Tasks to perform:

Week 1: Overview and basic configurations

Step 1: Choose a suitable cloud provider and set up a Spark shell environment

Step 2: Configure the necessary dependencies

Step 3: Execute basic Spark commands to make sure Spark is ready

Step 4: Use README.md for details, instructions, and commands

Week 2: Data ingestion

Step 1: Upload the entire data into Hive from CSV using cloud provider cluster setup (such as, EMR)

1. Log in to PuTTY with the username “hadoop”
2. Enter the command given below:

Command: hive

3. Create a database
4. Create a table with all the relevant details

Step 2: Create a bucket (for example: S3 and Azure Blob) and upload the csv file

Step 3: Load the data from the bucket into the Hive table

Step 4: Create a new directory in HDFS and copy the data from Hive into HDFS

Step 5: Check if the data has been successfully loaded in the HDFS path

Week 3: Data streaming

Step 1: Connect to Spark shell with all the dependencies (Hive, Hadoop,and HDFS).

1. Create Schema of the CSV files
2. Create a Spark session

- Add Object Storage Service details as per the Cloud provider
- Add all variables to your environment as they contain sensitive data

Step 2: Read the CSV file and convert the file to a data frame

Step 3: Convert “order_purchase_timestamp” to week and day using UDF

Step 4: Calculate the following data:

1. Total sales and order distribution per day and week for each city
2. Total sales and order distribution per day and week for each state
3. Average review score, average freight value, average order approval, and delivery time
4. The freight charges per city and total freight charges

Home > Create a resource > Marketplace > Storage account >

Create a storage account ...

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * Simplilearn SS - 013

Resource group * databricks-1393430

Create new

Instance details

Storage account name * retailanalytics

Region * (US) West US 2

Deploy to an Azure Extended Zone

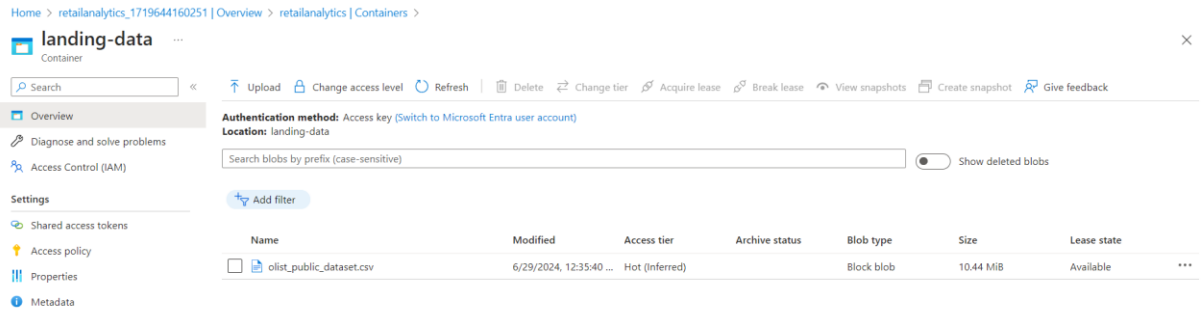
Performance * (i)

- (●) **Standard:** Recommended for most scenarios (general-purpose v2 account)
- ( ) **Premium:** Recommended for scenarios that require low latency.

Redundancy * Geo-redundant storage (GRS)

[x] Make read access to data available in the event of regional unavailability.

Previous | Next | Review + create



Home > Create a resource > Marketplace > Storage account >

Create a storage account ...

Basics | Advanced | Networking | Data protection | Encryption | Tags | **Review + create**

View automation template

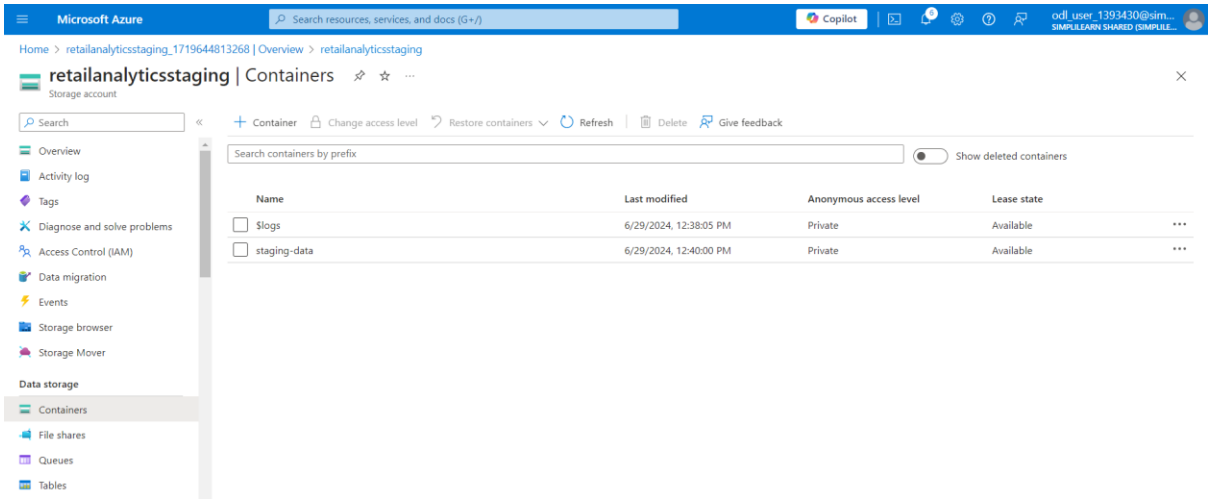
Basics

Subscription	Simplilearn SS - 013
Resource group	databricks-1393430
Location	West US 2
Storage account name	retailanalyticsstaging
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

Enable hierarchical namespace	Disabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

Previous | Next | Create



Home >

Create Data Factory ...

