-frame silver*

daily\_revenue\_partitioned=orders\_daily.\

    toDF('order\_date','order\_revenue').\

    write.\

        mode('overwrite').\

            partitionBy('order\_month').\

            parquet('/user/sovik/retail\_db/daily\_revenue\_parquet')

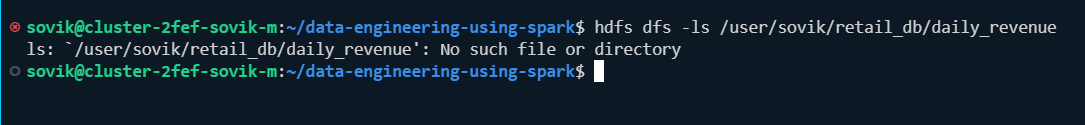
*#validate*

daily\_revenue\_partitioned.show()

daily\_revenue\_partitioned.count()

cleanup:



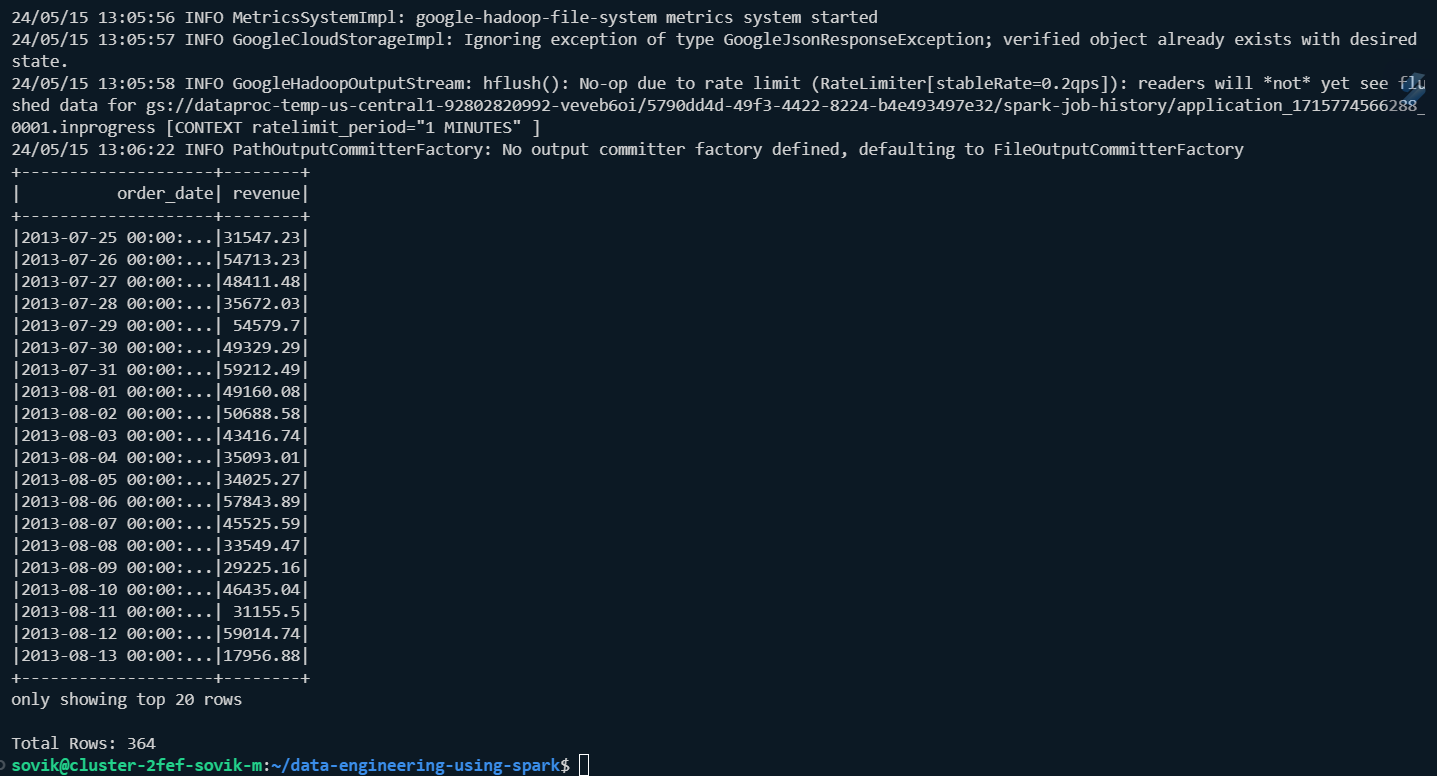


Now we use spark submit:

### Client-mode: For csv output we have the following command(non-partitioned ):

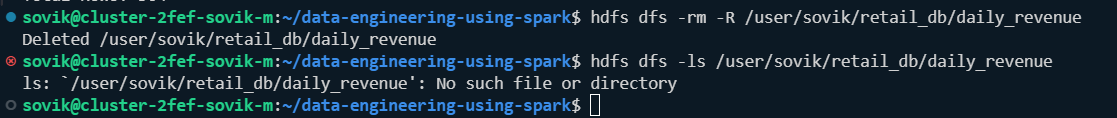
spark-submit /home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py

O/P: Validation:



### Cluster-mode: For csv output we have the following command(non-partitioned ):

Cleanup:



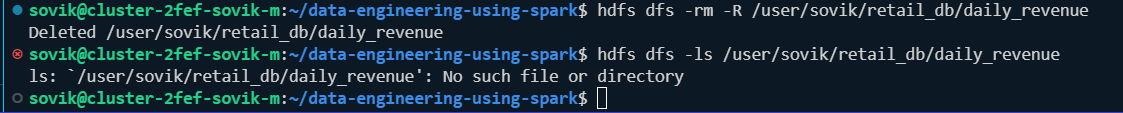
Command:

spark-submit --deploy-mode cluster /home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py

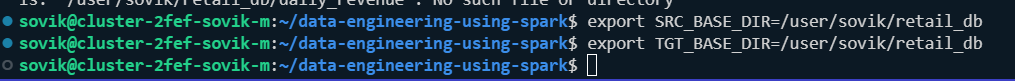
O/P: 

## Create Environment Variable

Cleanup



Provide environment variable:

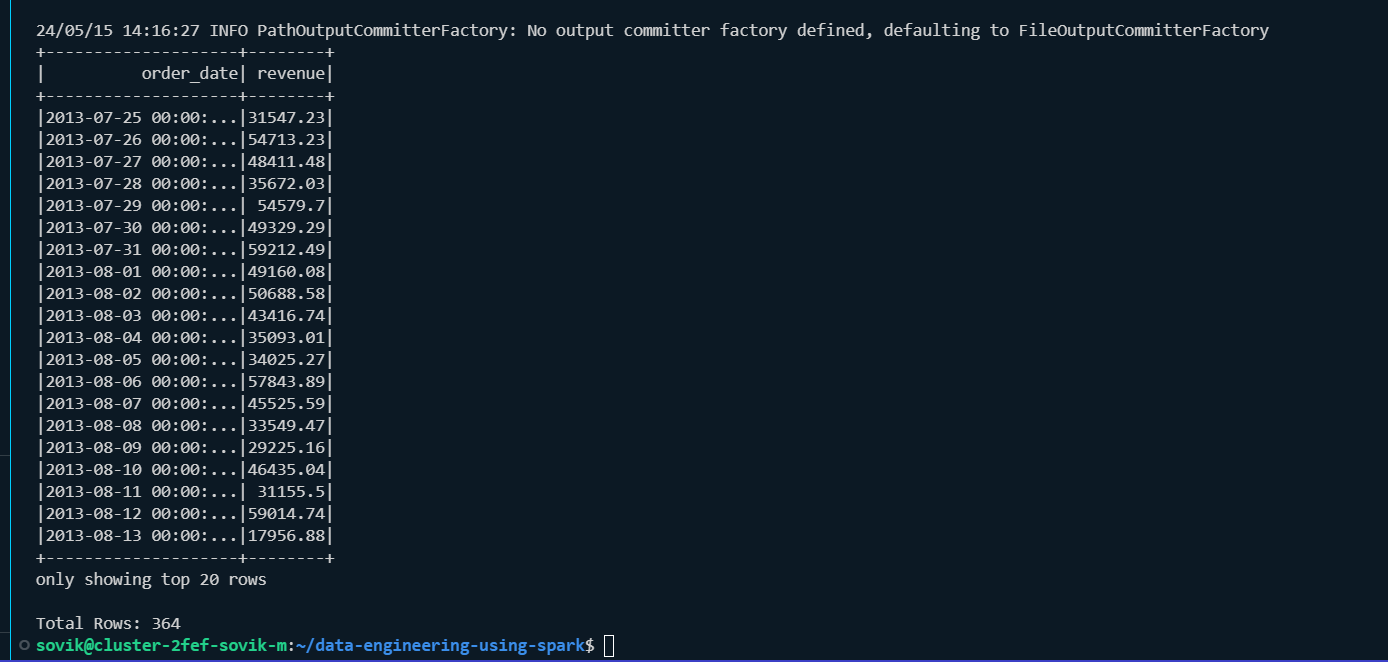


Note: in my case both are same

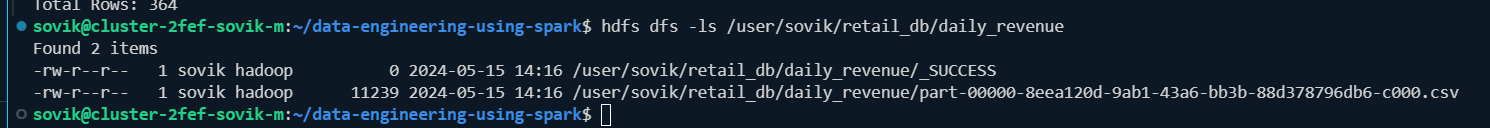
Deployin client mode:(non-partitioned):

