

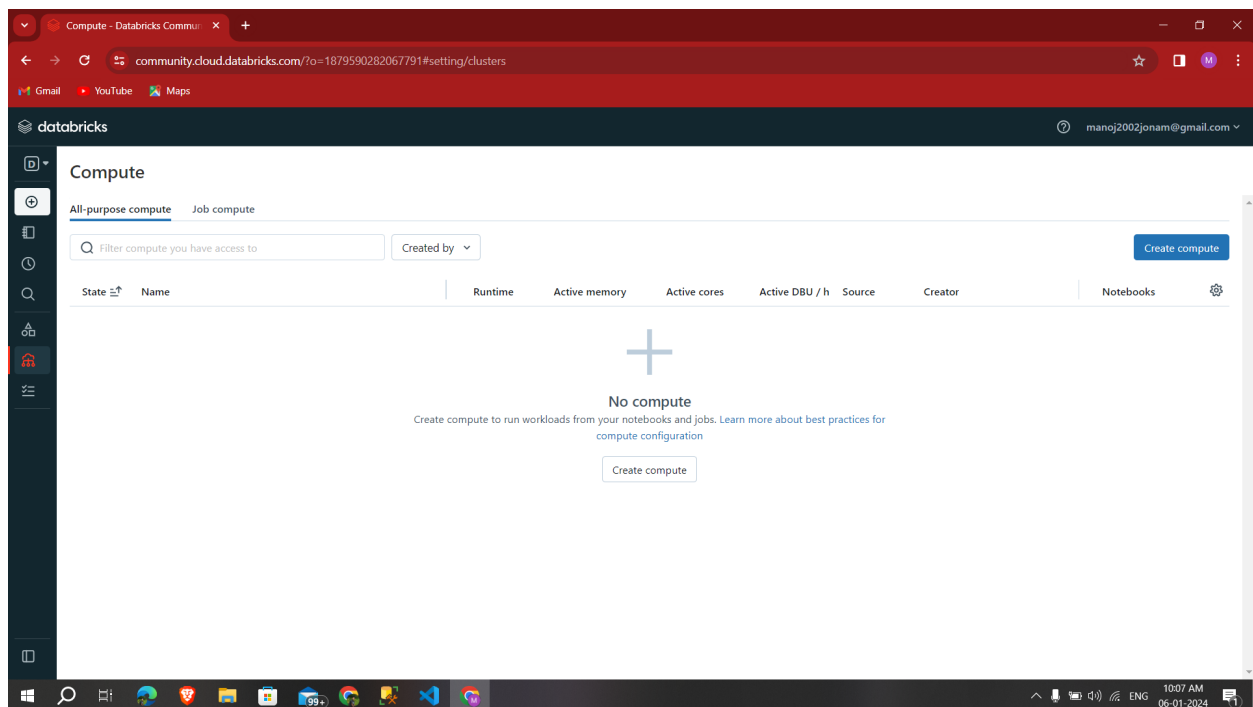
Name	Manoj Mani
Date	06/01/2024
Email Id	jonam1012@gmail.com
Type	Coding Assessment
Topic	Azure DataBricks

1. Create a cluster & Attach the notebook to the cluster and run all commands in the notebook & creates a DataFrame from a Databricks dataset & Create a Visualizations in Databricks notebooks & Rename, duplicate, or remove a visualization or data profile.

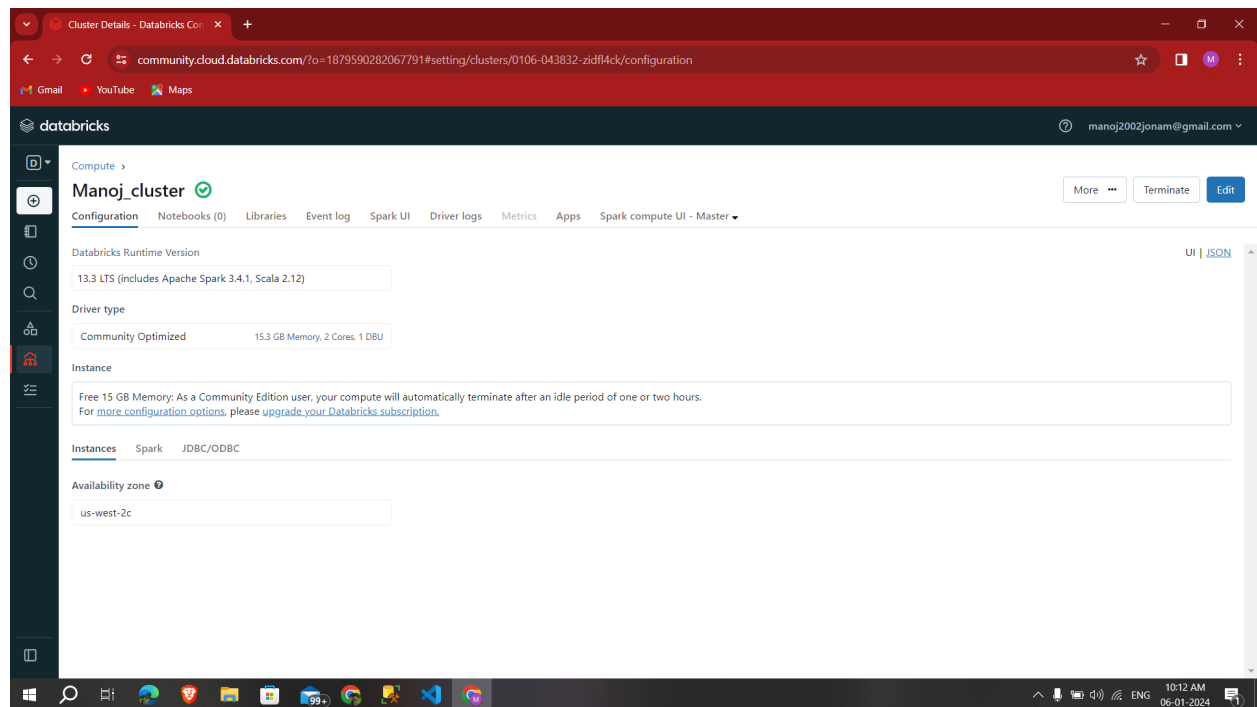
Azure Databricks is an Apache Spark-based analytics platform optimized for Azure. It provides a collaborative environment for big data analytics and machine learning.