Basics | Git configuration | Networking | Advanced | Tags | Review + create

One-click to create data factory with sample pipeline and datasets. Try it

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Simplilearn SS - 013

Resource group * : databricks-1393430

Create new

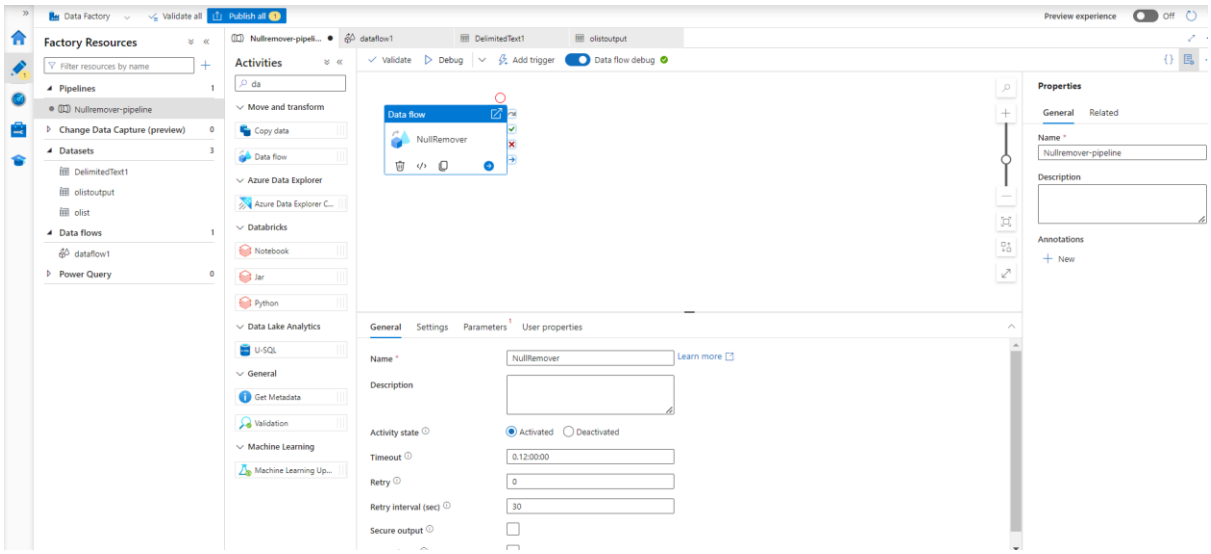
Instance details

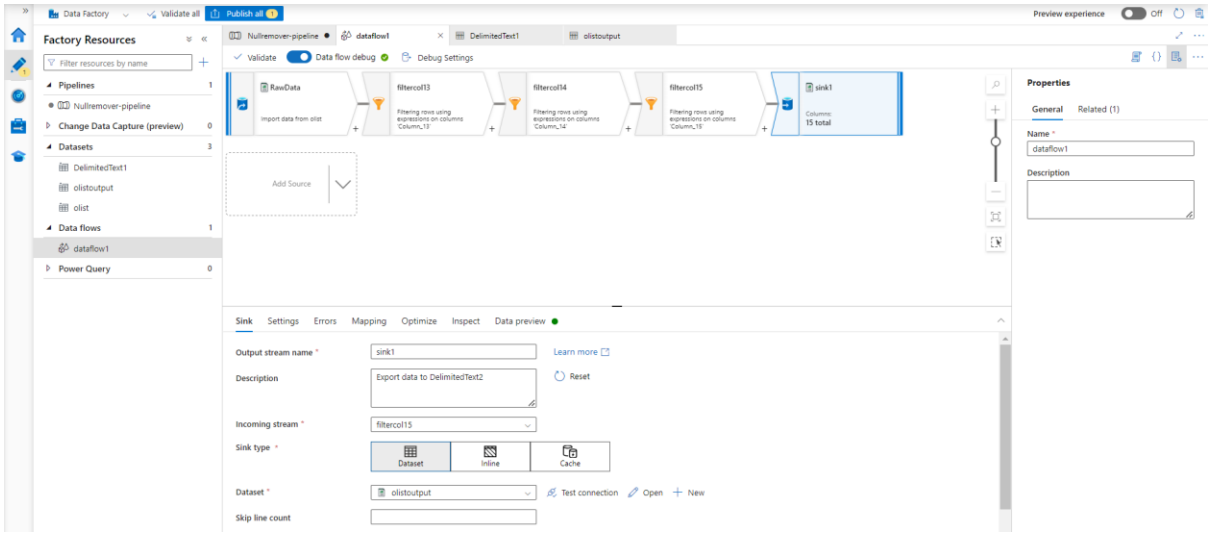
Name * : samir-datafactory

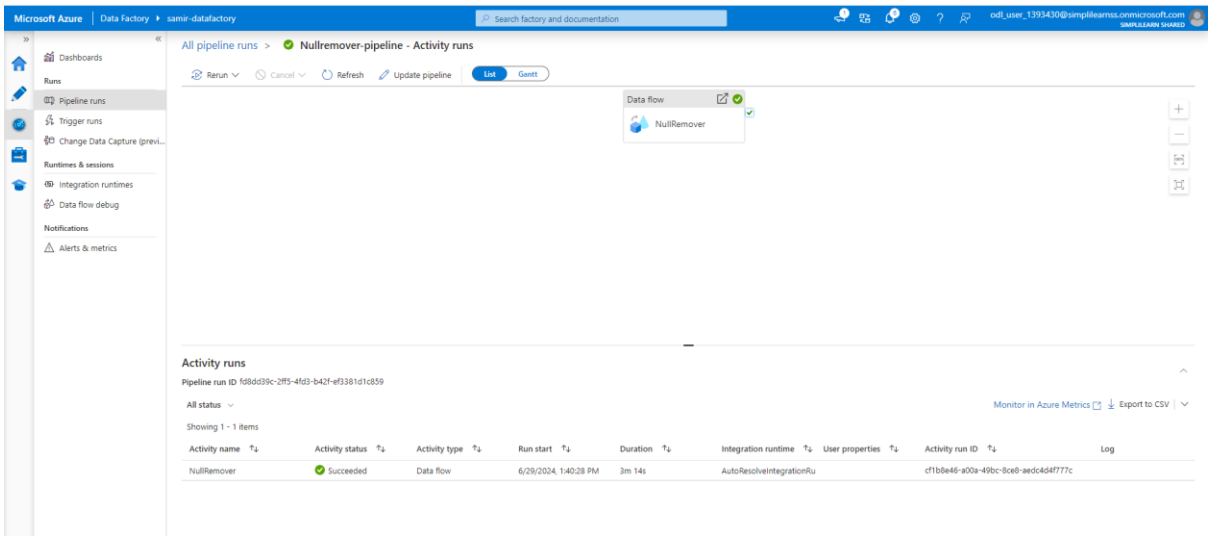
Region * : East US

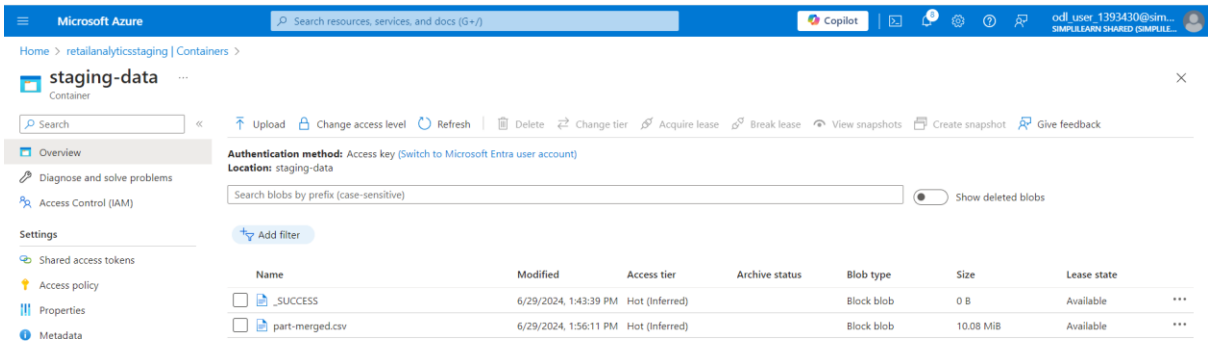
Version * : V2

Previous | Next | Review + create




Data Factory
Validate all
Publish all
Preview experience
Off
Factory Resources
Filter resources by name
Pipelines
Nullremover-pipeline
Change Data Capture (preview)
Datasets
DelimitedText1
olistoutput
olist
Data flows
dataflow1
Power Query
Nullremover-pipeline
dataflow1
DelimitedText1
olistoutput
Validate
Data flow debug
Debug Settings
RawData
Import data from olist
filtercol13
filtercol14
filtercol15
sink1
15 total
Add Source
Properties
General
Related (1)
Name *
dataflow1
Description
Sink
Settings
Errors
Mapping
Optimize
Inspect
Data preview
Output stream name *
sink1
Learn more
Description
Export data to DelimitedText2
Reset
Incoming stream *
filtercol15
Sink type *
Dataset
Inline
Cache
Dataset *
olistoutput
Test connection
Open
New
Skip line count


Microsoft Azure
Data Factory
samir-datafactory
Search factory and documentation
odl_user_1393430@simplilearnss.onmicrosoft.com
SIMPLILEARN SHARED
Dashboards
Runs
Pipeline runs
Trigger runs
Change Data Capture (previ...
Runtimes & sessions
Integration runtimes
Data flow debug
Notifications
Alerts & metrics
All pipeline runs
Nullremover-pipeline - Activity runs
Rerun
Cancel
Refresh
Update pipeline
List
Gantt
Data flow
NullRemover
Activity runs
Pipeline run ID fd8dd39c-2ff5-4fd3-b42f-ef3381d1c859
All status
Monitor in Azure Metrics
Export to CSV
Showing 1 - 1 items
Activity name
Activity status
Activity type
Run start
Duration
Integration runtime
User properties
Activity run ID
Log
NullRemover
Succeeded
Data flow
6/29/2024, 1:40:28 PM
3m 14s
AutoResolveIntegrationRu
cf1b8e46-a00a-49bc-8ce8-aedc4d4f777c


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
odl_user_1393430@sim...
SIMPLILEARN SHARED (SIMPLILE...
Home
retailanalyticsstaging | Containers
staging-data
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Access policy
Properties
Metadata
Upload
Change access level
Refresh
Delete
Change tier
Acquire lease
Break lease
View snapshots
Create snapshot
Give feedback
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: staging-data
Search blobs by prefix (case-sensitive)
Show deleted blobs
Add filter
Name
Modified
Access tier
Archive status
Blob type
Size
Lease state
_SUCCESS
6/29/2024, 1:43:39 PM
Hot (Inferred)
Block blob
0 B
Available
part-merged.csv
6/29/2024, 1:56:11 PM
Hot (Inferred)
Block blob
10.08 MiB
Available

Home > Azure Databricks >

Create an Azure Databricks workspace ...

Basics | Networking | Encryption | Security & compliance | Tags | Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Simplilearn SS - 013