**spark-submit --deploy-mode client /home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py**

output:

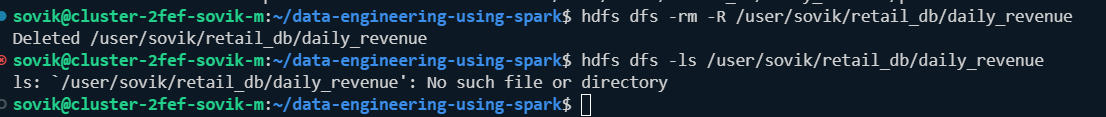


Validate file:

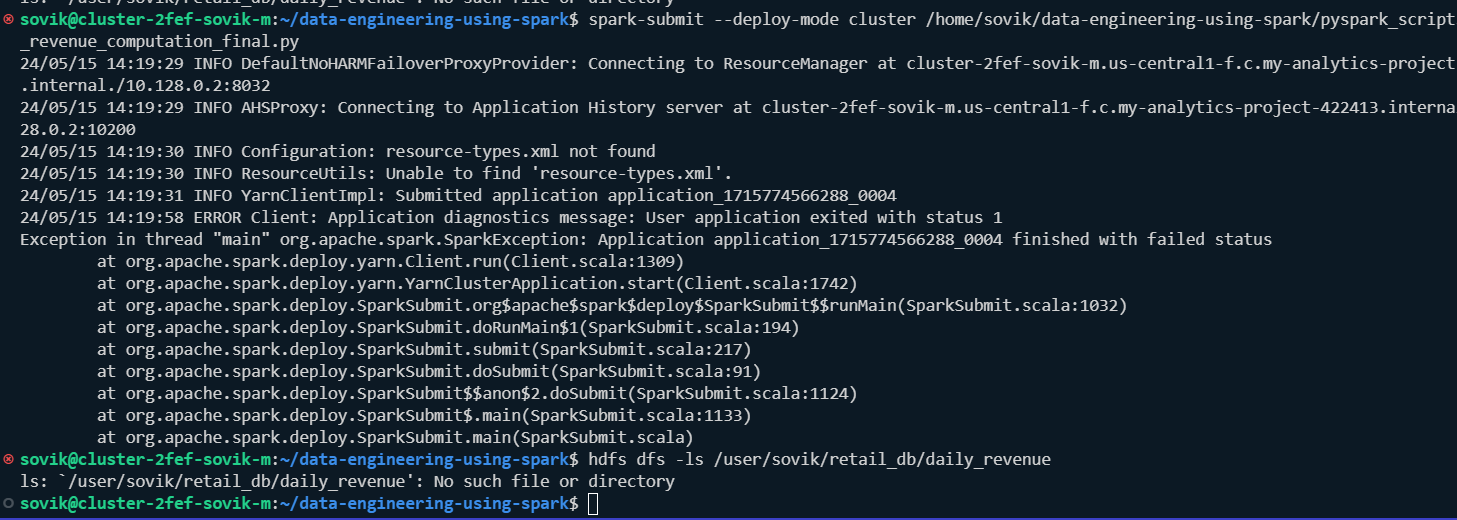


Deploy in cluster mode:(non-partitioned):

Cleanup:



**spark-submit --deploy-mode cluster /home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py**

****

**We get this error lets check the environment variables:**

****

Note: When running Spark on YARN in cluster mode, environment variables need to be set using the spark.yarn.appMasterEnv.[EnvironmentVariableName] property in your conf/spark-defaults.conf file. Environment variables that are set in spark-env.sh will not be reflected in the YARN Application Master process in cluster mode. See the [YARN-related Spark Properties](https://spark.apache.org/docs/3.5.0/running-on-yarn.html#spark-properties) for more information.

Our renewed command is:

**spark-submit --deploy-mode cluster --conf spark.yarn.appMasterEnv.SRC\_BASE\_DIR=/user/sovik/retail**

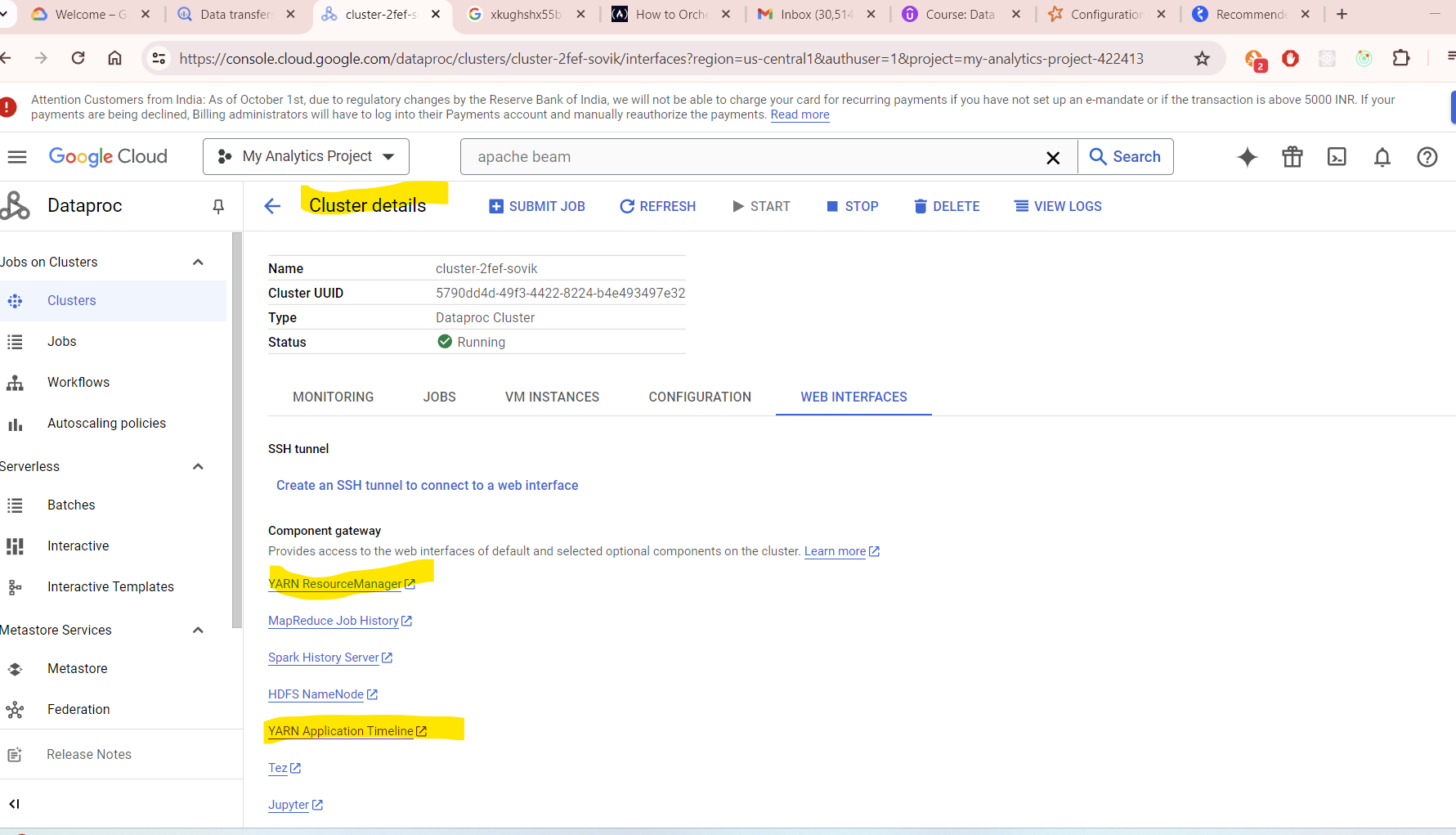
**\_db --conf spark.yarn.appMasterEnv.TGT\_BASE\_DIR=/user/sovik/retail\_db /home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_fin**