Cluster: In Databricks, a **cluster refers to a set of computation resources** that are used to execute the code in your notebooks or jobs.

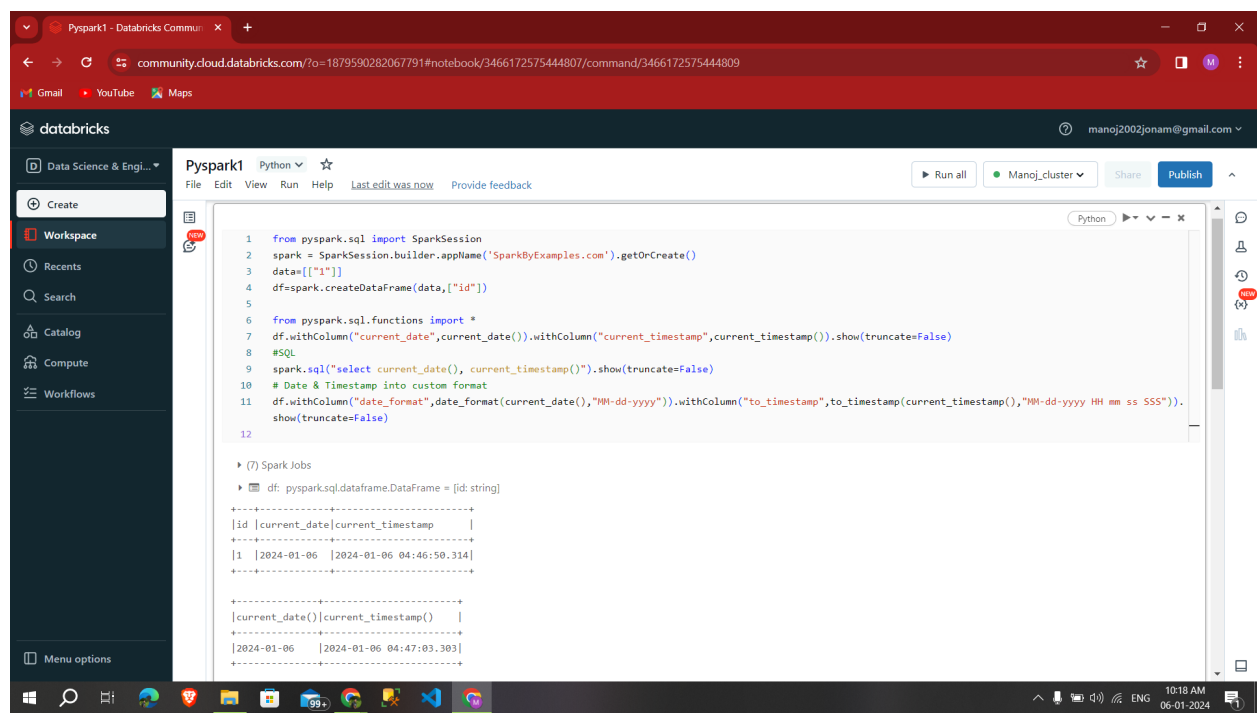
Step 1: I opened a databricks community edition and **by clicking on create compute able to create a cluster**



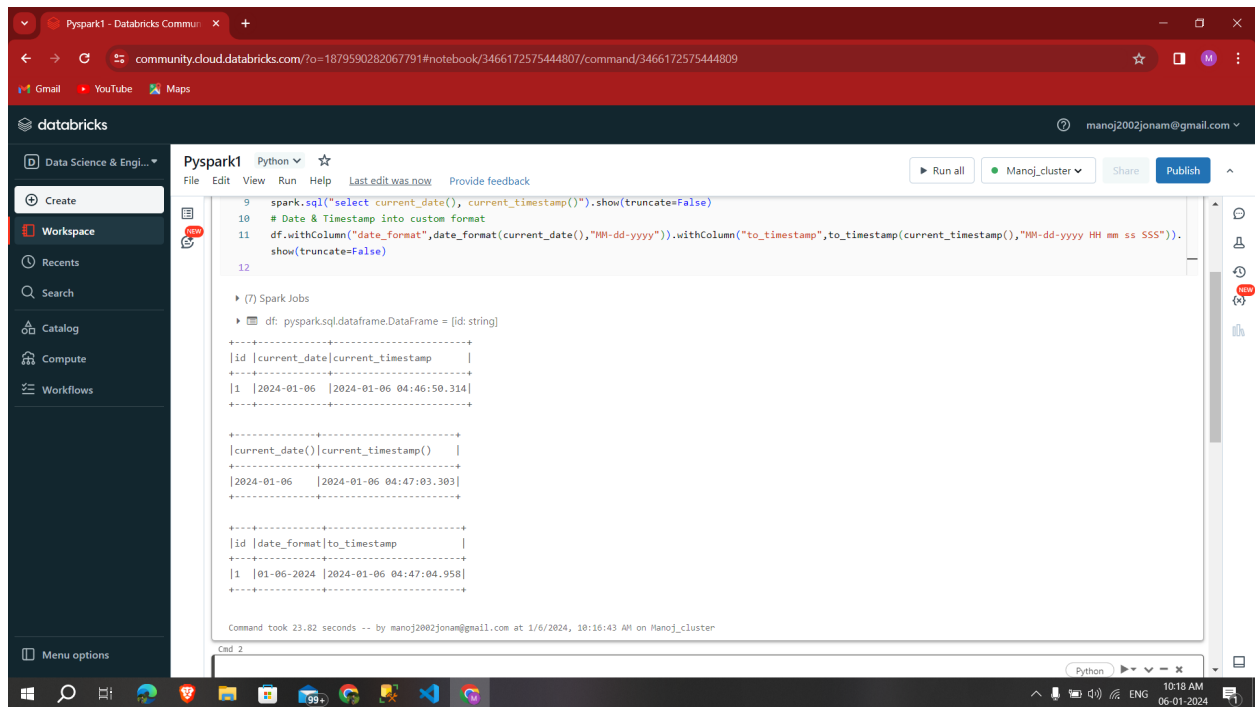
Step2 : I successfully created a cluster named **Manoj_cluster** and created a notebook to run the codes