Resource group * : databricks-1393430

Create new

Instance Details

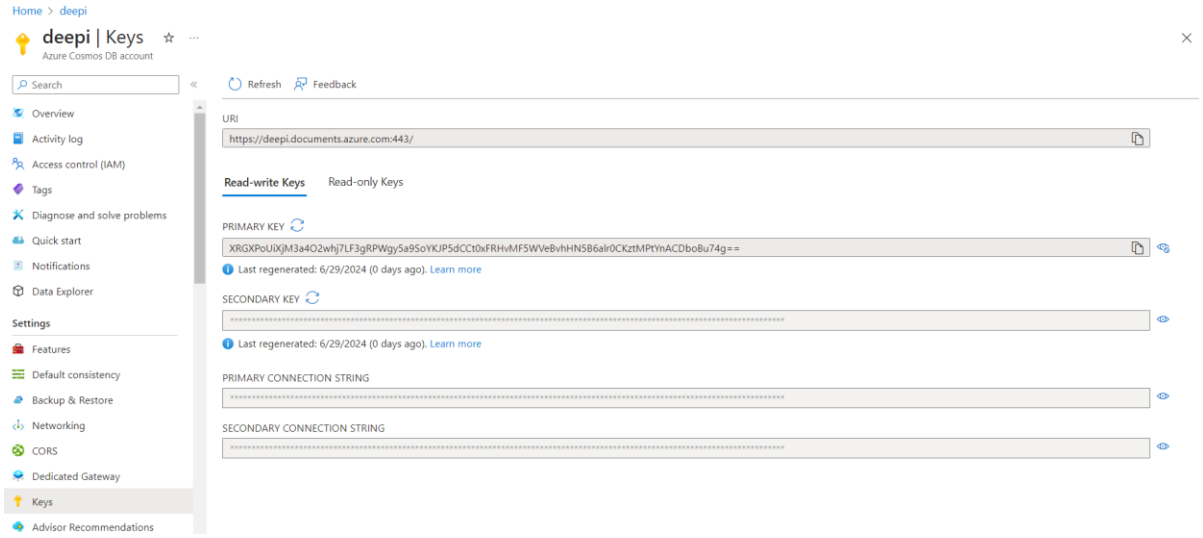
Workspace name * : databricksdeepi

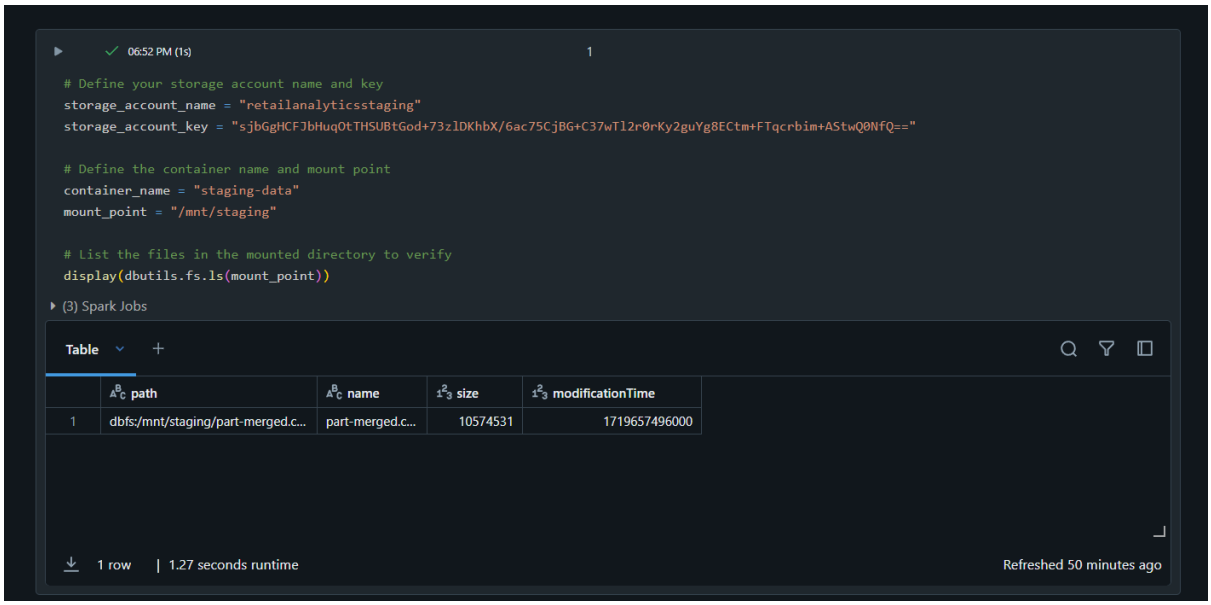
Region * : West US 2

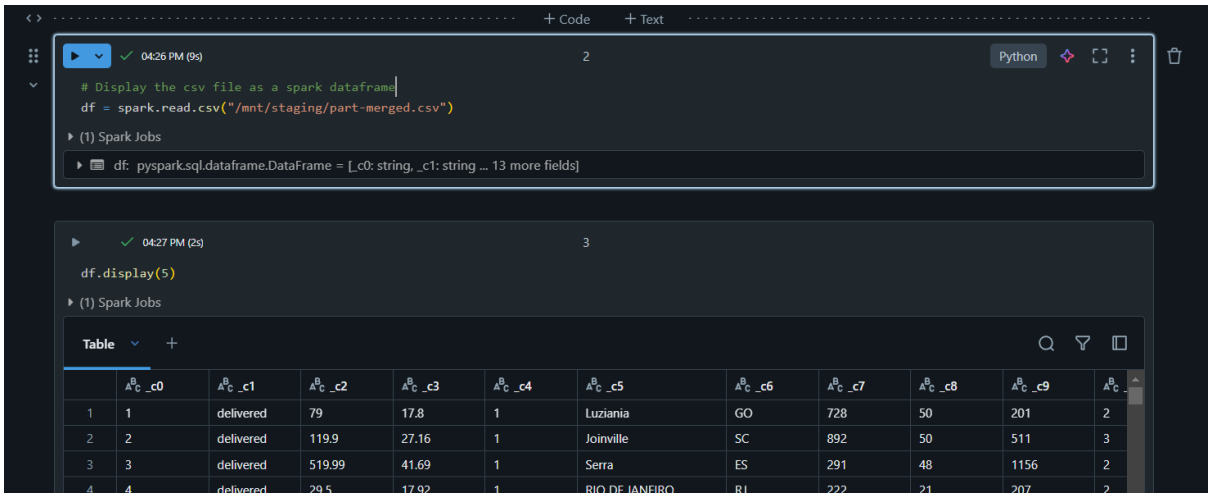
Pricing Tier * : Standard (Apache Spark, Secure with Microsoft Entra ID)

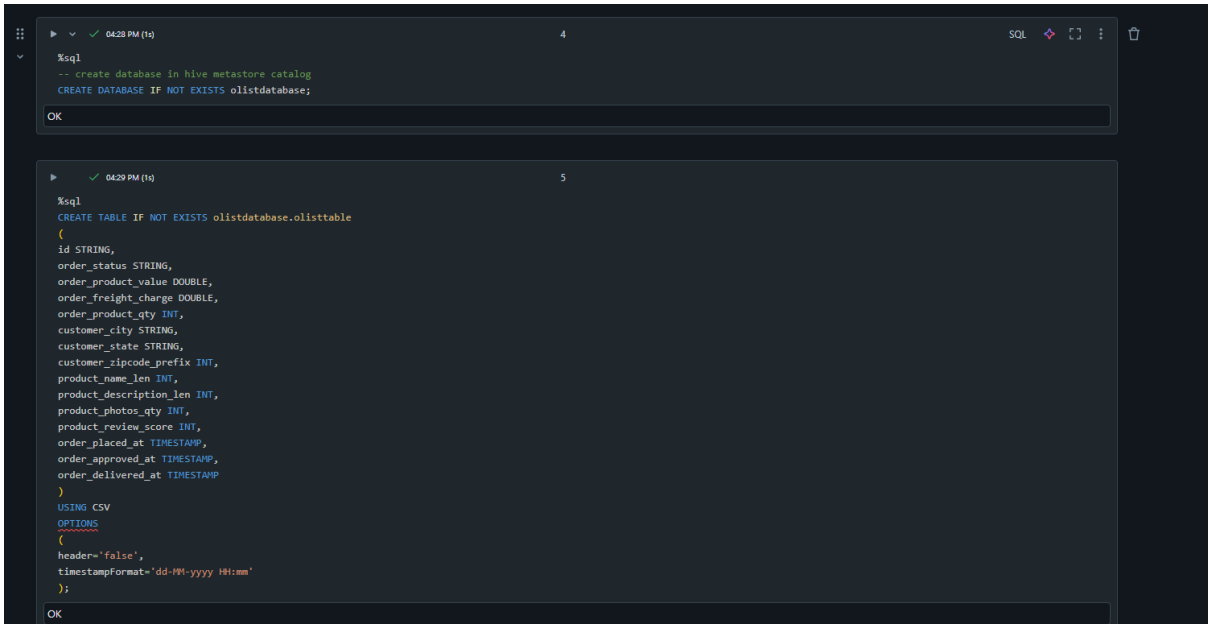
Managed Resource Group name : Enter name for managed resource group

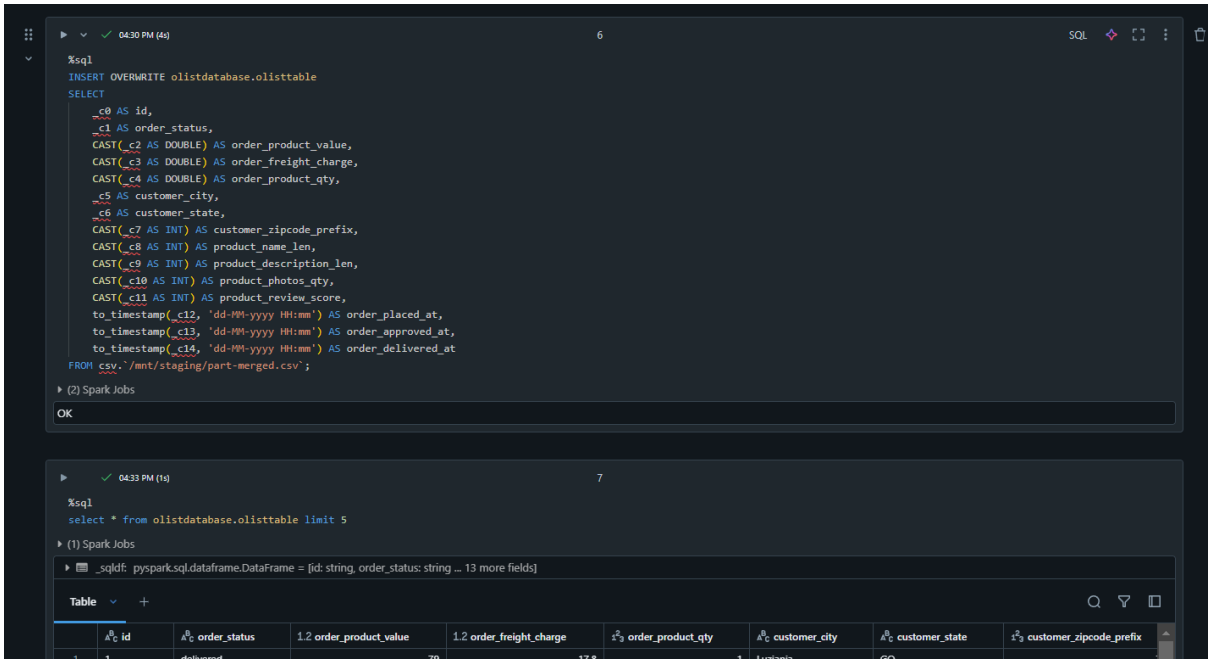
Review + create | < Previous | Next : Networking >

