

Introduction to Big Data

Graded Assignment 5

Name - Avijeet Palit

Roll - 21f1005675

Date - 09/11/2024

Assignment Question

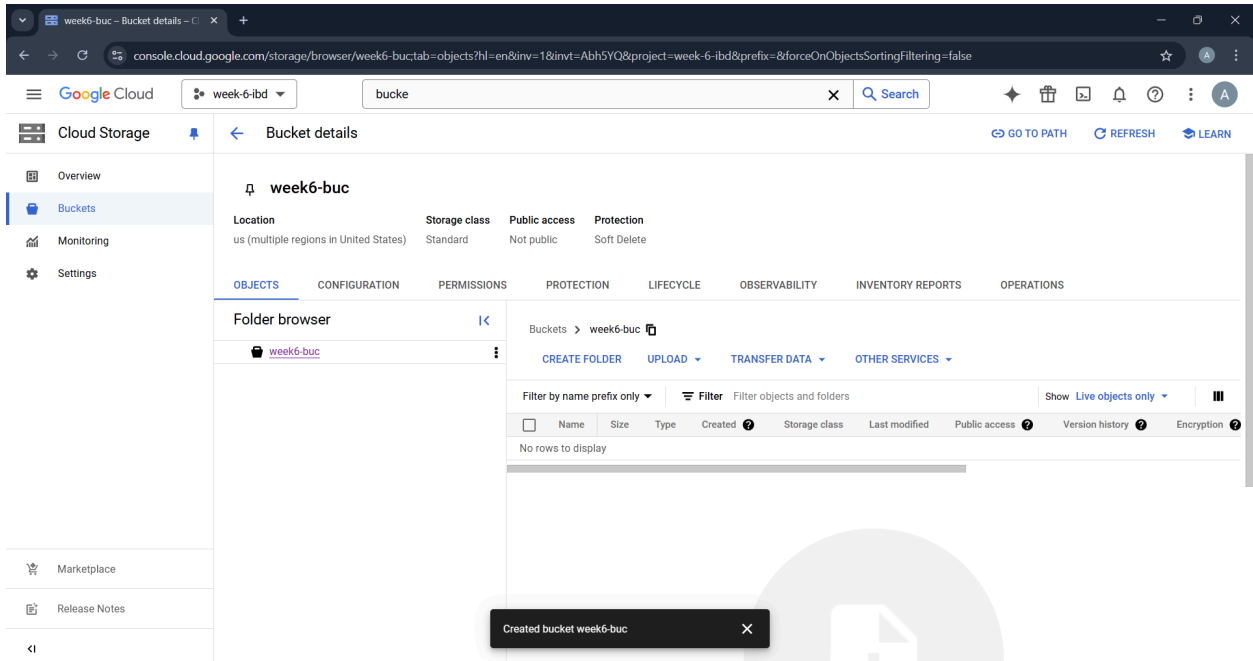
Write SparkSQL code to implement Slowly Changing Dimension (SCD) Type II on a customer master data frame. The SparkSQL code should be executed on a Dataproc cluster using the input files created for Assignment 4.

Implementation Steps

Step 1: Create a Cloud Storage Bucket

Google Cloud Storage (GCS) is used to store the input and output files for Spark jobs.