Step 3: Created a notebook named Pyspark1 and run the **current time date pyspark command** and the output is shown below



Continuation



The screenshot shows a Databricks notebook interface. The top bar indicates the notebook is named 'Pyspark1' and is running on the 'Manoj_cluster'. The left sidebar contains navigation options like 'Create', 'Workspace', 'Recents', 'Search', 'Catalog', 'Compute', and 'Workflows'. The main area displays a Spark SQL command and its output.

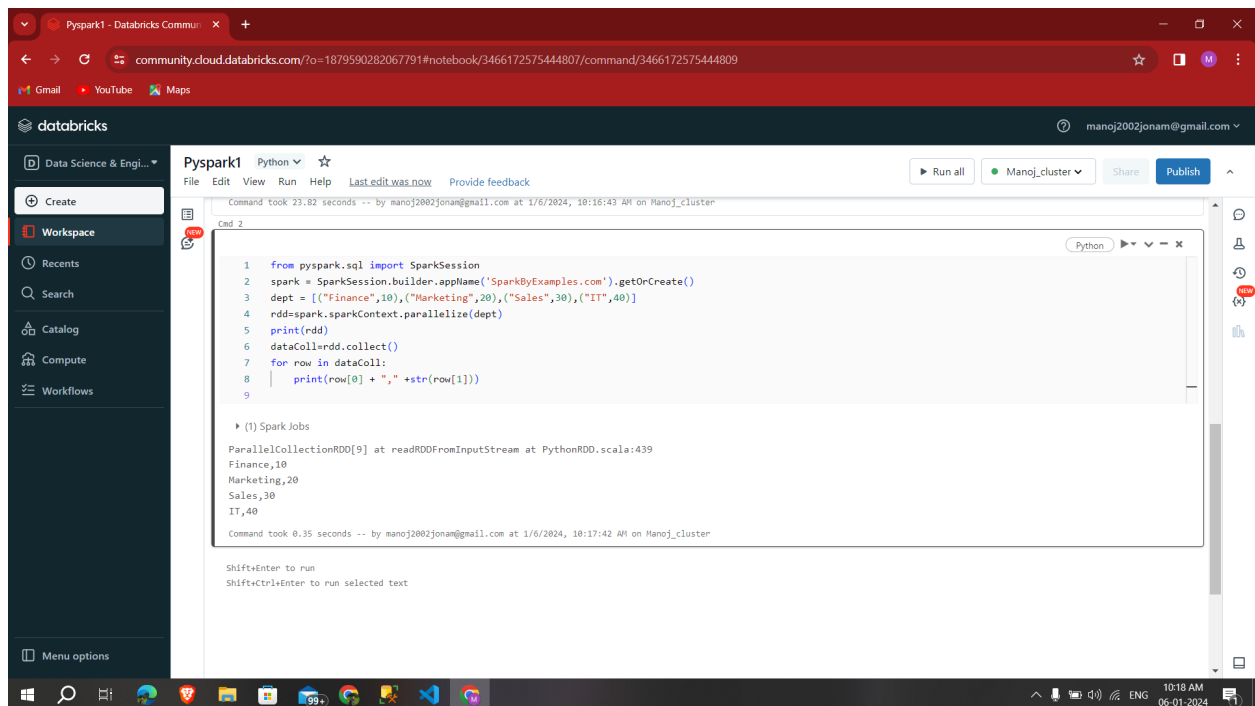
```
9 spark.sql("select current_date(), current_timestamp()").show(truncate=False)
10 # Date & Timestamp into custom format
11 df.withColumn("date_format",date_format(current_date(),"MM-dd-yyyy")).withColumn("to_timestamp",to_timestamp(current_timestamp(),"MM-dd-yyyy HH mm ss SSS")).
12 show(truncate=False)
```

Output:

```
┌── (7) Spark Jobs ──┐
┌── df: pyspark.sql.dataframe.DataFrame = [id: string] ───┐
┌───+───+───+───+───+───+───+───+───+───+───+───+───+───+───+───+───+───+───┐
|id|current_date|current_timestamp|
├──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──┐
|1|2024-01-06|2024-01-06 04:46:50.314|
├──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──┐
|current_date()|current_timestamp()|
├──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──┐
|2024-01-06|2024-01-06 04:47:03.303|
├──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──┐
|id|date_format|to_timestamp|
├──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──┐
|1|01-06-2024|2024-01-06 04:47:04.958|
├──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──+──┐
```

Command took 23.82 seconds -- by manoj2002jonam@gmail.com at 1/6/2024, 18:16:43 AM on Manoj_cluster

Step 4: Run another command **concept of rdd and parallelize** ,



The screenshot shows a Databricks notebook interface. The top bar indicates the notebook is named 'Pyspark1' and is running on the 'Manoj_cluster'. The left sidebar contains navigation options like 'Create', 'Workspace', 'Recents', 'Search', 'Catalog', 'Compute', and 'Workflows'. The main area displays a Spark RDD command and its output.

```
1 from pyspark.sql import SparkSession
2 spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()
3 dept = [("Finance",10),("Marketing",20),("Sales",30),("IT",40)]
4 rdd=spark.sparkContext.parallelize(dept)
5 print(rdd)
6 dataColl=rdd.collect()
7 for row in dataColl:
8     print(row[0] + ", " +str(row[1]))
9
```