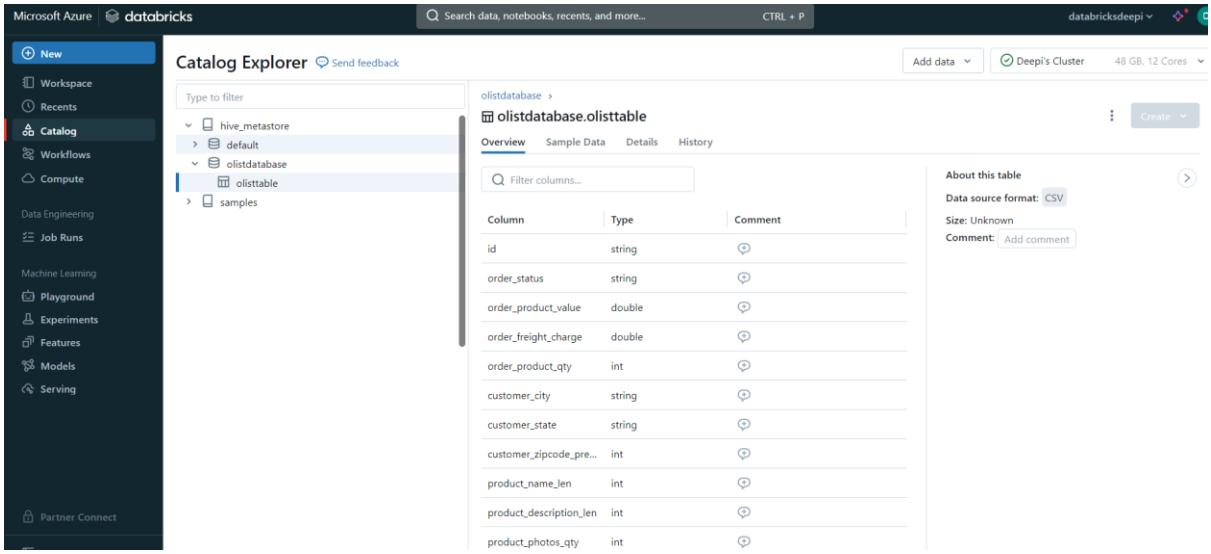

Home > deepi
deepi | Keys
Azure Cosmos DB account
Search
Refresh
Feedback
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Quick start
Notifications
Data Explorer
Settings
Features
Default consistency
Backup & Restore
Networking
CORS
Dedicated Gateway
Keys
Advisor Recommendations
URI
https://deepi.documents.azure.com:443/
Read-write Keys
Read-only Keys
PRIMARY KEY
XRGXPoUiXjM3a4O2whj7LF3gRPWgy5a9SoYKJP5dCCt0xFRHvMF5WVeBvhHN5B6alr0CKztMPtYnACDboBu74g==
Last regenerated: 6/29/2024 (0 days ago). Learn more
SECONDARY KEY
Last regenerated: 6/29/2024 (0 days ago). Learn more
PRIMARY CONNECTION STRING
SECONDARY CONNECTION STRING

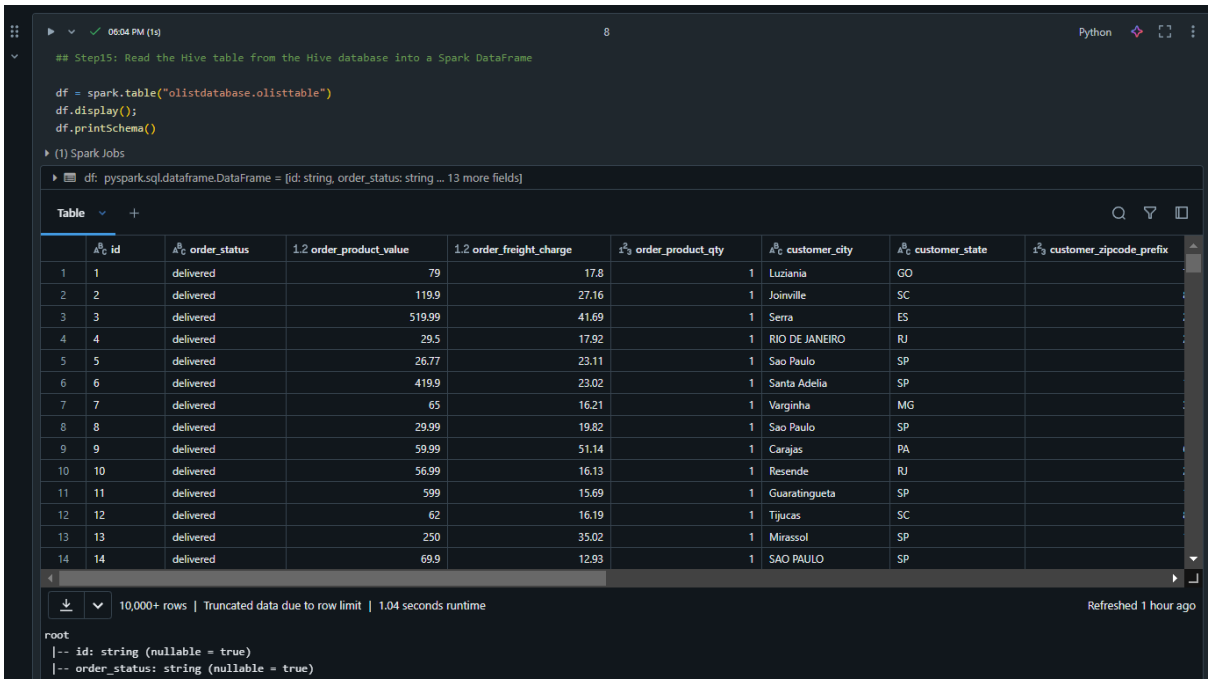

06:52 PM (1s)
1
# Define your storage account name and key
storage_account_name = "retailanalyticsstaging"
storage_account_key = "sjbGgHCFJbHuqOtTHSUBtGod+73zlDKhbX/6ac75CjBG+C37wTl2r0rKy2guYg8ECtm+FTqcrbim+AStwQ0NfQ=="
# Define the container name and mount point
container_name = "staging-data"
mount_point = "/mnt/staging"
# List the files in the mounted directory to verify
display(dbutils.fs.ls(mount_point))
(3) Spark Jobs
Table
path
name
size
modificationTime
1
dbfs:/mnt/staging/part-merged.c...
part-merged.c...
10574531
1719657496000
1 row | 1.27 seconds runtime
Refreshed 50 minutes ago


+ Code
+ Text
04:26 PM (9s)
2
Python
# Display the csv file as a spark dataframe
df = spark.read.csv("/mnt/staging/part-merged.csv")
(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 13 more fields]
04:27 PM (2s)
3
df.display(5)
(1) Spark Jobs
Table
_c0
_c1
_c2
_c3
_c4
_c5
_c6
_c7
_c8
_c9
1
1
delivered
79
17.8
1
Luziania
GO
728
50
201
2
2
2
delivered
119.9
27.16
1
Joinville
SC
892
50
511
3
3
3
delivered
519.99
41.69
1
Serra
ES
291
48
1156
2
4
4
delivered
29.5
17.92
1
RIO DE JANEIRO
RJ
222
21
207
2


04:28 PM (1s)
4
SQL
%sql
-- create database in hive metastore catalog
CREATE DATABASE IF NOT EXISTS olistdatabase;
OK
04:29 PM (1s)
5
%sql
CREATE TABLE IF NOT EXISTS olistdatabase.olisttable
(
id STRING,
order_status STRING,
order_product_value DOUBLE,
order_freight_charge DOUBLE,
order_product_qty INT,
customer_city STRING,
customer_state STRING,
customer_zipcode_prefix INT,
product_name_len INT,
product_description_len INT,
product_photos_qty INT,
product_review_score INT,
order_placed_at TIMESTAMP,
order_approved_at TIMESTAMP,
order_delivered_at TIMESTAMP
)
USING CSV
OPTIONS
(
header='false',
timestampFormat='dd-MM-yyyy HH:mm'
);
OK


04:30 PM (4s)
6
SQL
%sql
INSERT OVERWRITE olistdatabase.olisttable
SELECT
_c0 AS id,
_c1 AS order_status,
CAST(_c2 AS DOUBLE) AS order_product_value,
CAST(_c3 AS DOUBLE) AS order_freight_charge,
CAST(_c4 AS DOUBLE) AS order_product_qty,
_c5 AS customer_city,
_c6 AS customer_state,
CAST(_c7 AS INT) AS customer_zipcode_prefix,
CAST(_c8 AS INT) AS product_name_len,
CAST(_c9 AS INT) AS product_description_len,
CAST(_c10 AS INT) AS product_photos_qty,
CAST(_c11 AS INT) AS product_review_score,
to_timestamp(_c12, 'dd-MM-yyyy HH:mm') AS order_placed_at,
to_timestamp(_c13, 'dd-MM-yyyy HH:mm') AS order_approved_at,
to_timestamp(_c14, 'dd-MM-yyyy HH:mm') AS order_delivered_at
FROM csv.`/mnt/staging/part-merged.csv`;
(2) Spark Jobs
OK
04:33 PM (1s)
7
%sql
select * from olistdatabase.olisttable limit 5
(1) Spark Jobs
_sqldf: pyspark.sql.dataframe.DataFrame = [id: string, order_status: string ... 13 more fields]
Table
id
order_status
order_product_value
order_freight_charge
order_product_qty
customer_city
customer_state
customer_zipcode_prefix


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
databricksdeepi
New
Workspace
Recents
Catalog
Workflows
Compute
Data Engineering
Job Runs
Machine Learning
Playground
Experiments
Features
Models
Serving
Partner Connect
Catalog Explorer
Send feedback
Add data
Deepi's Cluster
48 GB, 12 Cores
Type to filter
hive_metastore
default
olistdatabase
olisttable
samples
olistdatabase
olistdatabase.olisttable
Create
Overview
Sample Data
Details
History
Filter columns...
Column
Type
Comment
id
string
order_status
string
order_product_value
double
order_freight_charge
double
order_product_qty
int
customer_city
string
customer_state
string
customer_zipcode_pre...
int
product_name_len
int
product_description_len
int
product_photos_qty
int
About this table
Data source format: CSV
Size: Unknown
Comment: Add comment