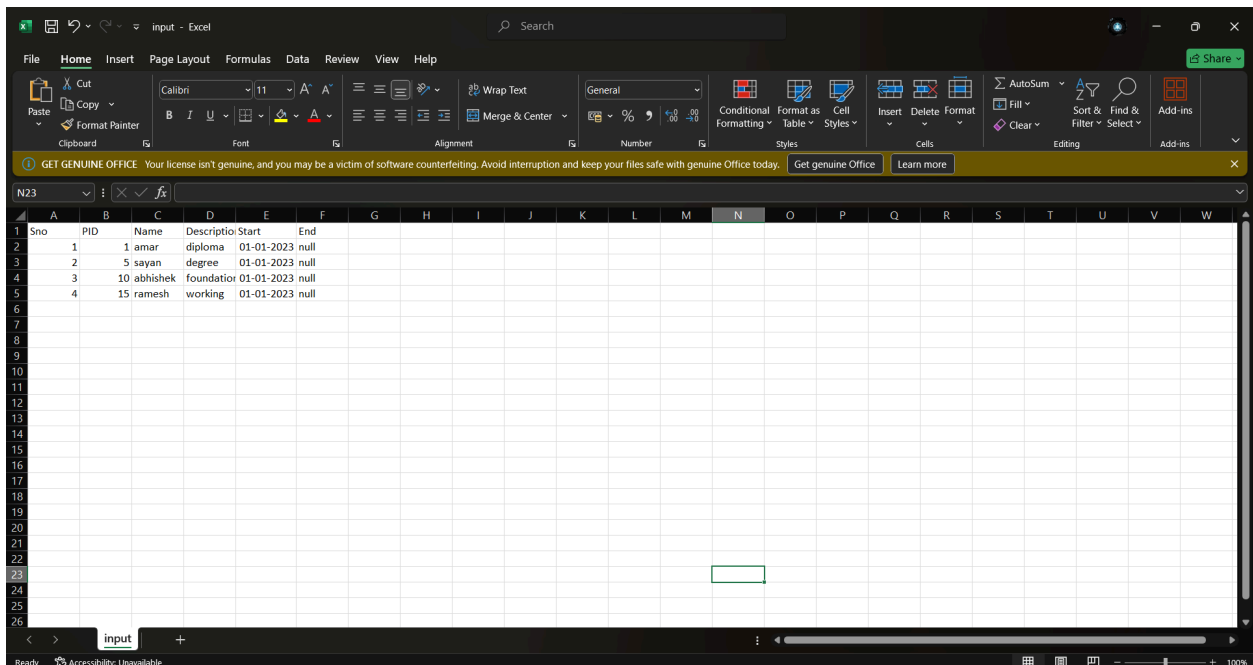
1. Navigate to Google Cloud Console.
2. Click on "Create Bucket."
3. Name the bucket (e.g., **week6-buc**).