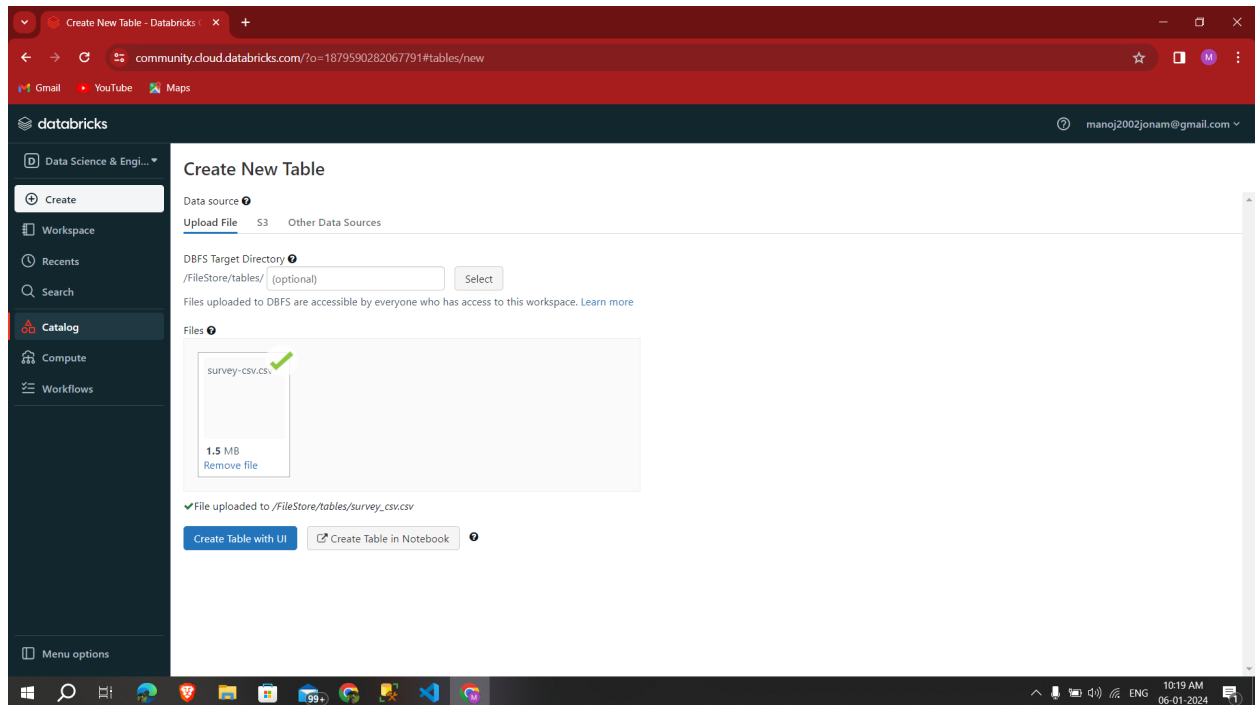
Output:

```
┌── (1) Spark Jobs ──┐
ParallelCollectionRDD[9] at readRDDFromInputStream at PythonRDD.scala:439
Finance,10
Marketing,20
Sales,30
IT,40
```

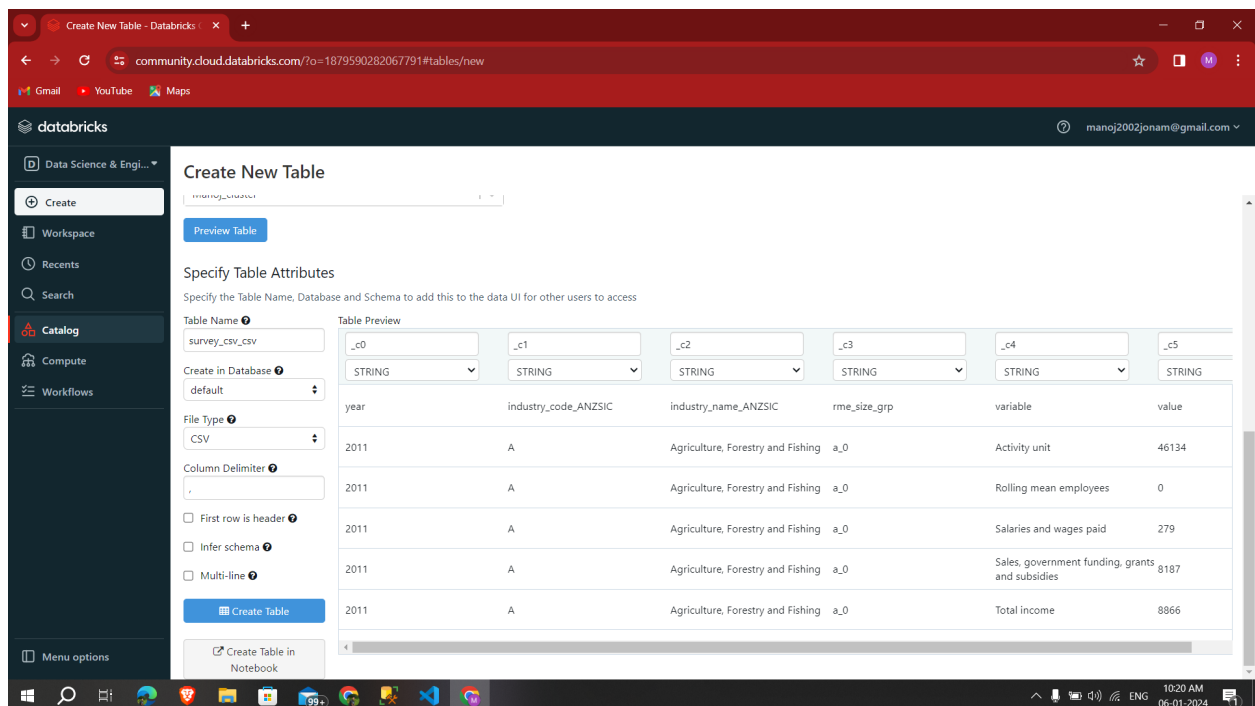
Command took 0.35 seconds -- by manoj2002jonam@gmail.com at 1/6/2024, 18:17:42 AM on Manoj_cluster