06:04 PM (1s)
8
Python
Step15: Read the Hive table from the Hive database into a Spark DataFrame
df = spark.table("olistdatabase.olisttable")
df.display();
df.printSchema()
(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [id: string, order_status: string ... 13 more fields]
Table
id
order_status
order_product_value
order_freight_charge
order_product_qty
customer_city
customer_state
customer_zipcode_prefix
1 1 delivered 79 17.8 1 Luziania GO
2 2 delivered 119.9 27.16 1 Joinville SC
3 3 delivered 519.99 41.69 1 Serra ES
4 4 delivered 29.5 17.92 1 RIO DE JANEIRO RJ
5 5 delivered 26.77 23.11 1 Sao Paulo SP
6 6 delivered 419.9 23.02 1 Santa Adelia SP
7 7 delivered 65 16.21 1 Varginha MG
8 8 delivered 29.99 19.82 1 Sao Paulo SP
9 9 delivered 59.99 51.14 1 Carajas PA
10 10 delivered 56.99 16.13 1 Resende RJ
11 11 delivered 599 15.69 1 Guaratingueta SP
12 12 delivered 62 16.19 1 Tijucas SC
13 13 delivered 250 35.02 1 Mirassol SP
14 14 delivered 69.9 12.93 1 SAO PAULO SP
10,000+ rows | Truncated data due to row limit | 1.04 seconds runtime
Refreshed 1 hour ago
root
|-- id: string (nullable = true)
|-- order_status: string (nullable = true)

06:04 PM (<1s) 9 Python

```
from pyspark.sql import functions as F
from pyspark.sql.functions import col
from functools import reduce
```

06:04 PM (1s) 10

```
### Step16: Calculate the order metrics

df = df.withColumn("order_sales", col("order_product_value") * col("order_product_qty"))
df = df.withColumn("order_approval_time", col("order_approved_at") - col("order_placed_at"))
df = df.withColumn("order_delivery_time", col("order_delivered_at") - col("order_placed_at"))
df = df.withColumn("order_placed_date", F.to_date(col("order_placed_at")))
df = df.withColumn("order_placed_weekdate", F.trunc(col("order_placed_at"), "week"))
df.display();
```

(1) Spark Jobs

df: pyspark.sql.dataframe.DataFrame = [id: string, order_status: string ... 18 more fields]

Table

	id	order_status	order_product_value	order_freight_charge	order_product_qty	customer_city	customer_state	customer_zipcode_prefix
1	1	delivered	79	17.8	1	Luziania	GO	
2	2	delivered	119.9	27.16	1	Joinville	SC	
3	3	delivered	519.99	41.69	1	Serra	ES	
4	4	delivered	29.5	17.92	1	RIO DE JANEIRO	RJ	
5	5	delivered	26.77	23.11	1	Sao Paulo	SP	
6	6	delivered	419.9	23.02	1	Santa Adelia	SP	
7	7	delivered	65	16.21	1	Varginha	MG	
8	8	delivered	29.99	19.82	1	Sao Paulo	SP	
9	9	delivered	59.99	51.14	1	Carajas	PA	
10	10	delivered	56.99	16.13	1	Resende	RJ	

06:04 PM (<1s) 11

```
# Step17:Compute the historical daily insights (We will need this values in further steps.)

grp_date = df.groupBy(col("order_placed_date"))
grp_city_date = df.groupBy(col("customer_city"), col("order_placed_date"))
grp_state_date = df.groupBy(col("customer_state"), col("order_placed_date"))
```

06:04 PM (<1s) 12

```
## Step18:

## Total sales
grp_date_total_sales = grp_date.agg(F.sum("order_sales").alias("total_sales")).orderBy(col("order_placed_date").asc())

## Total sales per city
grp_city_date_total_sales = grp_city_date.agg(F.sum("order_sales").alias("total_sales")).orderBy(col("customer_city").asc(),
col("order_placed_date").asc())

##Total sales per state
grp_state_date_total_sales = grp_state_date.agg(F.sum("order_sales").alias("total_sales")).orderBy(col("customer_state").asc(),
col("order_placed_date").asc())
```

grp_date_total_sales: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, total_sales: double]
grp_city_date_total_sales: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_total_sales: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]

06:04 PM (<1s) 13

```
## Step19:

#Total Freight Charge
#i. Total freight charge
#ii. Total freight charge per city
#iii. Total freight charge per state
## Total freight charge
grp_date_total_freight_charge = grp_date.agg(F.sum("order_freight_charge").alias("total_freight_charge")).orderBy(col("order_placed_date").asc())

## Total freight charge per city.
grp_city_date_total_freight_charge = grp_city_date.agg(F.sum("order_freight_charge").alias("total_freight_charge")).orderBy(col("customer_city").asc(),col("order_placed_date").asc
())

## Total freight charge per state.
grp_state_date_total_freight_charge = grp_state_date.agg(F.sum("order_freight_charge").alias("total_freight_charge")).orderBy(col("customer_state").asc(),col("order_placed_date").
asc())
```

grp_date_total_freight_charge: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, total_freight_charge: double]
grp_city_date_total_freight_charge: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_total_freight_charge: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]

06:04 PM (<1s) 14

```
## Step20:

# Total Order Count
# i. Total order count
# ii. Total order count per city
# iii. Total order count per state

## Total order count
grp_date_total_order_count = grp_date.agg(F.count("id").alias("total_order_count")).orderBy(col("order_placed_date").asc())

## Total order count per city
grp_city_date_total_order_count = grp_city_date.agg(F.count("id").alias("total_order_count")).orderBy(col("customer_city").asc(),
col("order_placed_date").asc())

## Total order count per state
grp_state_date_total_order_count = grp_state_date.agg(F.count("id").alias("total_order_count")).orderBy(col("customer_state").asc(),
col("order_placed_date").asc())
```

```
grp_date_total_order_count: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, total_order_count: long]
grp_city_date_total_order_count: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_total_order_count: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]
```

06:04 PM (<1s) 15

```
# Step21: Average Freight charge

## Total average freight charge.
grp_date_avg_freight_charge = grp_date.agg(F.avg("order_freight_charge").alias("avg_freight_charge")).orderBy(col("order_placed_date").asc())

## Total order freight charge per city
grp_city_date_vg_freight_charge = grp_city_date.agg(F.avg("order_freight_charge").alias("avg_freight_charge")).orderBy(col("customer_city").asc(),col("order_placed_date").asc())

## Total order freight charge per state
grp_state_date_vg_freight_charge = grp_state_date.agg(F.avg("order_freight_charge").alias("avg_freight_charge")).orderBy(col("customer_state").asc(),col("order_placed_date").asc())
```

```
grp_date_avg_freight_charge: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, avg_freight_charge: double]
grp_city_date_vg_freight_charge: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_vg_freight_charge: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]
```

06:04 PM (<1s) 16

```
# Step22: Average Review score

# Total average review charge.
grp_date_avg_review_score = grp_date.agg(F.avg("product_review_score").alias("avg_review_score")).orderBy(col("order_placed_date")
.asc())

## Total order average review per city
grp_city_date_avg_review_score = grp_city_date.agg(F.avg("product_review_score").alias("avg_review_score")).orderBy(col("customer_city").asc(),col("order_placed_date").asc())

## Total order average review per state
grp_state_date_avg_review_score = grp_state_date.agg(F.avg("product_review_score").alias("avg_review_score")).orderBy(col("customer_state").asc(),col("order_placed_date").asc())
```

```
grp_date_avg_review_score: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, avg_review_score: double]
grp_city_date_avg_review_score: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_avg_review_score: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]
```