4. Use the default options to complete bucket creation.

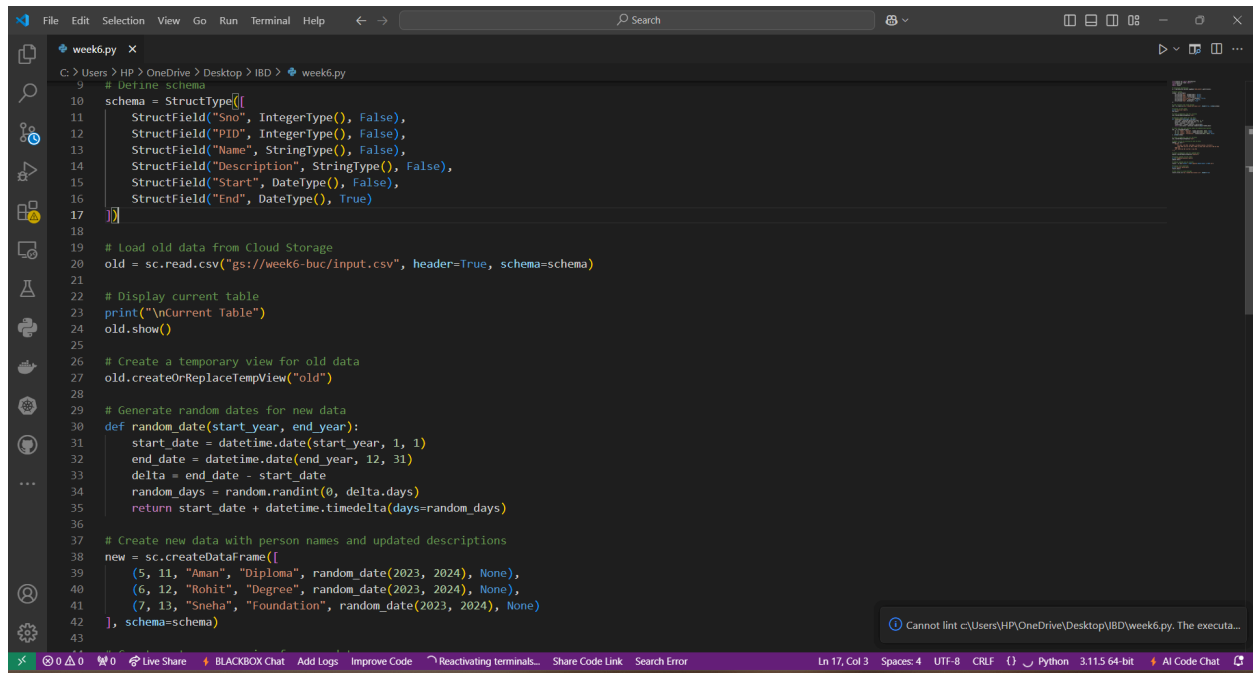
Step 2: Prepare Input File

Create a CSV file manually to serve as the input for the customer master data frame processing.



Step 3: Write PySpark Code

Develop PySpark code to implement SCD Type II logic on the customer master data frame. Save the code in a file named `week6.py`.

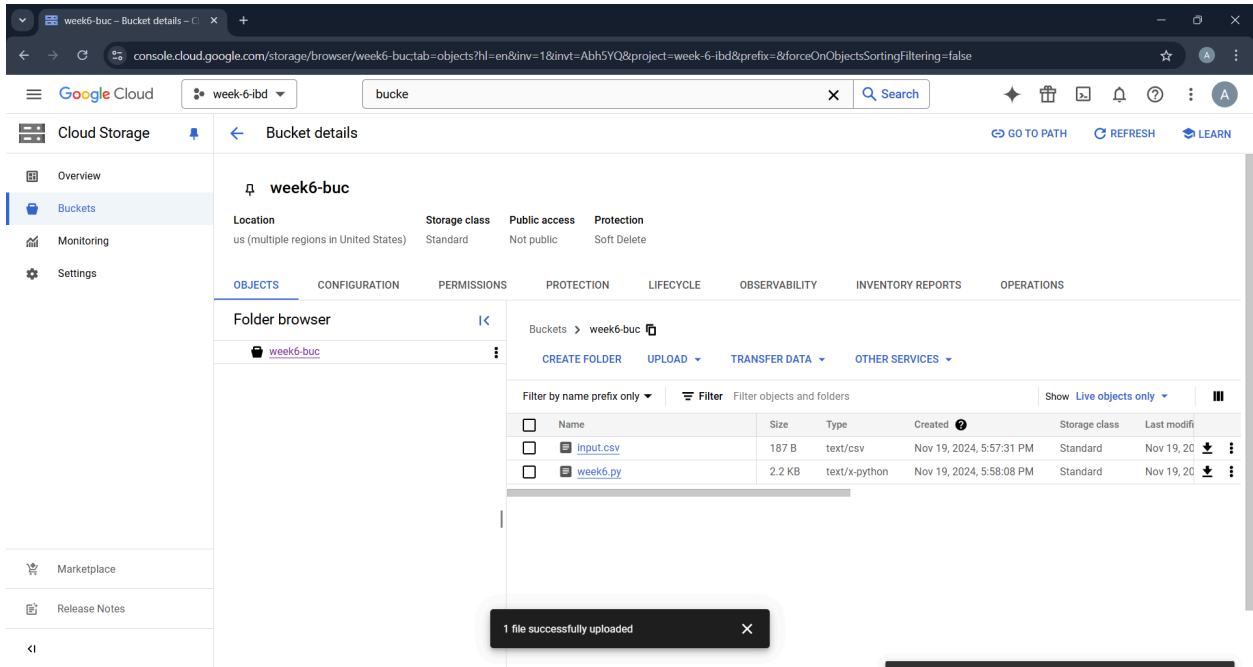


```
1 # Define schema
10 schema = StructType([
11     StructField("Sno", IntegerType(), False),
12     StructField("PID", IntegerType(), False),
13     StructField("Name", StringType(), False),
14     StructField("Description", StringType(), False),
15     StructField("Start", DateType(), False),
16     StructField("End", DateType(), True)
17 ])
18
19 # Load old data from Cloud Storage
20 old = sc.read.csv("gs://week6-buc/input.csv", header=True, schema=schema)
21
22 # Display current Table
23 print("\nCurrent Table")
24 old.show()
25
26 # Create a temporary view for old data
27 old.createOrReplaceTempView("old")
28
29 # Generate random dates for new data
30 def random_date(start_year, end_year):
31     start_date = datetime.date(start_year, 1, 1)
32     end_date = datetime.date(end_year, 12, 31)
33     delta = end_date - start_date
34     random_days = random.randint(0, delta.days)
35     return start_date + datetime.timedelta(days=random_days)
36
37 # Create new data with person names and updated descriptions
38 new = sc.createDataFrame([
39     (5, 11, "Aman", "Diploma", random_date(2023, 2024), None),
40     (6, 12, "Rohit", "Degree", random_date(2023, 2024), None),
41     (7, 13, "Sneha", "Foundation", random_date(2023, 2024), None)
42 ], schema=schema)
43
```