Shift+Enter to run
Shift+Ctrl+Enter to run selected text

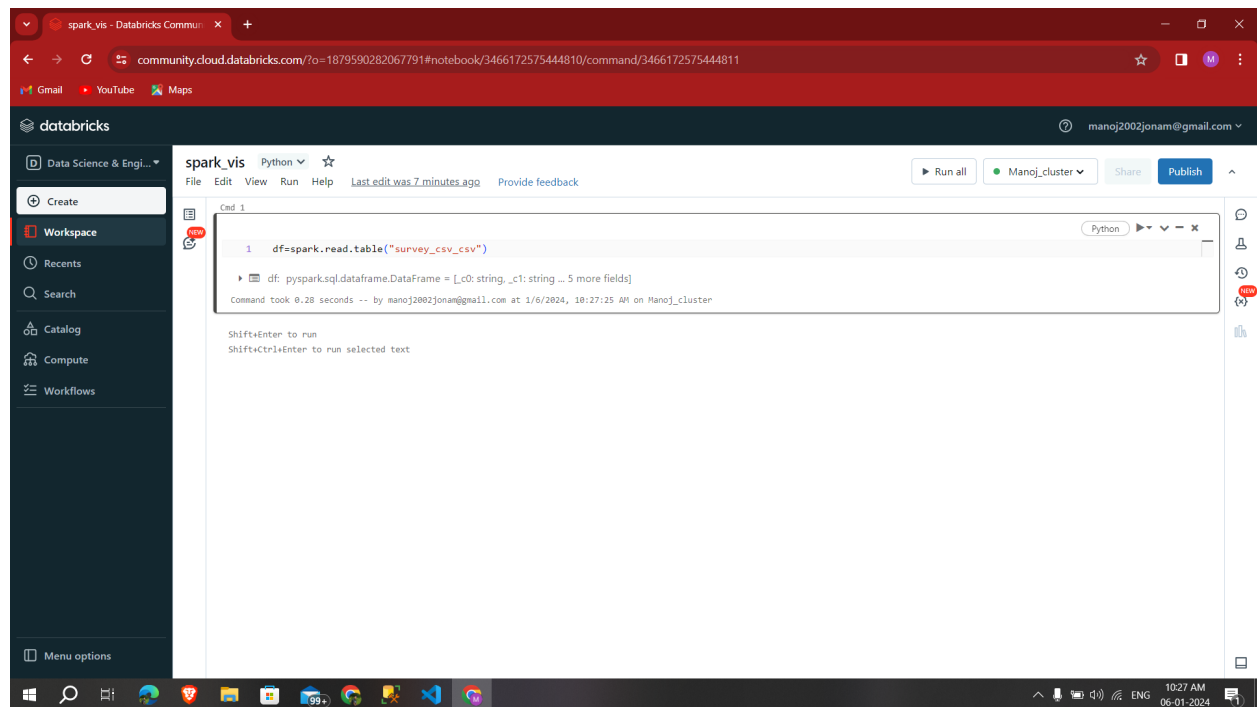
Step 5: Created a table by uploading the csv file from local to databricks



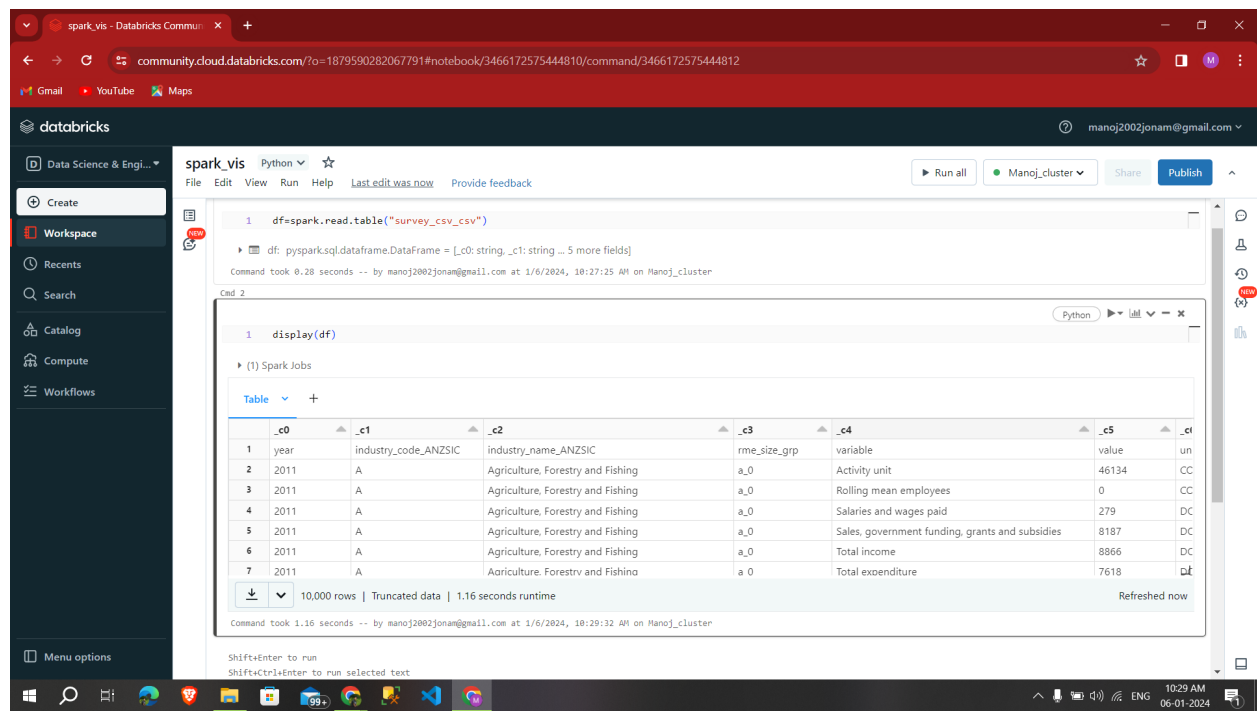
Step 6: By selecting the create table, the table was successfully added to the default storage