06:04 PM (<1s) 17

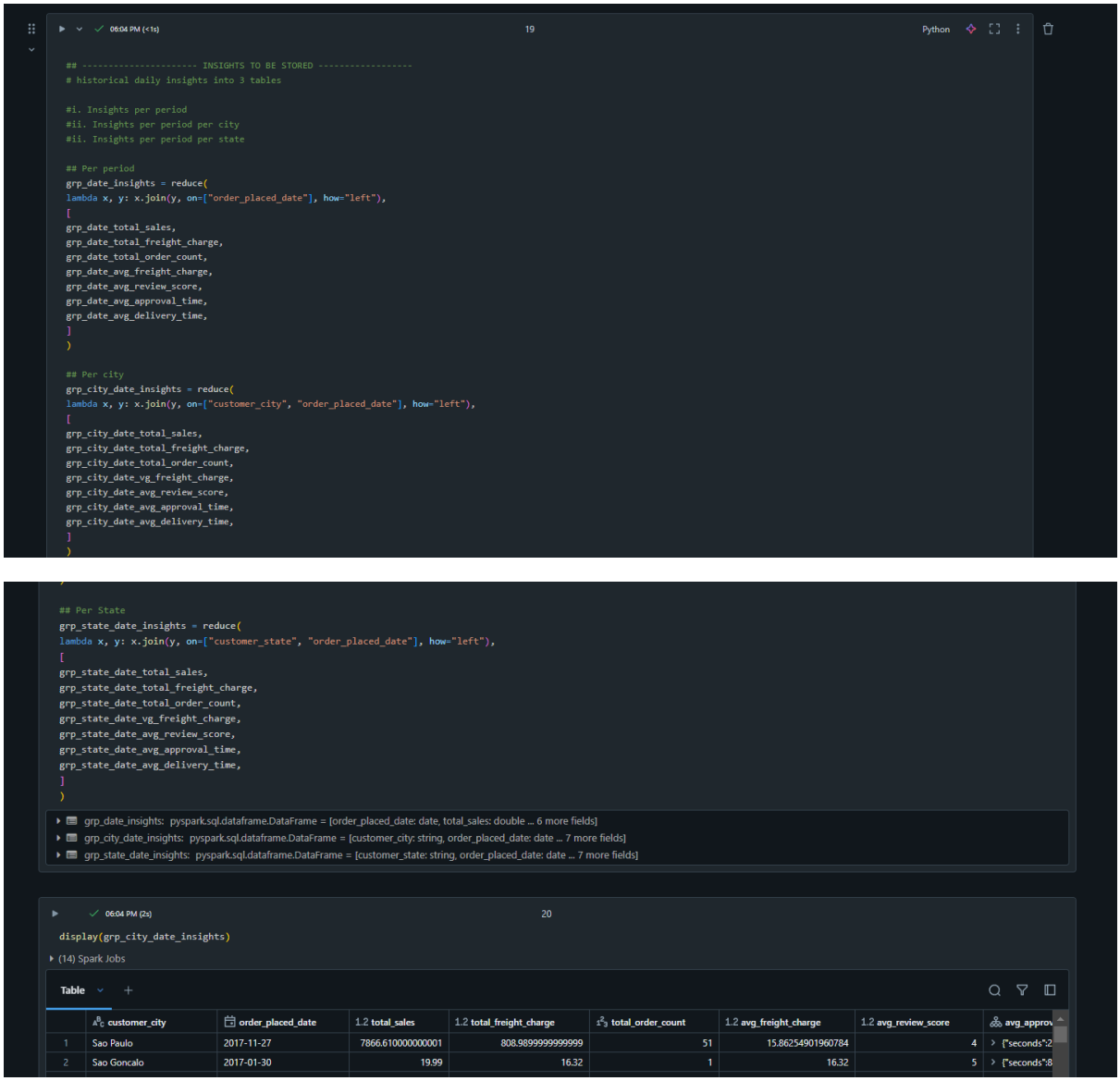
```
# Step23: Average Approval Time

grp_date_avg_approval_time = grp_date.agg(F.avg("order_approval_time").alias("avg_approval_time")).orderBy(col("order_placed_date")
.asc())

## Average Approval Time per city
grp_city_date_avg_approval_time = grp_city_date.agg(F.avg("order_approval_time").alias("avg_approval_time")).orderBy(col("customer_city").asc(),col("order_placed_date").asc())

## Average Approval Time per state
grp_state_date_avg_approval_time = grp_state_date.agg(F.avg("order_approval_time").alias("avg_approval_time")).orderBy(col("customer_state").asc(),col("order_placed_date").asc())
```

```
grp_date_avg_approval_time: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, avg_approval_time: interval day to second]
grp_city_date_avg_approval_time: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_avg_approval_time: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]
```

06:04 PM (<1s) 18

```
# Step24: Average Delivery Time

grp_date_avg_delivery_time = grp_date.agg(F.avg("order_delivery_time").alias("avg_delivery_time")).orderBy(col("order_placed_date").asc())

## Average Approval Time per city
grp_city_date_avg_delivery_time = grp_city_date.agg(F.avg("order_delivery_time").alias("avg_delivery_time")).orderBy(col("customer_city").asc(),col("order_placed_date").asc())

## Average Approval Time per state
grp_state_date_avg_delivery_time = grp_state_date.agg(F.avg("order_delivery_time").alias("avg_delivery_time")).orderBy(col("customer_state").asc(),col("order_placed_date").asc())
```

```
grp_date_avg_delivery_time: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, avg_delivery_time: interval day to second]
grp_city_date_avg_delivery_time: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 1 more field]
grp_state_date_avg_delivery_time: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 1 more field]
```



```
06:04 PM (<1s)                                                    19                                                    Python

## ---------------------- INSIGHTS TO BE STORED -----------------
# historical daily insights into 3 tables

#i. Insights per period
#ii. Insights per period per city
#ii. Insights per period per state

## Per period
grp_date_insights = reduce(
lambda x, y: x.join(y, on=["order_placed_date"], how="left"),
[
grp_date_total_sales,
grp_date_total_freight_charge,
grp_date_total_order_count,
grp_date_avg_freight_charge,
grp_date_avg_review_score,
grp_date_avg_approval_time,
grp_date_avg_delivery_time,
]
)

## Per city
grp_city_date_insights = reduce(
lambda x, y: x.join(y, on=["customer_city", "order_placed_date"], how="left"),
[
grp_city_date_total_sales,
grp_city_date_total_freight_charge,
grp_city_date_total_order_count,
grp_city_date_vg_freight_charge,
grp_city_date_avg_review_score,
grp_city_date_avg_approval_time,
grp_city_date_avg_delivery_time,
]
)
```



```
## Per State
grp_state_date_insights = reduce(
lambda x, y: x.join(y, on=["customer_state", "order_placed_date"], how="left"),
[
grp_state_date_total_sales,
grp_state_date_total_freight_charge,
grp_state_date_total_order_count,
grp_state_date_vg_freight_charge,
grp_state_date_avg_review_score,
grp_state_date_avg_approval_time,
grp_state_date_avg_delivery_time,
]
)
```

```
grp_date_insights: pyspark.sql.dataframe.DataFrame = [order_placed_date: date, total_sales: double ... 6 more fields]
grp_city_date_insights: pyspark.sql.dataframe.DataFrame = [customer_city: string, order_placed_date: date ... 7 more fields]
grp_state_date_insights: pyspark.sql.dataframe.DataFrame = [customer_state: string, order_placed_date: date ... 7 more fields]
```

```
06:04 PM (2s)                                                    20

display(grp_city_date_insights)
(14) Spark Jobs
```

Table

	customer_city	order_placed_date	1.2 total_sales	1.2 total_freight_charge	total_order_count	1.2 avg_freight_charge	1.2 avg_review_score	avg_approv
1	Sao Paulo	2017-11-27	7866.610000000001	808.9899999999999	51	15.86254901960784	4	{"seconds":2
2	Sao Goncalo	2017-01-30	19.99	16.32	1	16.32	5	{"seconds":8