Cannot lint c:\Users\HP\OneDrive\Desktop\IBD\week6.py. The execution...

Step 4: Upload Files to Cloud Storage

1. Open the created bucket (`week6-buc`).
2. Upload the required files (`input.csv` and `week6.py`) into the bucket.



Step 5: Create a Dataproc Cluster

1. Navigate to **Dataproc** in the Google Cloud Console.
 - Go to the "Big Data" section on the left-hand menu or search for "Dataproc."
2. Click "Create Cluster" and configure the following settings:
 - **Cluster Name:** e.g., `spark-cluster`.
 - **Region:** Closest region, e.g., `us-central1`.
 - **Zone:** Leave as "No preference" if unsure.
 - **Cluster Mode:**
 - Use **Standard** for multi-role clusters (for large jobs).
 - Use **Single Node** for testing purposes.
 - **Machine Type:**
 - For small/test jobs: `n1-standard-2` or `n1-standard-4`.
 - **Worker Nodes:** Specify 2-4 nodes for small to medium workloads.
 - **Components:** Ensure PySpark is selected (default).
3. Click "Create" to initialize the cluster.

Step 6: Run Jobs on Dataproc

1. SSH into the cluster and run the following commands:

Failed to validate permissions required for default service account: '278755983826-compute@developer.gserviceaccount.com'. Cluster creation could still be successful if required permissions have been granted to the respective service accounts as mentioned in the document https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/service-accounts#dataproc_service_accounts_2. This could be due to Cloud Resource Manager API hasn't been enabled in your project '278755983826' before or it is disabled. Enable it by visiting <https://console.developers.google.com/apis/api/cloudresourcemanager.googleapis.com/overview?project=278755983826>.

Cluster details

Name: spark-cluster-2
Cluster UUID: 126007e3-55ee-48a9-9151-5e51d379995d
Type: Dataproc Cluster
Status: Running

MONITORING JOBS VM INSTANCES CONFIGURATION WEB INTERFACES

Filter: Filter instances

Name	Role	Machine type
spark-cluster-2-m	Master	n1-standard-2

EQUIVALENT REST

Copy the PySpark script from GCS:

```
gsutil cp gs://week6-buc/week6.py .
```

Execute the PySpark script:

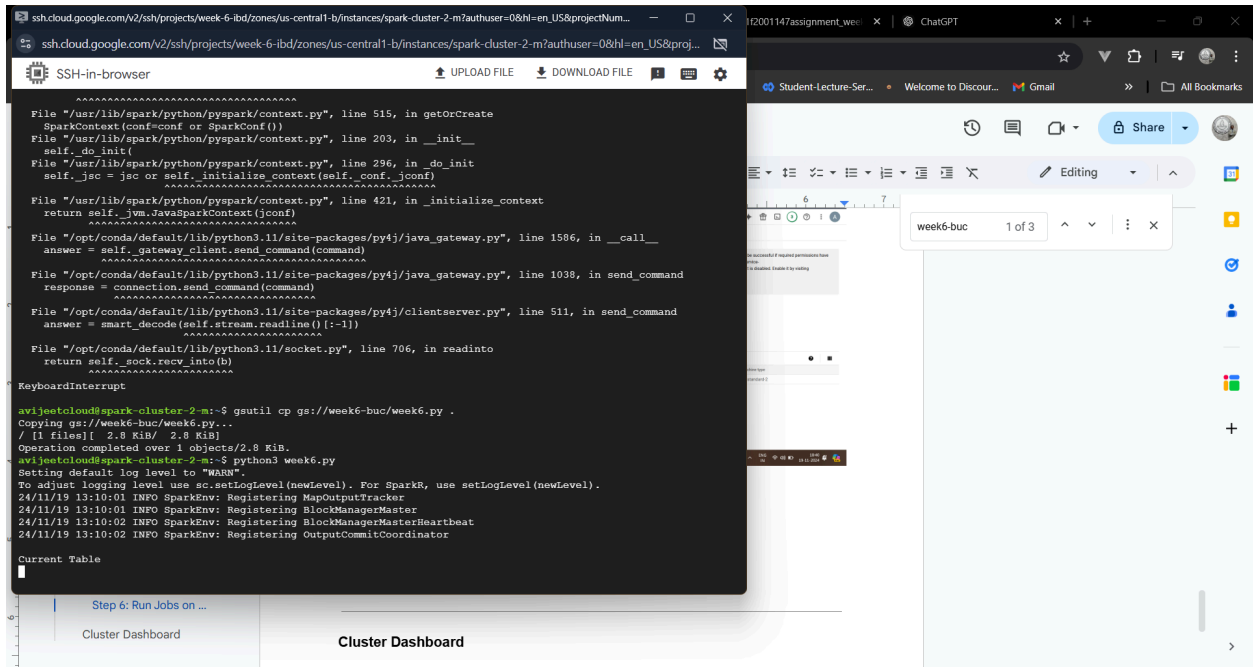
```
python3 week6.py
```

```
File "/usr/lib/spark/python/pyspark/context.py", line 515, in getOrCreate
SparkContext(conf=conf or SparkConf())
File "/usr/lib/spark/python/pyspark/context.py", line 203, in __init__
self.do_init()
File "/usr/lib/spark/python/pyspark/context.py", line 296, in do_init
self.jsc = jsc or self.initialize_context(self.conf._jconf)
File "/usr/lib/spark/python/pyspark/context.py", line 421, in initialize_context
return self._jvm.JavaSparkContext(jconf)
File "/opt/conda/default/lib/python3.11/site-packages/py4j/java_gateway.py", line 1586, in __call__
answer = self.gateway_client.send_command(command)
File "/opt/conda/default/lib/python3.11/site-packages/py4j/java_gateway.py", line 1038, in send_command
response = connection.send_command(command)
File "/opt/conda/default/lib/python3.11/site-packages/py4j/clientserver.py", line 511, in send_command
answer = smart_decode(self.stream.readline()[1:-1])
File "/opt/conda/default/lib/python3.11/socket.py", line 706, in readinto
return self._sock.recv_into(b)
KeyboardInterrupt

avijeetcloud@spark-cluster-2-m:~$ gsutil cp gs://week6-buc/week6.py .
Copying gs://week6-buc/week6.py...
/ [1 files][ 2.8 KiB/ 2.8 KiB]
Operation completed over 1 objects/2.8 KiB.
avijeetcloud@spark-cluster-2-m:~$ python3 week6.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/11/19 13:10:01 INFO SparkEnv: Registering MapOutputTracker
24/11/19 13:10:01 INFO SparkEnv: Registering BlockManagerMaster
24/11/19 13:10:02 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/11/19 13:10:02 INFO SparkEnv: Registering OutputCommitCoordinator

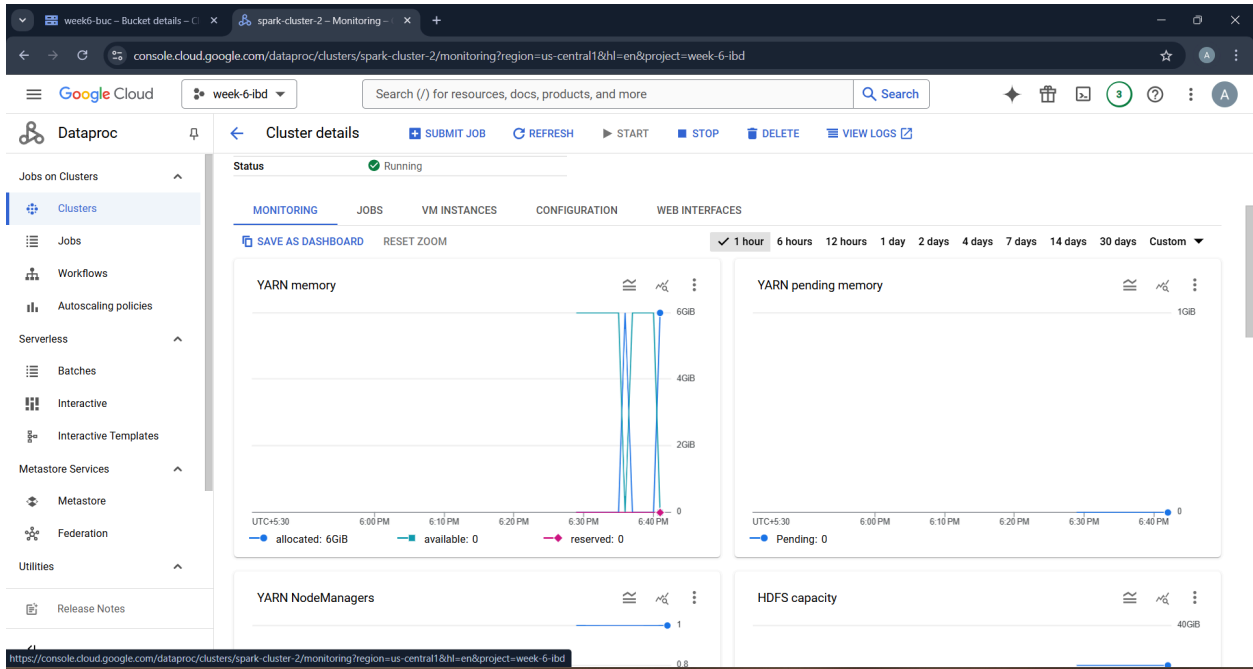
Current Table
```