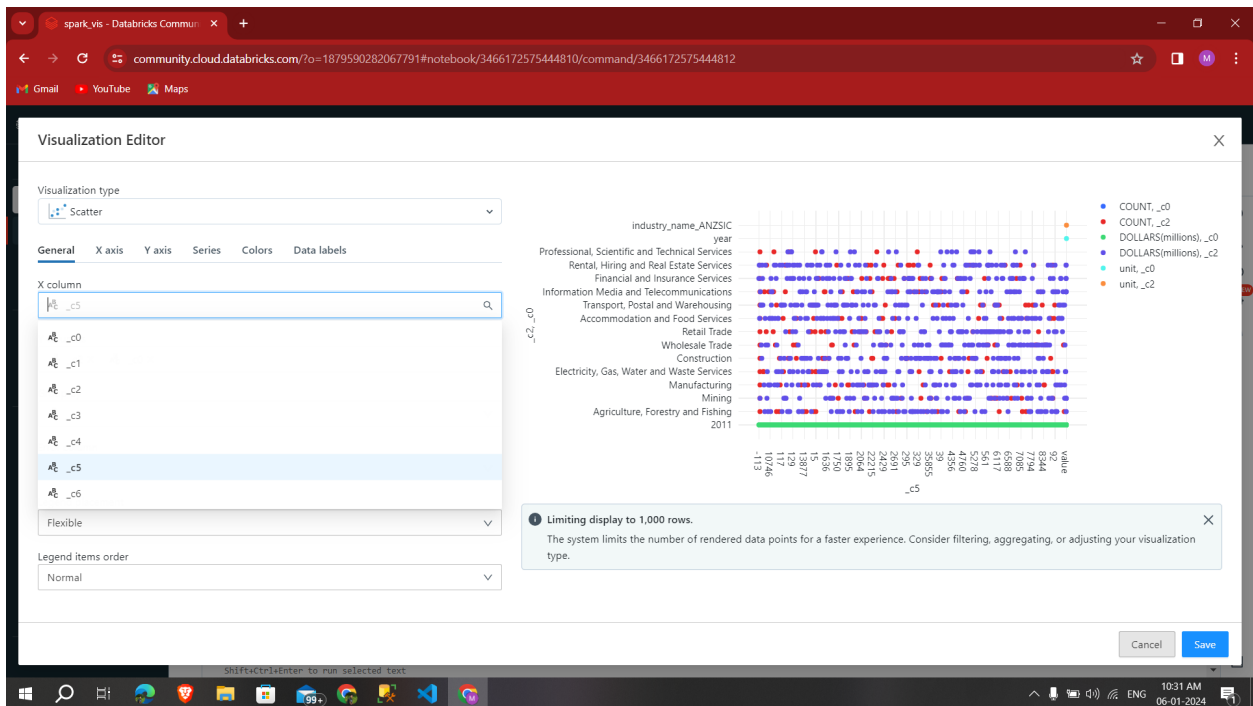
Step 7: Then i used the pyspark command to read the table and convert into the dataframe
`df=spark.read.csv("survey_csv_csv")`



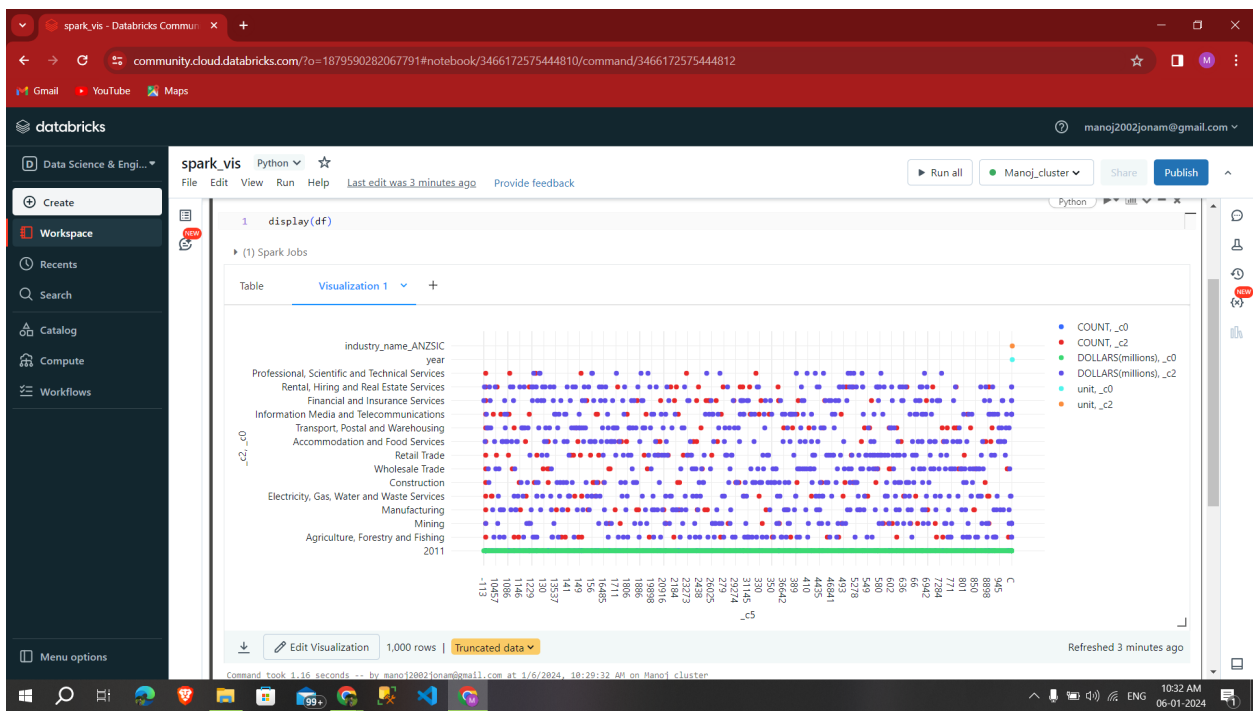
Step 8: By using display command displaying the table in the notebook, then click the plus option next to the table ,**select visualization**