Week 4: Data analysis and visualization

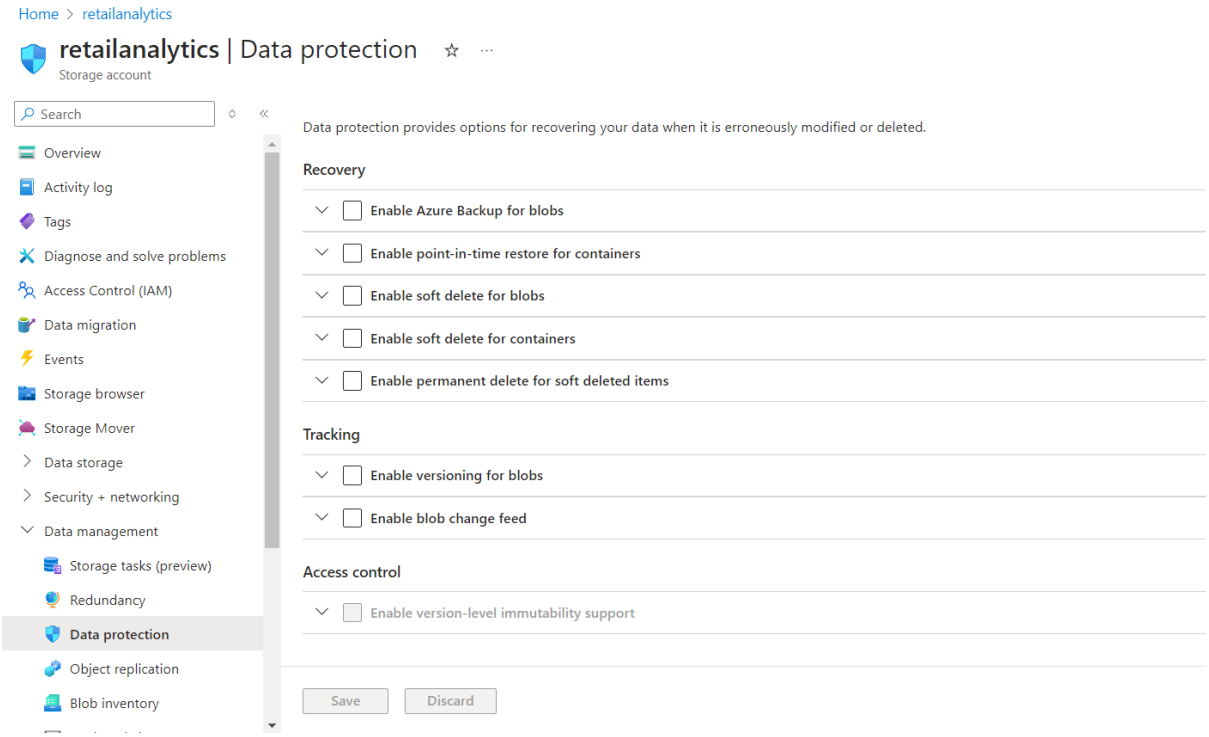
Step 1: Write the results into HDFS

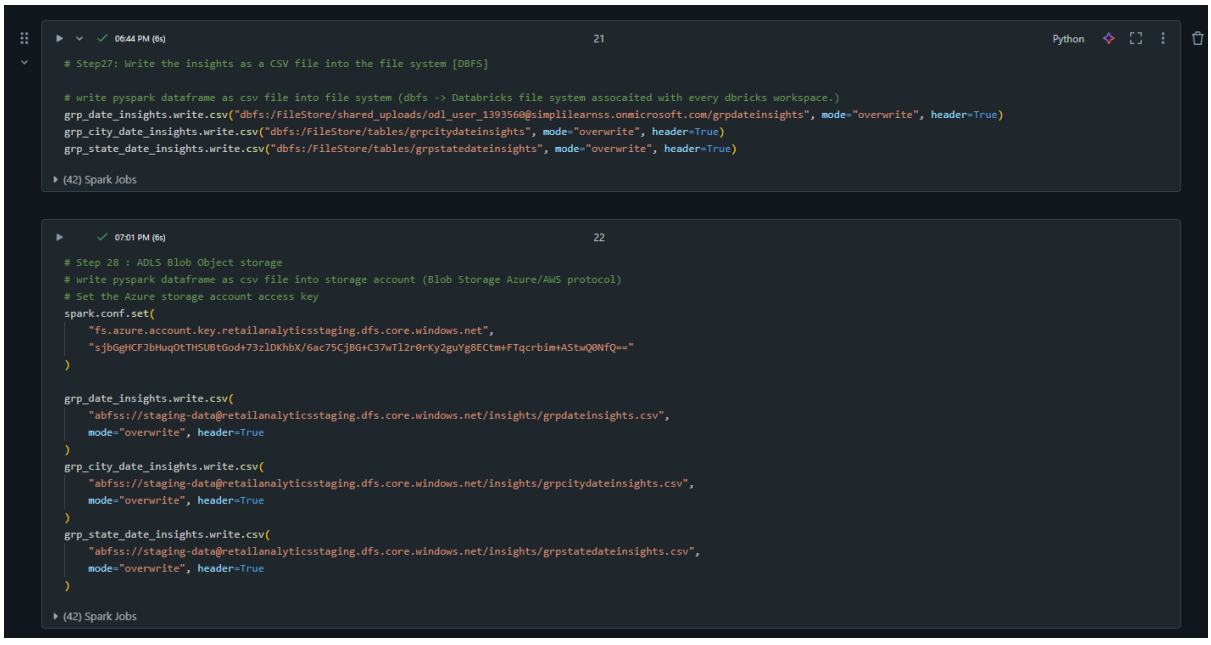
Step 2: Save the final dataset into object storage service per the cloud platform

Step 3: Create a DB cluster that is also a NoSQL using the relevant service on the cloud platform

Step 4: Save insights in the NoSQL DB mentioned in the previous step

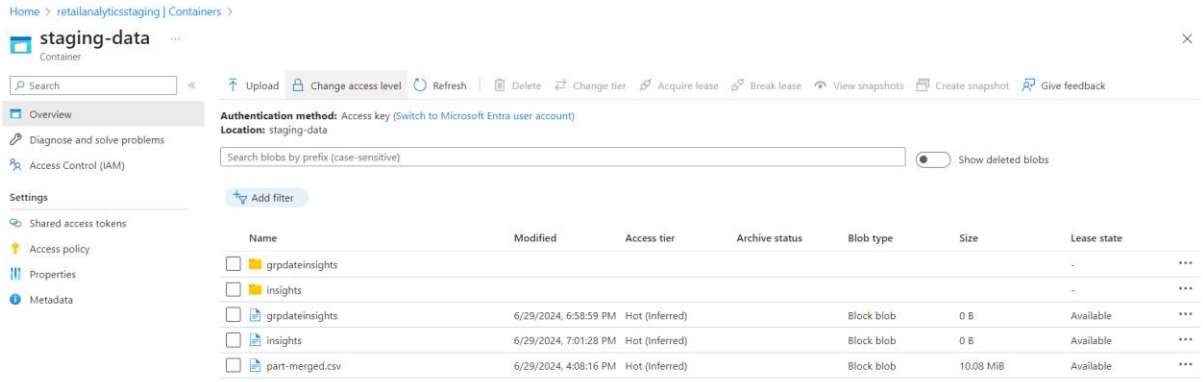
Remove “Enable soft delete for blobs” and “Enable soft delete for containers