**al.py**

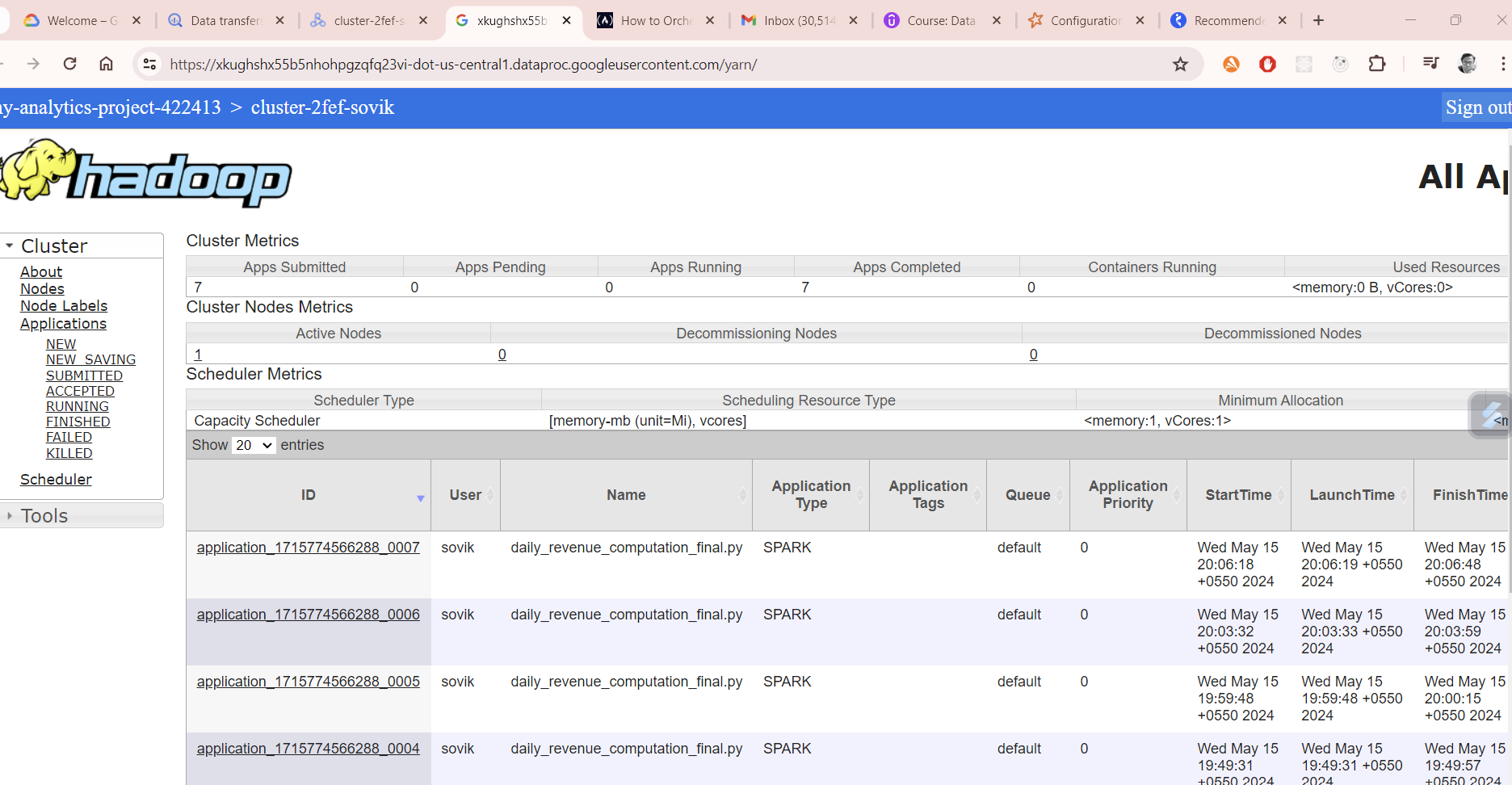
**Validated:**

****

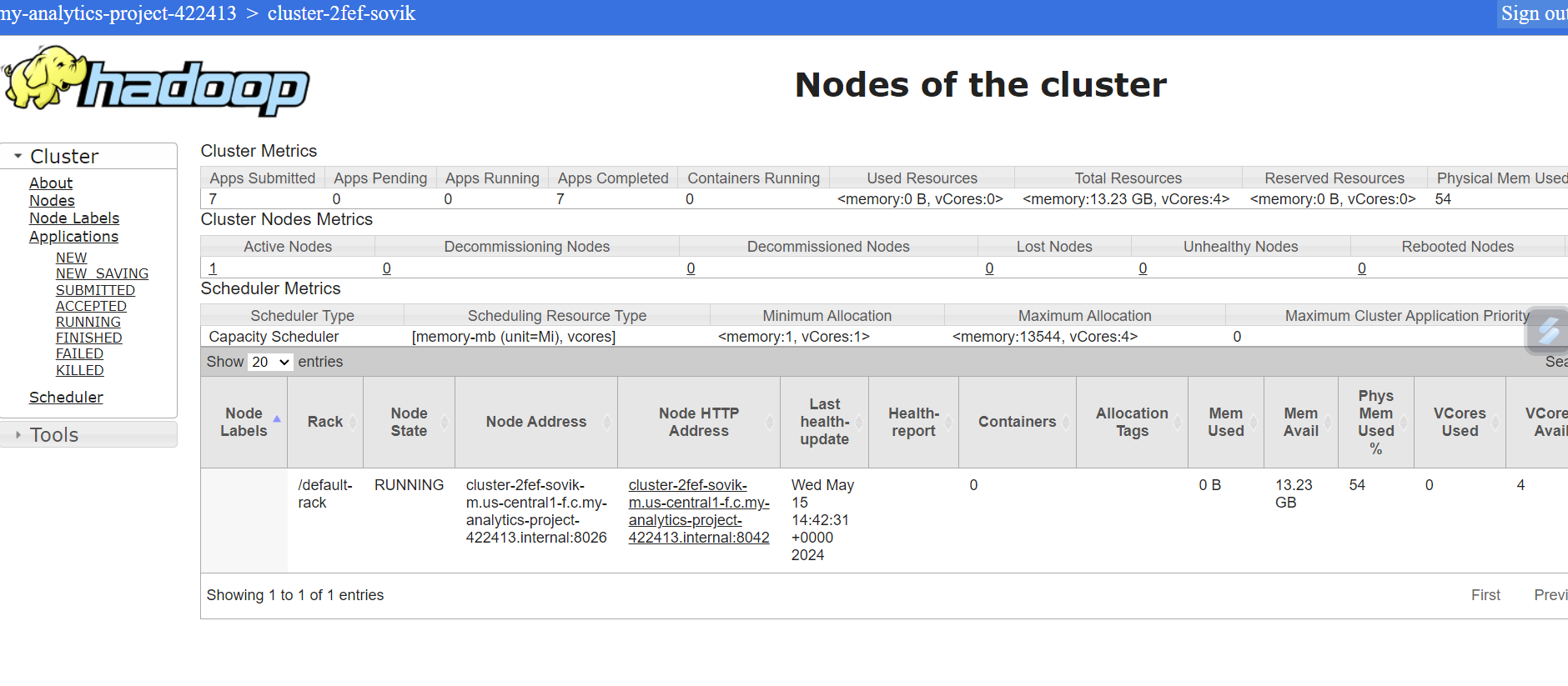
## See the details in web interface:



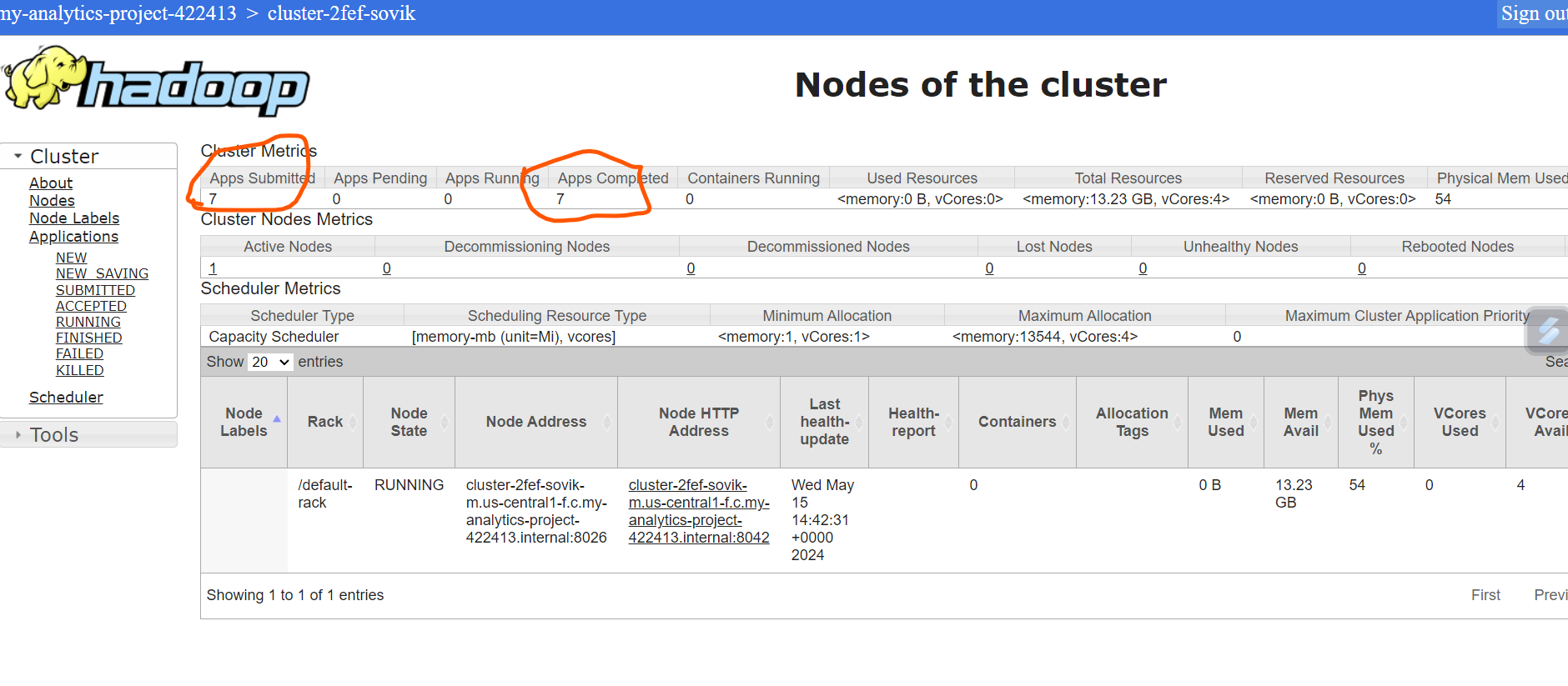
Yarn resource manager:



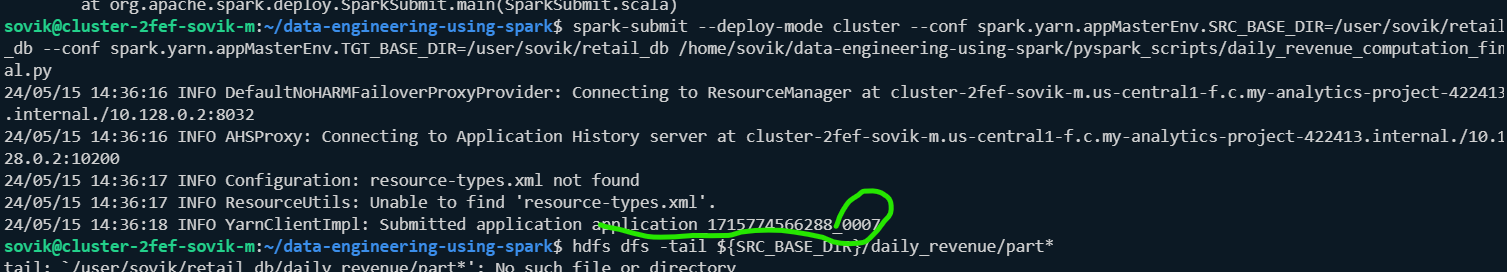
Nodes:



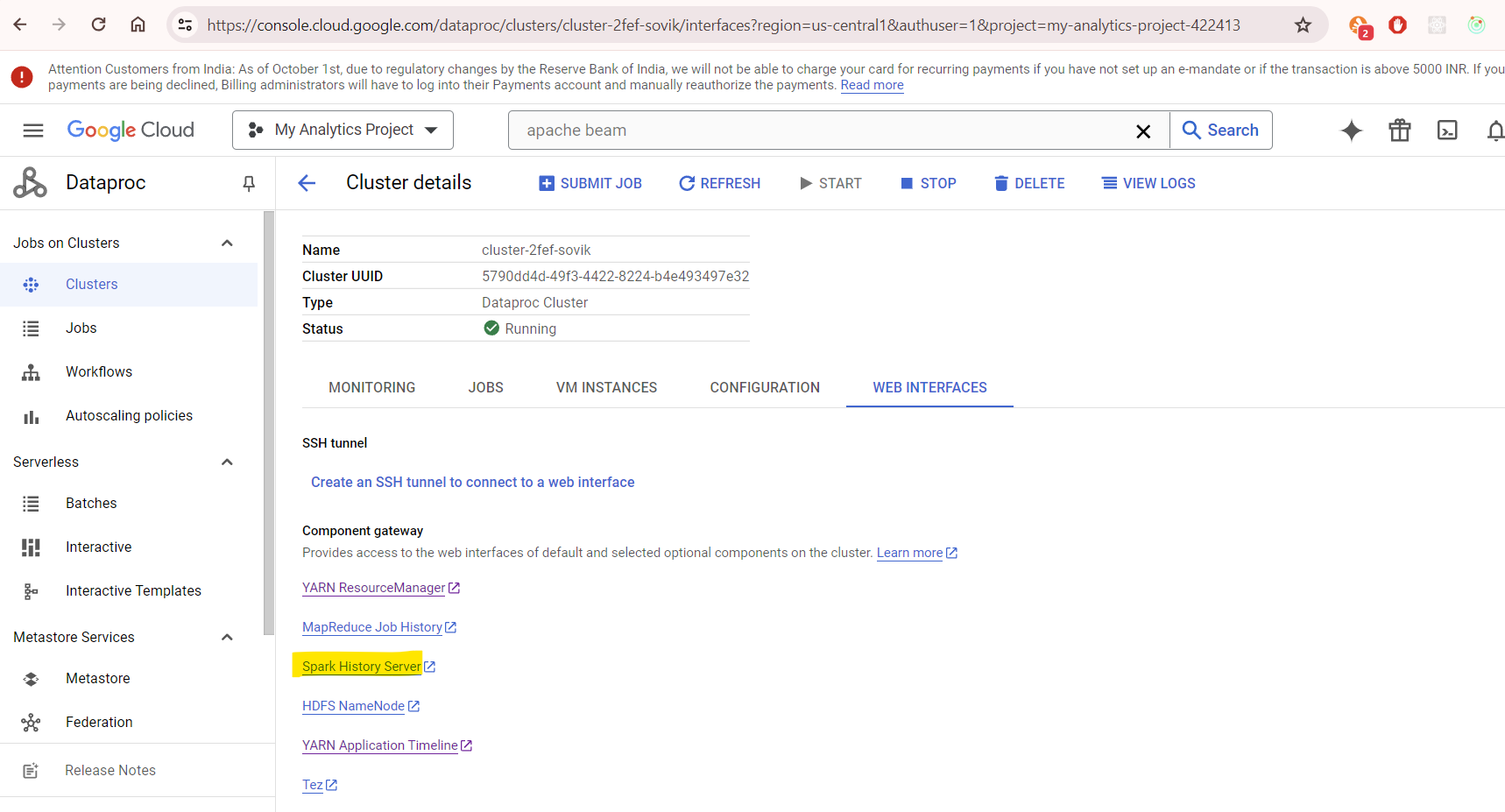
Apps completed:

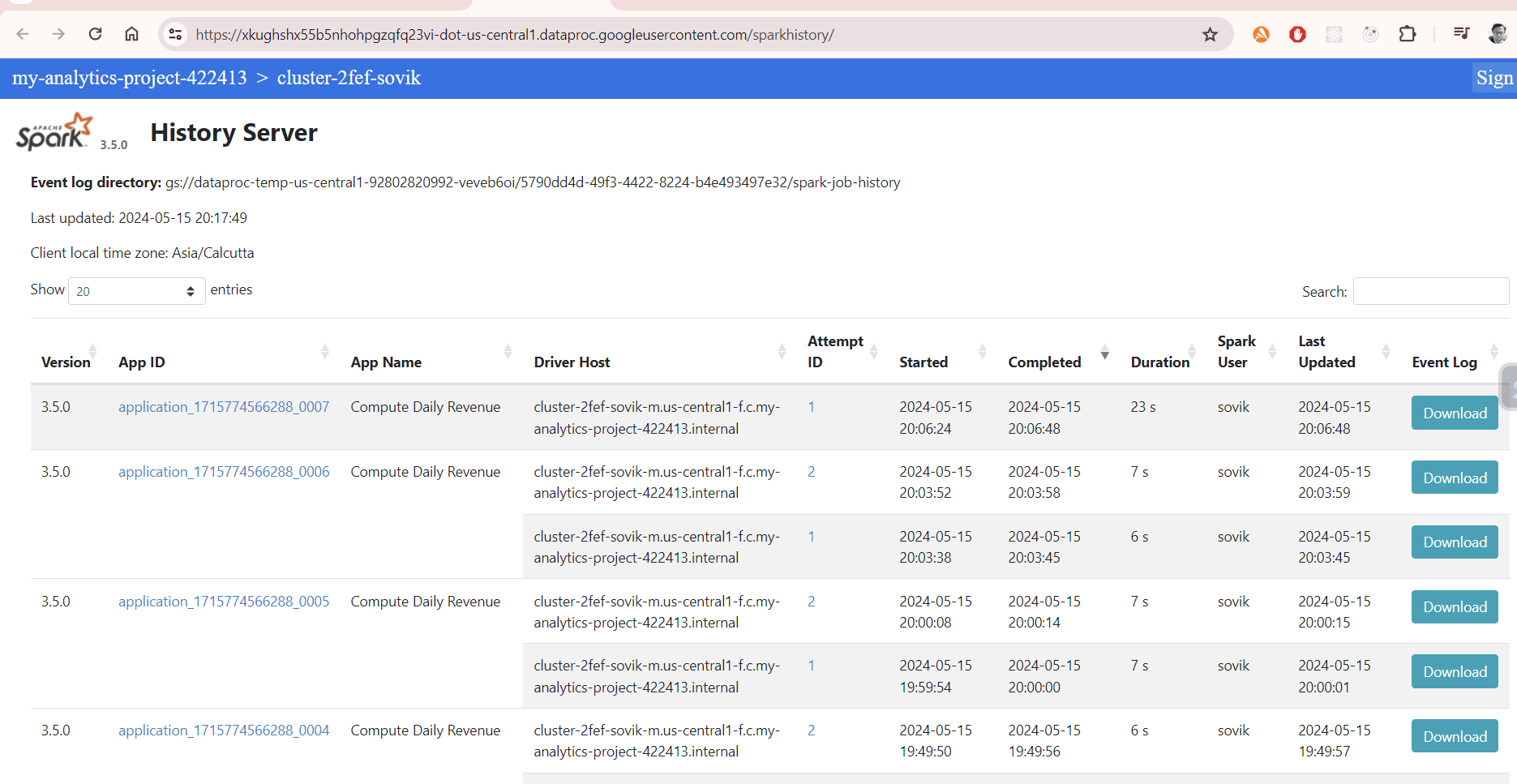


See from console for the last cluster run

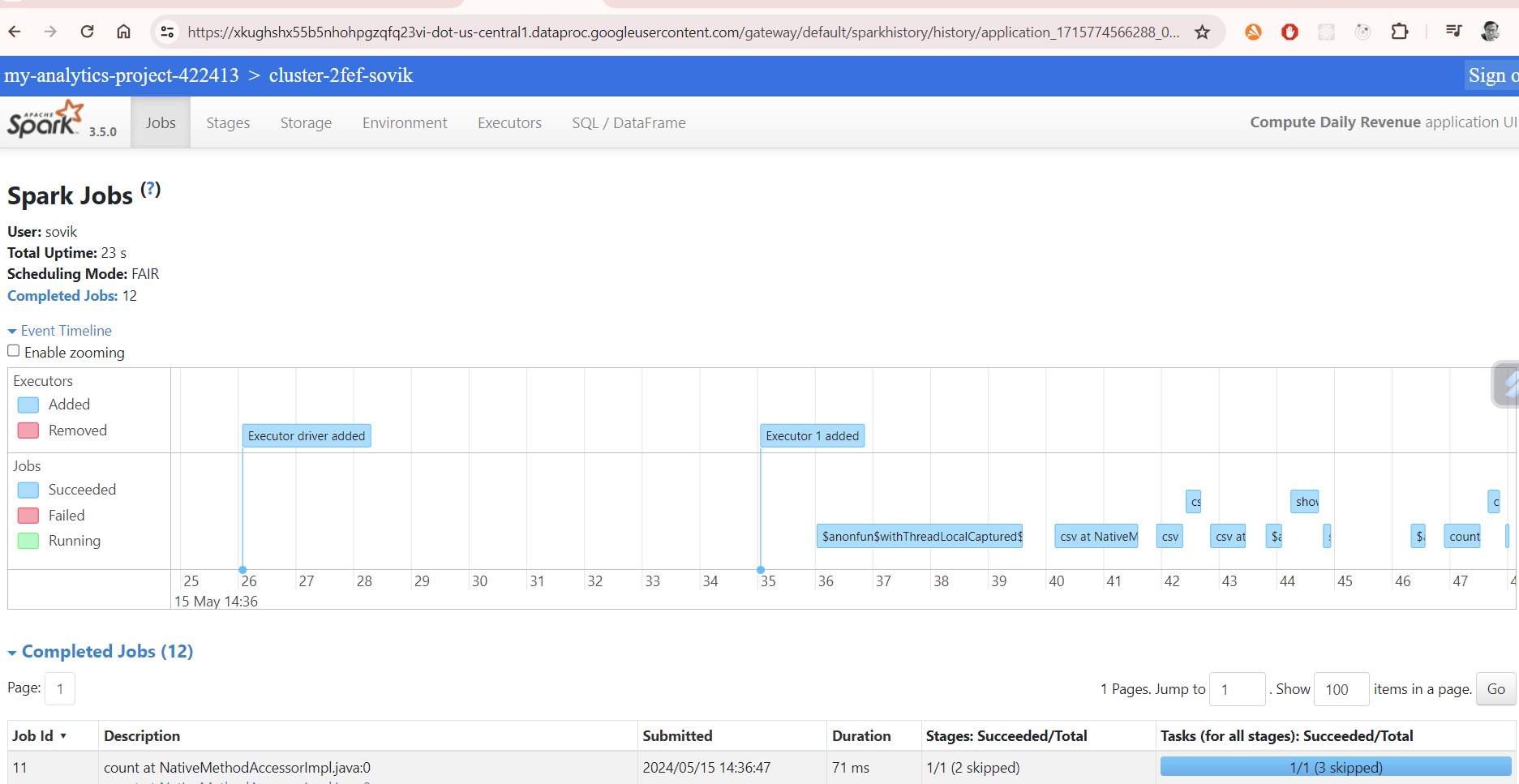


Spark history server:

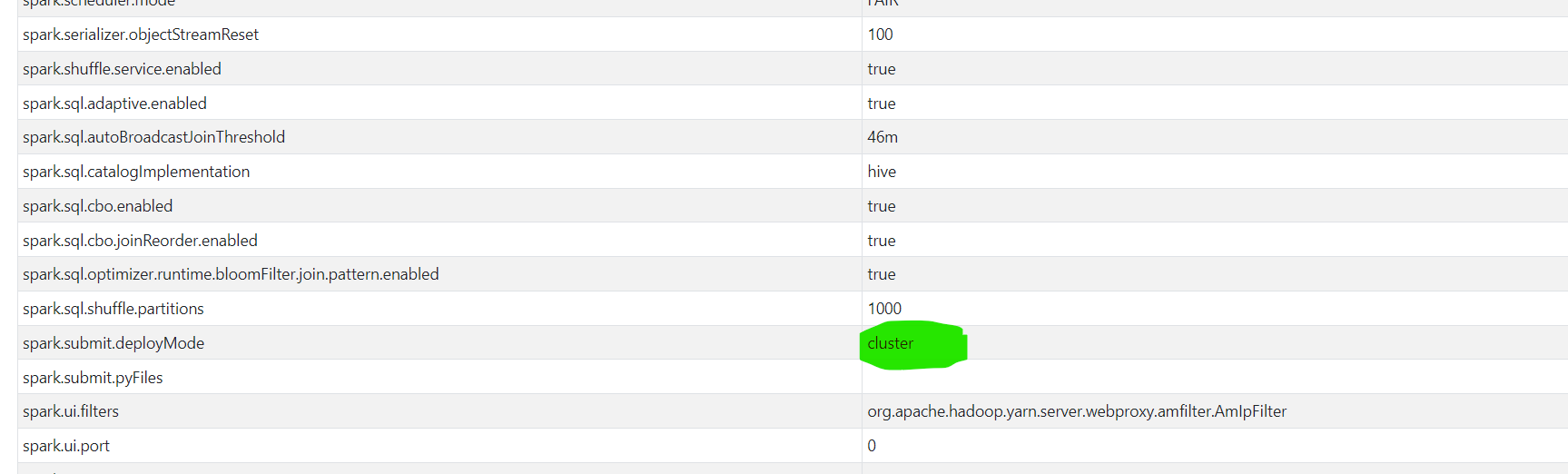




Select a latest job:

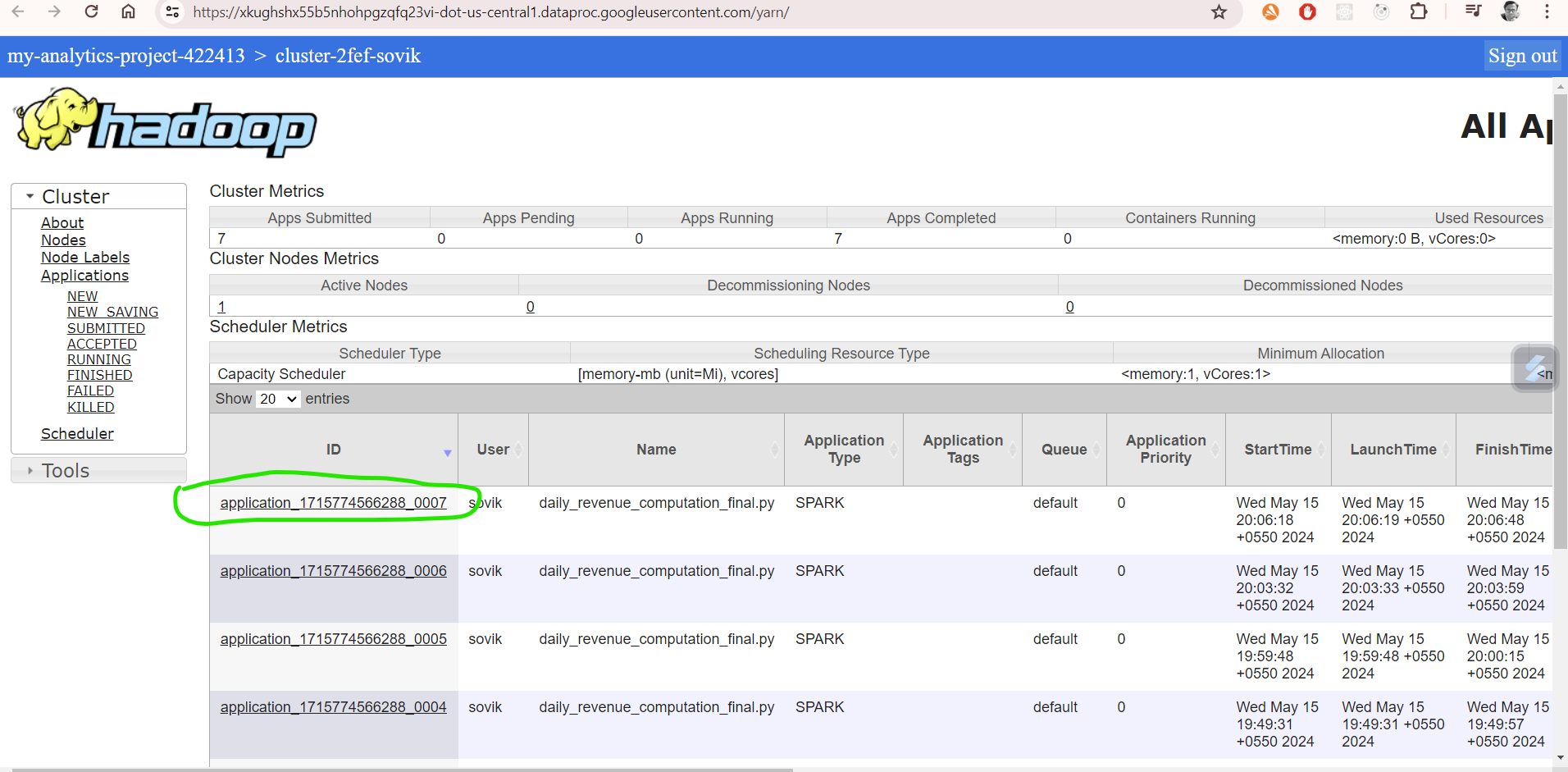


On Properties:deploy mode:



YARN logs:

From Cluster Details-> Web interfaces->



Attempt:



Logs for an app run attempt:



Lets Digress the code which is executed in the Driver and which is executed in the Executers:

Driver till SparkSession Object is created:

Eg:

*import* os

*from* pyspark.sql *import* SparkSession

*from* pyspark.sql.functions *import* sum, round

*# initialize the environmnet variables*

*#SRC\_BASE\_DIR=/user/sovik/retail\_db*

src\_base\_dir=os.environ.get('SRC\_BASE\_DIR')

tgt\_base\_dir=os.environ.get('TGT\_BASE\_DIR')

*# Initialize Spark Session*

spark = SparkSession. \

    builder. \

    appName('Compute Daily Revenue'). \

    master('yarn'). \

    getOrCreate()

Executer:

*# Initialize Spark Session*

spark = SparkSession. \

    builder. \

    appName('Compute Daily Revenue'). \

    master('yarn'). \

    getOrCreate()

*# Read orders data frame*

orders = spark. \

    read. \

    csv(f"{src\_base\_dir}/orders",

*schema*='''

        order\_id INT, order\_date STRING, order\_customer\_id INT, order\_status STRING

        ''',

*header*=False

    )

*# Read order-items data frame*

order\_items = spark. \

    read. \

    csv(f"{src\_base\_dir}/order\_items",

*schema*='''

        order\_item\_id INT,

        order\_item\_order\_id INT,

        order\_item\_product\_id INT,

        order\_item\_quantity INT,

        order\_item\_subtotal FLOAT,

        order\_item\_product\_price FLOAT

        ''',

*header*=False

    )

*# Filter, join, group and aggregate to get daily revenue*

orders\_daily = orders.filter("order\_status IN ('COMPLETE', 'CLOSED')"). \

    join(order\_items, orders['order\_id'] == order\_items['order\_item\_order\_id']). \

    groupBy('order\_date'). \

    agg(round(sum('order\_item\_subtotal'), 2).alias('revenue')). \

    orderBy('order\_date')

*# Write the result to CSV*

orders\_daily. \

    write. \

    mode('overwrite'). \

    csv(f'{tgt\_base\_dir}/daily\_revenue')

*# Validate - Display and count rows in the resultant data frame*

orders\_daily.show()

print("Total Rows:", orders\_daily.count())

Difference between cluster mode and client mode:

In client mode the The Driver will run on Client Gateway but the business logic will run on executer:

In cluster mode the driver and the business logic will run on executer.

***For issue with cluster if there is any issue the Issue can be seen in Yarn cluster timeline not Yarn Node Manager***

## Spark submit with Delta partitioned by ordermonth

Change in code for delta partitioning

orders\_daily. \

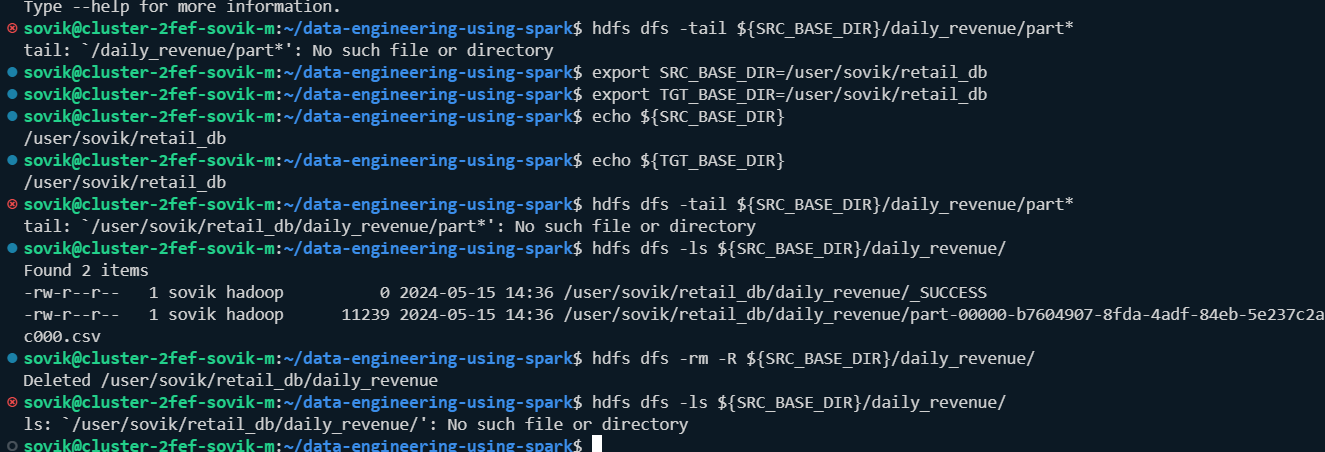
    write. \

    mode('overwrite'). \

    format('delta'). \

    save(f'{tgt\_base\_dir}/daily\_revenue',*header*=True)

cleanup:



**Command:**

spark-submit \

--deploy-mode cluster \

--packages io.delta:delta-spark\_2.12:3.0.0 \

--conf spark.yarn.appMasterEnv.SRC\_BASE\_DIR=/user/sovik/retail\_db \

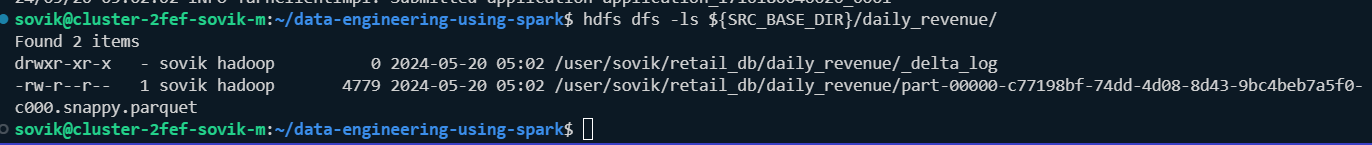
--conf spark.yarn.appMasterEnv.TGT\_BASE\_DIR=/user/sovik/retail\_db \

--conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension \

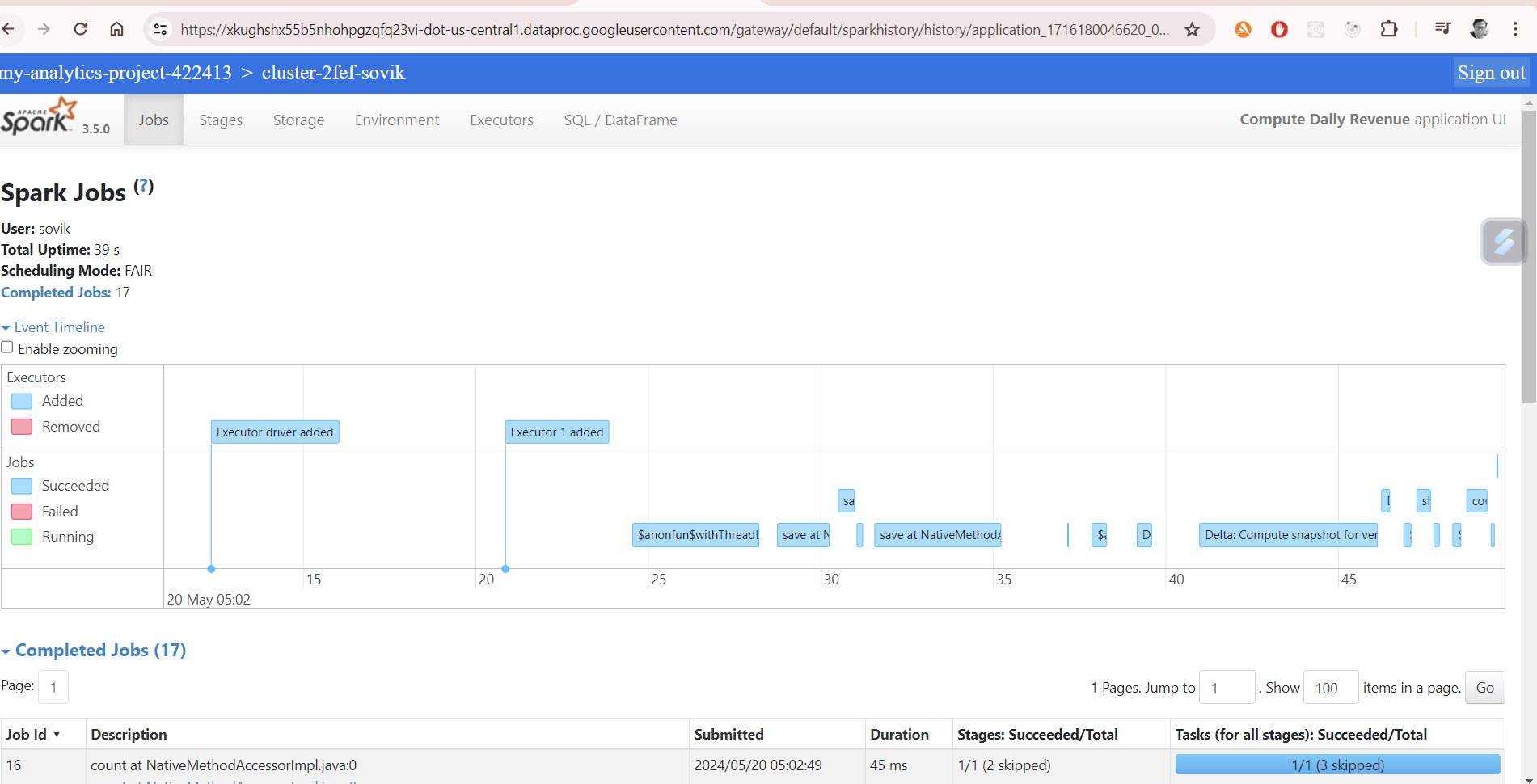
--conf spark.sql.catalog.spark\_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \

/home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py

Validate:



Check the app log in spark in dataproc cluster: Webinstance->YARN Resource Manager



Validate via code:

pyspark --packages io.delta:delta-spark\_2.12:3.0.0 \

> --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension \

> --conf spark.sql.catalog.spark\_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog

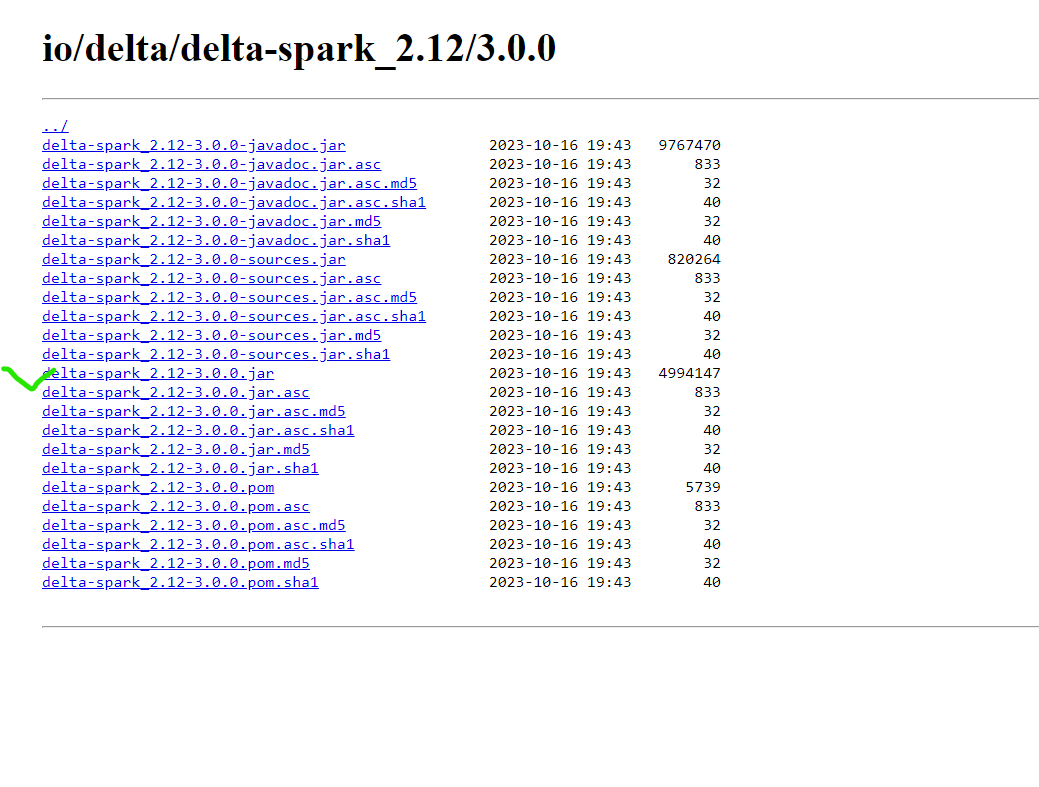
o/P:



**Lets download the jar files rather that calling it independently all the time:**

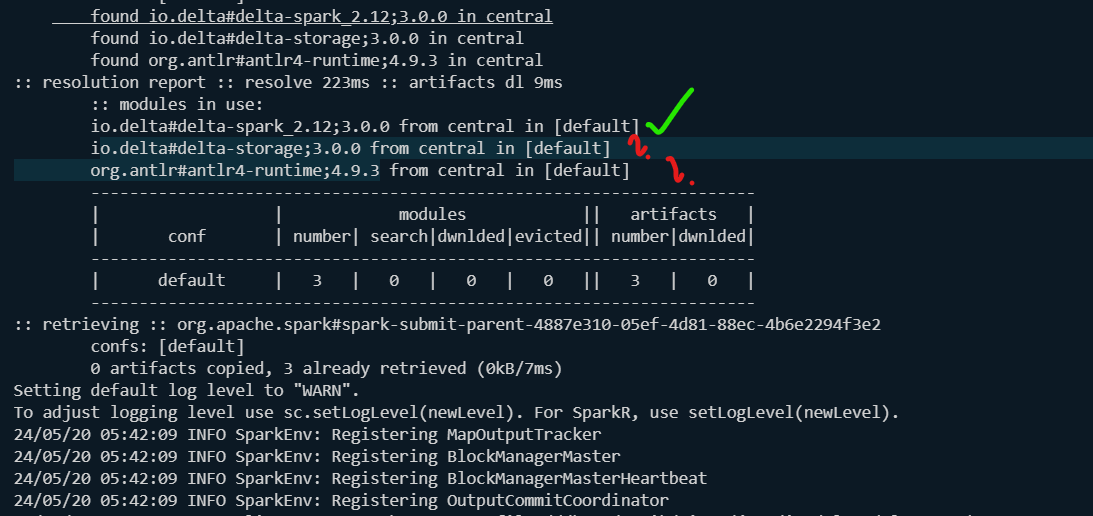
io.delta:delta-spark\_2.12:3.0.0

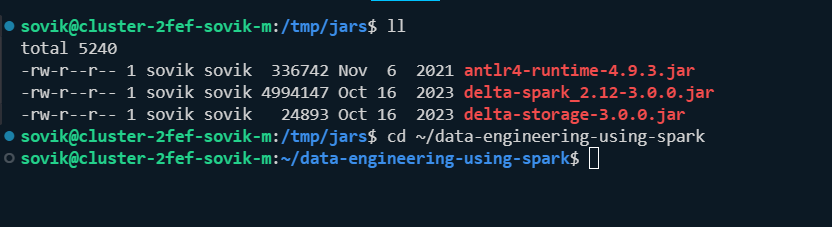
Copy the jar link: <https://github.com/delta-io/delta/releases>





Along with the above we need both the question marked jars as well:





Command run:

spark-submit --deploy-mode cluster \

--conf spark.yarn.appMasterEnv.SRC\_BASE\_DIR=/user/sovik/retail\_db \

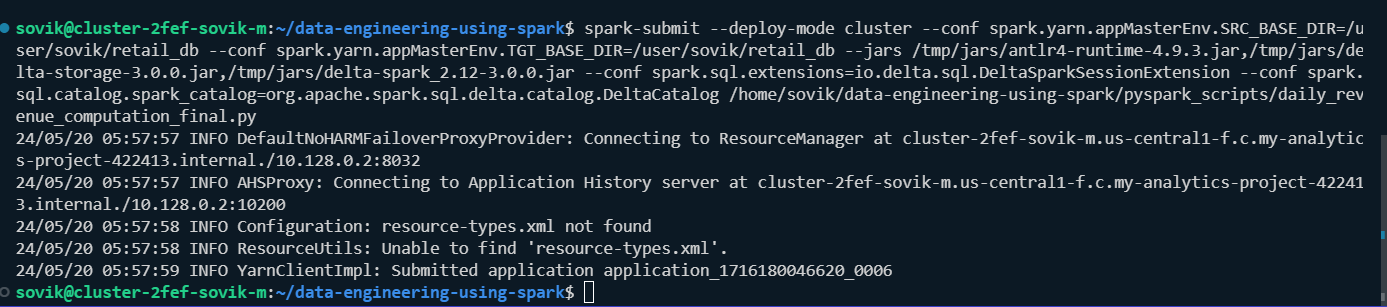
--conf spark.yarn.appMasterEnv.TGT\_BASE\_DIR=/user/sovik/retail\_db \

--jars /tmp/jars/antlr4-runtime-4.9.3.jar,/tmp/jars/delta-storage-3.0.0.jar,/tmp/jars/delta-spark\_2.12-3.0.0.jar \

--conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension \

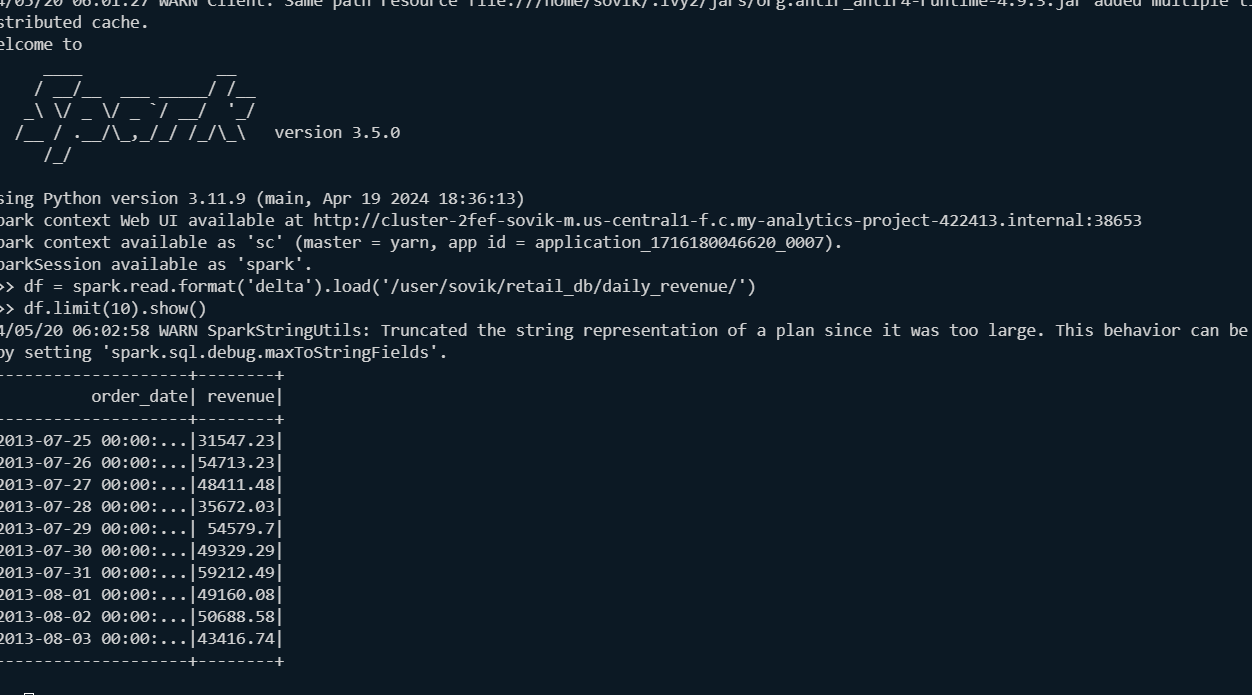
--conf spark.sql.catalog.spark\_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \

/home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py



Validate:

*Same as previous validation:*



Create a shell wrapper with command as below->give permission chmod 777 -run:

spark-submit --deploy-mode cluster \

--conf spark.yarn.appMasterEnv.SRC\_BASE\_DIR=/user/sovik/retail\_db \

--conf spark.yarn.appMasterEnv.TGT\_BASE\_DIR=/user/sovik/retail\_db \

--jars /tmp/jars/antlr4-runtime-4.9.3.jar,/tmp/jars/delta-storage-3.0.0.jar,/tmp/jars/delta-spark\_2.12-3.0.0.jar \

--conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension \

--conf spark.sql.catalog.spark\_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog \

/home/sovik/data-engineering-using-spark/pyspark\_scripts/daily\_revenue\_computation\_final.py