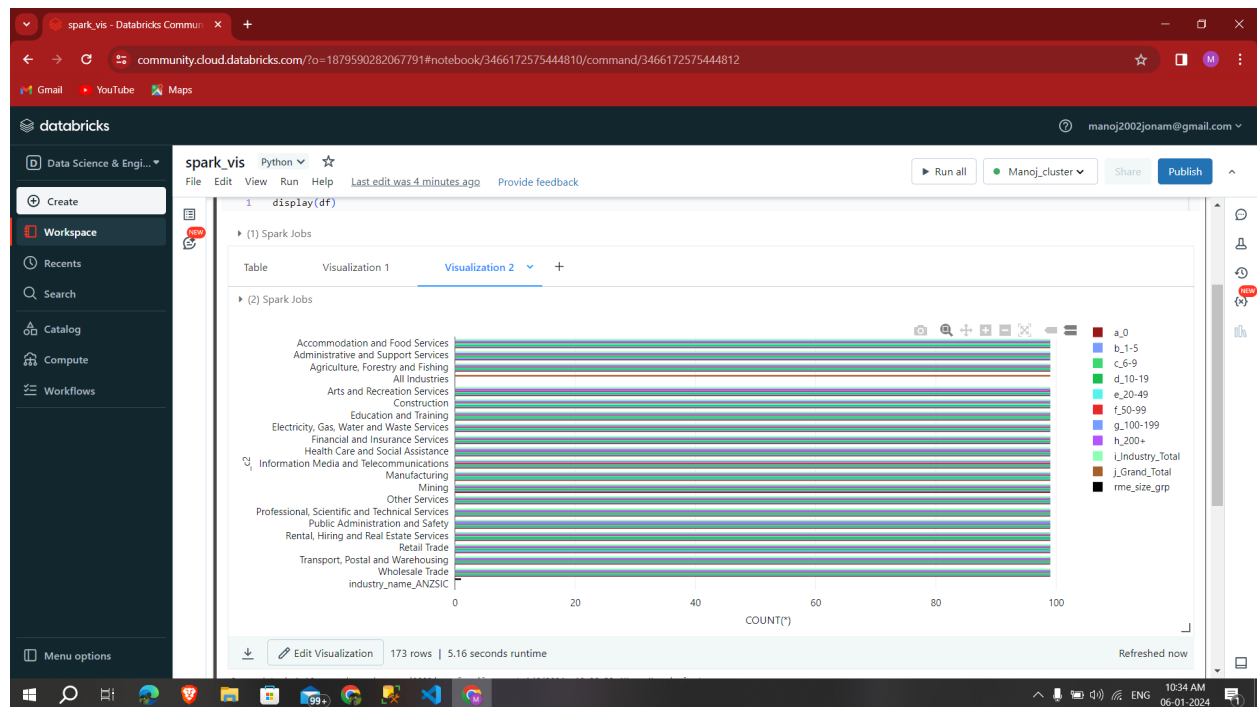
Step 9:After selecting visualization tab the editor pops up and **we can manage the x-axis and y-axis and save the visualization**



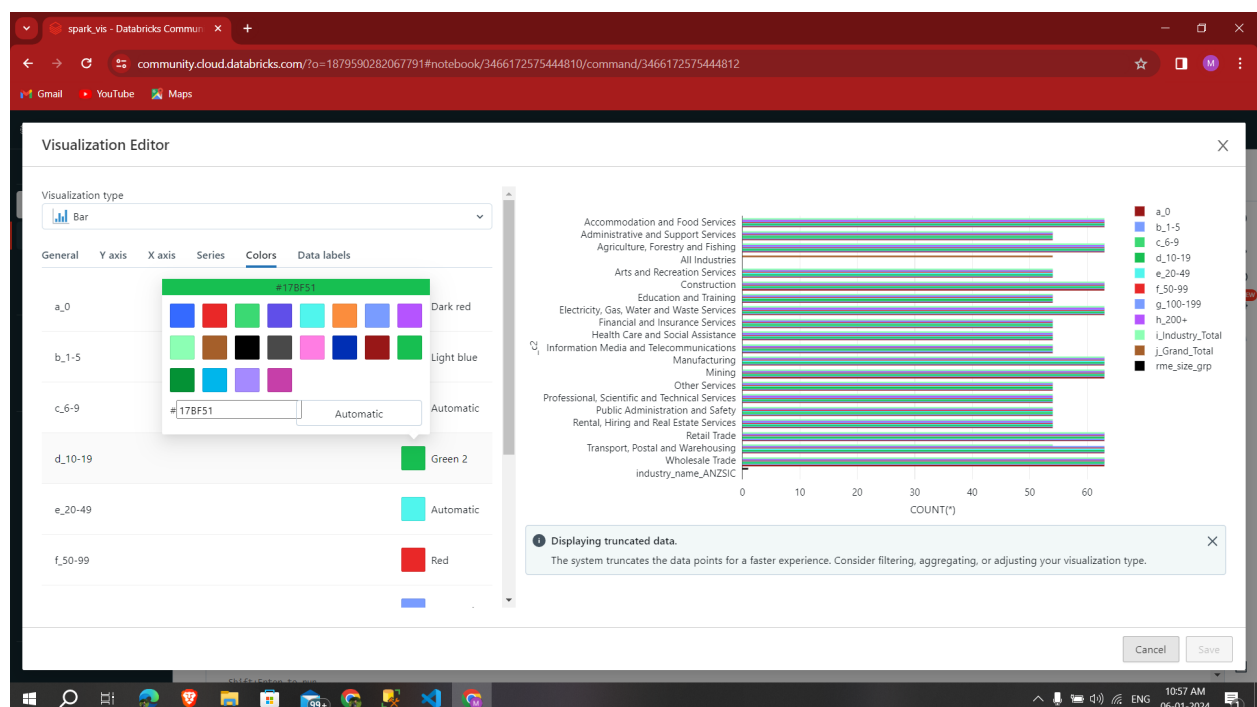
Step 10:The saved visualization in the output