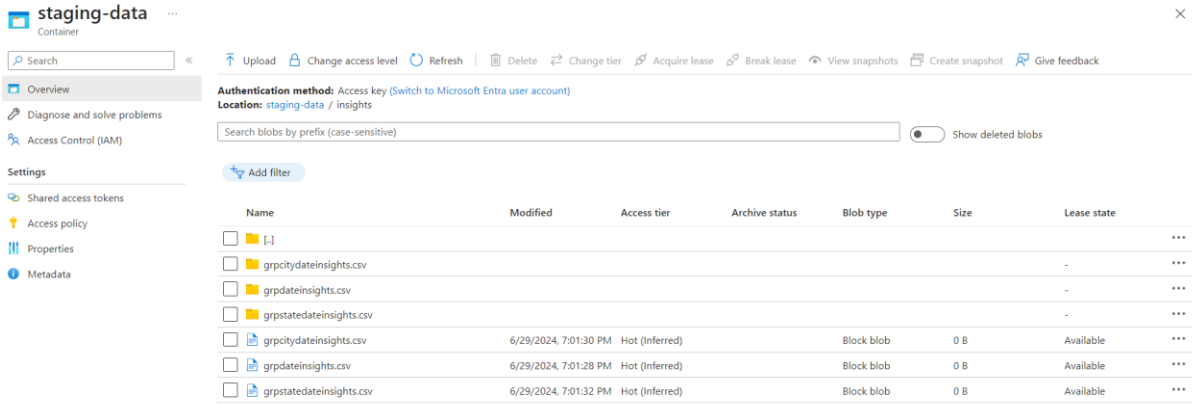


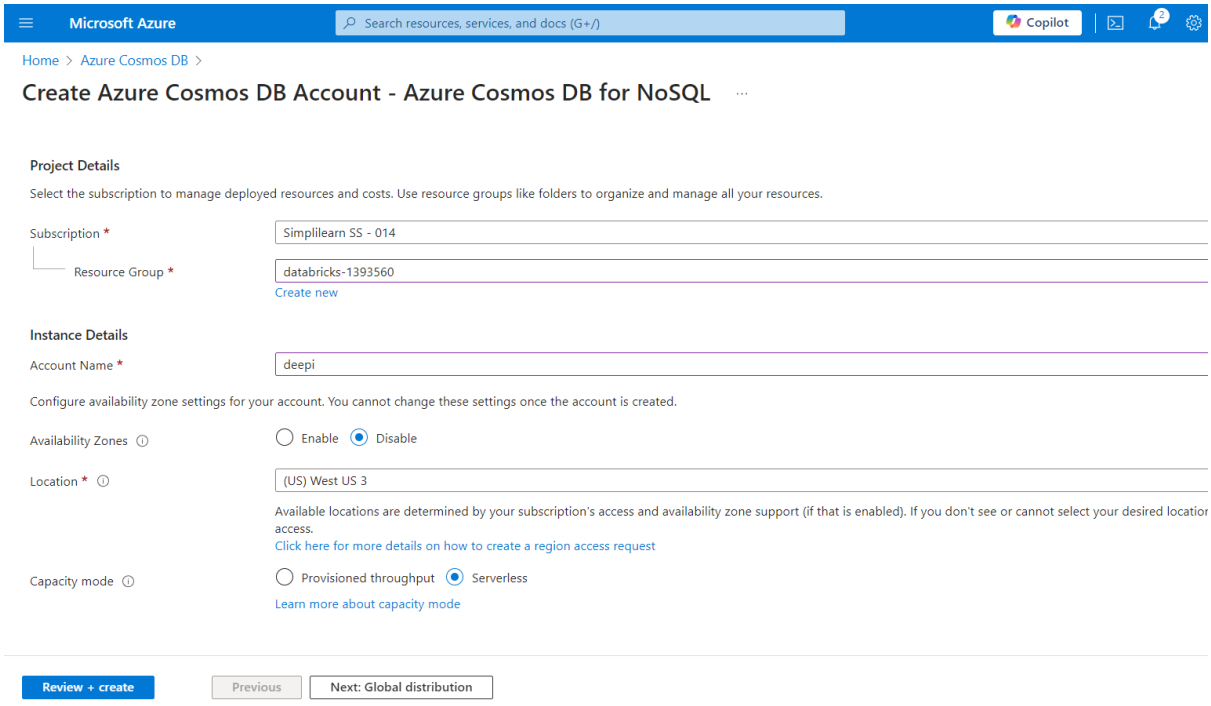


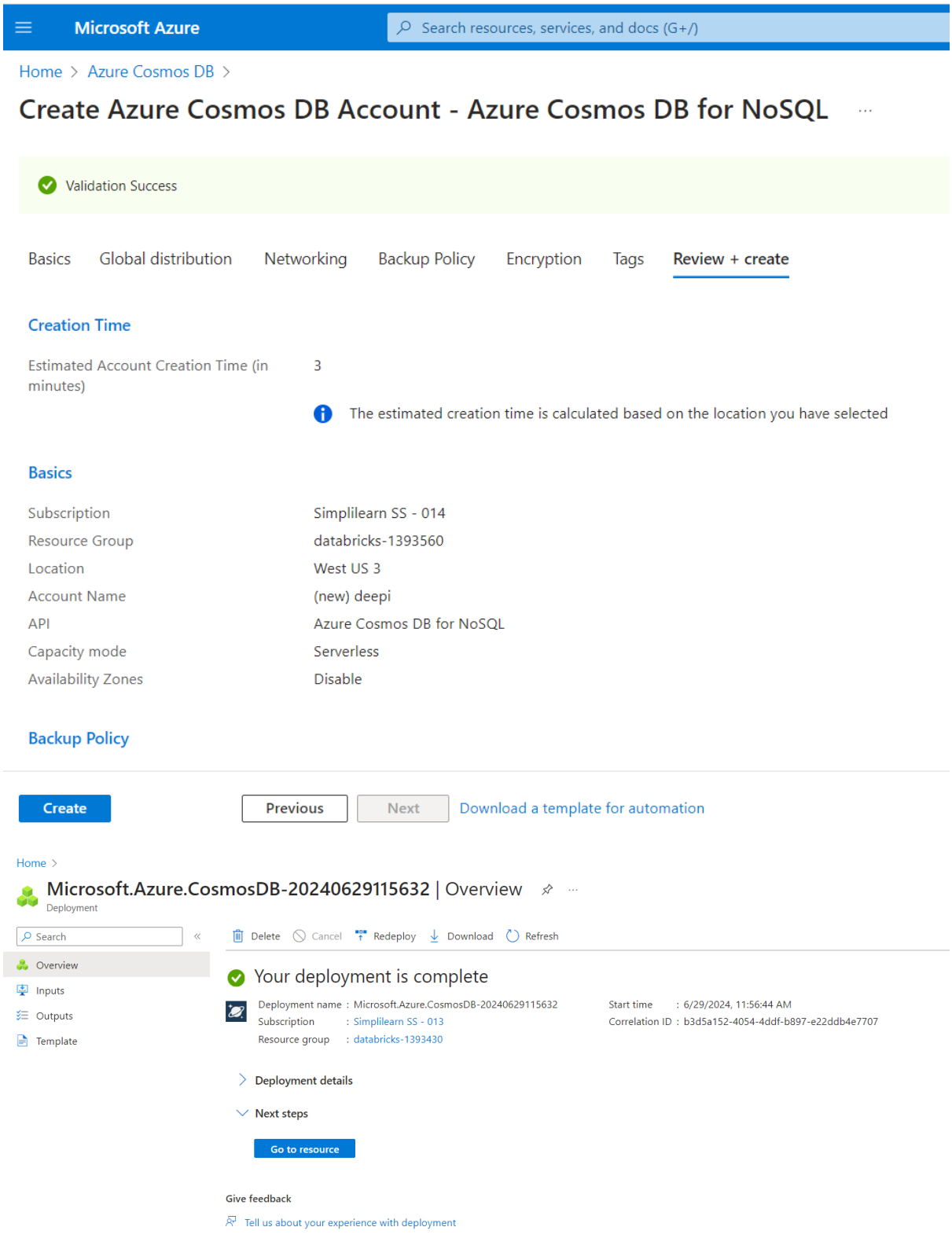
```
# Step27: Write the insights as a CSV file into the file system [DBFS]

# write pyspark dataframe as csv file into file system (dbfs -> Databricks file system assocaited with every dbricks workspace.)
grp_date_insights.write.csv("dbfs:/FileStore/shared_uploads/odl_user_1393560@simplilearnss.onmicrosoft.com/grpdateinsights", mode="overwrite", header=True)
grp_city_date_insights.write.csv("dbfs:/FileStore/tables/grpcitydateinsights", mode="overwrite", header=True)
grp_state_date_insights.write.csv("dbfs:/FileStore/tables/grpstatedateinsights", mode="overwrite", header=True)
```

```
# Step 28 : ADLS Blob Object storage
# write pyspark dataframe as csv file into storage account (Blob Storage Azure/AWS protocol)
# Set the Azure storage account access key
spark.conf.set(
    "fs.azure.account.key.retailanalyticsstaging.dfs.core.windows.net",
    "sjbGgHCFJbHuqOtHSUBtGod+73zlDKhbX/6ac75CjBG+C37wTl2r0rKy2guYg8ECtm+FTqcrbim+AStwQ0NfQ=="
)

grp_date_insights.write.csv(
    "abfss://staging-data@retailanalyticsstaging.dfs.core.windows.net/insights/grpdateinsights.csv",
    mode="overwrite", header=True
)
grp_city_date_insights.write.csv(
    "abfss://staging-data@retailanalyticsstaging.dfs.core.windows.net/insights/grpcitydateinsights.csv",
    mode="overwrite", header=True
)
grp_state_date_insights.write.csv(
    "abfss://staging-data@retailanalyticsstaging.dfs.core.windows.net/insights/grpstatedateinsights.csv",
    mode="overwrite", header=True
)
```

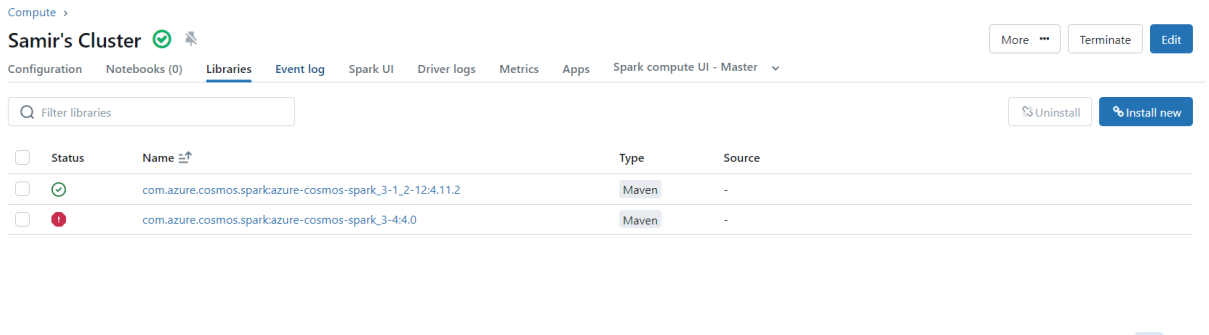

Home > retailanalyticsstaging | Containers >
staging-data
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Access policy
Properties
Metadata
Upload
Change access level
Refresh
Delete
Change tier
Acquire lease
Break lease
View snapshots
Create snapshot
Give feedback
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: staging-data
Search blobs by prefix (case-sensitive)
Show deleted blobs
Add filter
Name
Modified
Access tier
Archive status
Blob type
Size
Lease state
grpdateinsights
insights
grpdateinsights 6/29/2024, 6:58:59 PM Hot (Inferred) Block blob 0 B Available
insights 6/29/2024, 7:01:28 PM Hot (Inferred) Block blob 0 B Available
part-merged.csv 6/29/2024, 4:08:16 PM Hot (Inferred) Block blob 10.08 MiB Available


staging-data
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Access policy
Properties
Metadata
Upload
Change access level
Refresh
Delete
Change tier
Acquire lease
Break lease
View snapshots
Create snapshot
Give feedback
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: staging-data / insights
Search blobs by prefix (case-sensitive)
Show deleted blobs
Add filter
Name
Modified
Access tier
Archive status
Blob type
Size
Lease state
[..]
grpcitydateinsights.csv
grpdateinsights.csv
grpstatedateinsights.csv
grpcitydateinsights.csv 6/29/2024, 7:01:30 PM Hot (Inferred) Block blob 0 B Available
grpdateinsights.csv 6/29/2024, 7:01:28 PM Hot (Inferred) Block blob 0 B Available
grpstatedateinsights.csv 6/29/2024, 7:01:32 PM Hot (Inferred) Block blob 0 B Available


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > Azure Cosmos DB >
Create Azure Cosmos DB Account - Azure Cosmos DB for NoSQL
Project Details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
Simplilearn SS - 014
Resource Group *
databricks-1393560
Create new
Instance Details
Account Name *
deepi
Configure availability zone settings for your account. You cannot change these settings once the account is created.
Availability Zones
Enable
Disable
Location *
(US) West US 3
Available locations are determined by your subscription's access and availability zone support (if that is enabled). If you don't see or cannot select your desired locatior
access.
Click here for more details on how to create a region access request
Capacity mode
Provisioned throughput
Serverless
Learn more about capacity mode
Review + create
Previous
Next: Global distribution


Microsoft Azure
Search resources, services, and docs (G+/)
Home > Azure Cosmos DB >
Create Azure Cosmos DB Account - Azure Cosmos DB for NoSQL
Validation Success
Basics Global distribution Networking Backup Policy Encryption Tags Review + create
Creation Time
Estimated Account Creation Time (in minutes) 3
The estimated creation time is calculated based on the location you have selected
Basics
Subscription Simplilearn SS - 014
Resource Group databricks-1393560
Location West US 3
Account Name (new) deepi
API Azure Cosmos DB for NoSQL
Capacity mode Serverless
Availability Zones Disable
Backup Policy
Create
Previous
Next
Download a template for automation
Home >
Microsoft.Azure.CosmosDB-20240629115632 | Overview
Deployment
Search
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Download
Refresh
Your deployment is complete
Deployment name : Microsoft.Azure.CosmosDB-20240629115632
Subscription : Simplilearn SS - 013
Resource group : databricks-1393430
Start time : 6/29/2024, 11:56:44 AM
Correlation ID : b3d5a152-4054-4ddf-b897-e22ddb4e7707
Deployment details
Next steps
Go to resource
Give feedback
Tell us about your experience with deployment

1. Navigate to your Databricks workspace.
2. Go to the "Clusters" section and select the cluster where you want to install the connector.
3. In the cluster configuration, find the "Libraries" section and click on "Install New".
4. Choose "Maven" as the source for the library.
5. In the search box, enter the Maven coordinates for the Azure Cosmos DB Spark connector (com.azure.cosmos.spark:azure-cosmos-spark_3-4:4.0)




10:52 PM (5s)
25
from pyspark.sql.functions import lit
from pyspark.sql.types import StringType
# Add the "id" field with a default value if it is missing
if "id" not in grp_date_insights.columns:
grp_date_insights = grp_date_insights.withColumn("id", lit("default_id"))
# Convert the "id" field to string if it is not already
if grp_date_insights.schema["id"].dataType != StringType():
grp_date_insights = grp_date_insights.withColumn("id", grp_date_insights["id"].cast(StringType()))
# Save the DataFrame
grp_date_insights.write.format("cosmos.oltp").options(**config).mode("append").save()
(14) Spark Jobs