2. View the output generated by the PySpark code.



Cluster Dashboard

After successfully creating the cluster, monitor the status and details on the Dataproc Clusters page. The status should display **Running** when the cluster is ready for use.



week6-buc - Bucket details

console.cloud.google.com/storage/browser/week6-buc?tab=objects&forceOnBucketsSortingFiltering=true&hl=en&project=week-6-ibd&prefix=&forceOnObjectsSortingFiltering=false

Google Cloud week-6-ibd Search (/) for resources, docs, products, and more

Cloud Storage Bucket details

GO TO PATH REFRESH LEARN

Overview Buckets Monitoring Settings

week6-buc

Location: us (multiple regions in United States) Storage class: Standard Public access: Not public Protection: Soft Delete

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE OBSERVABILITY INVENTORY REPORTS OPERATIONS

Folder browser

Buckets > week6-buc

CREATE FOLDER UPLOAD TRANSFER DATA OTHER SERVICES

Filter by name prefix only Filter objects and folders Show Live objects only

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	
<input type="checkbox"/>	google-cloud-dataproc-metainfo/	—	Folder	—	—	—	
<input type="checkbox"/>	input.csv	187 B	text/csv	Nov 19, 2024, 5:57:31 PM	Standard	Nov 19, 2024	
<input type="checkbox"/>	output.csv/	—	Folder	—	—	—	
<input type="checkbox"/>	week6.py	2.8 KB	text/x-python	Nov 19, 2024, 6:39:12 PM	Standard	Nov 19, 2024	

Marketplace Release Notes