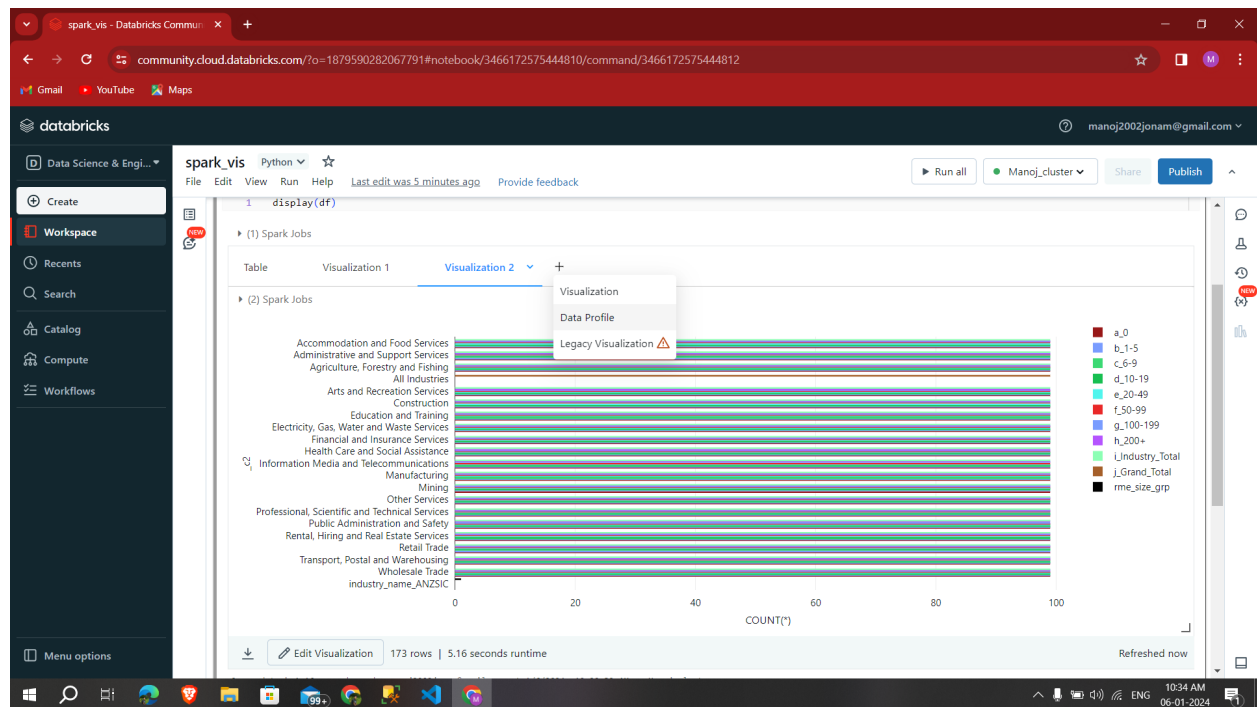
Step 11: Then i created a another **visualization** by using **bargraph** method and the output is shown below



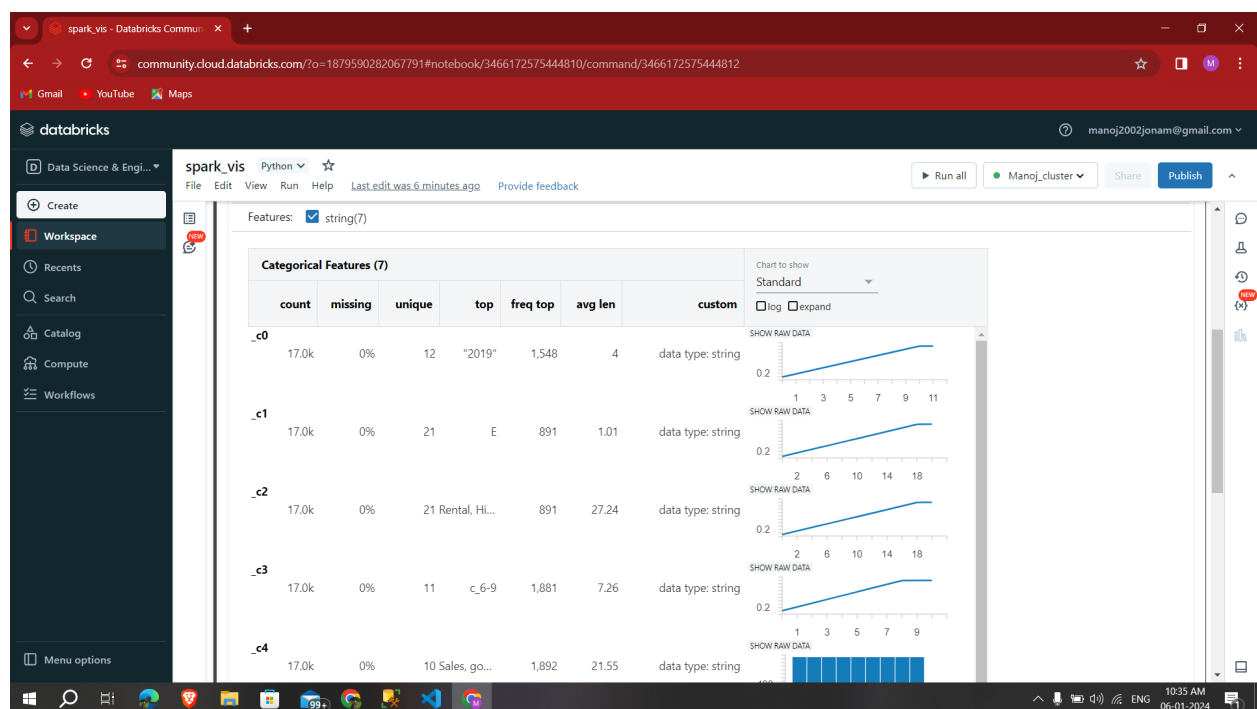
We can also able to **change the colors**



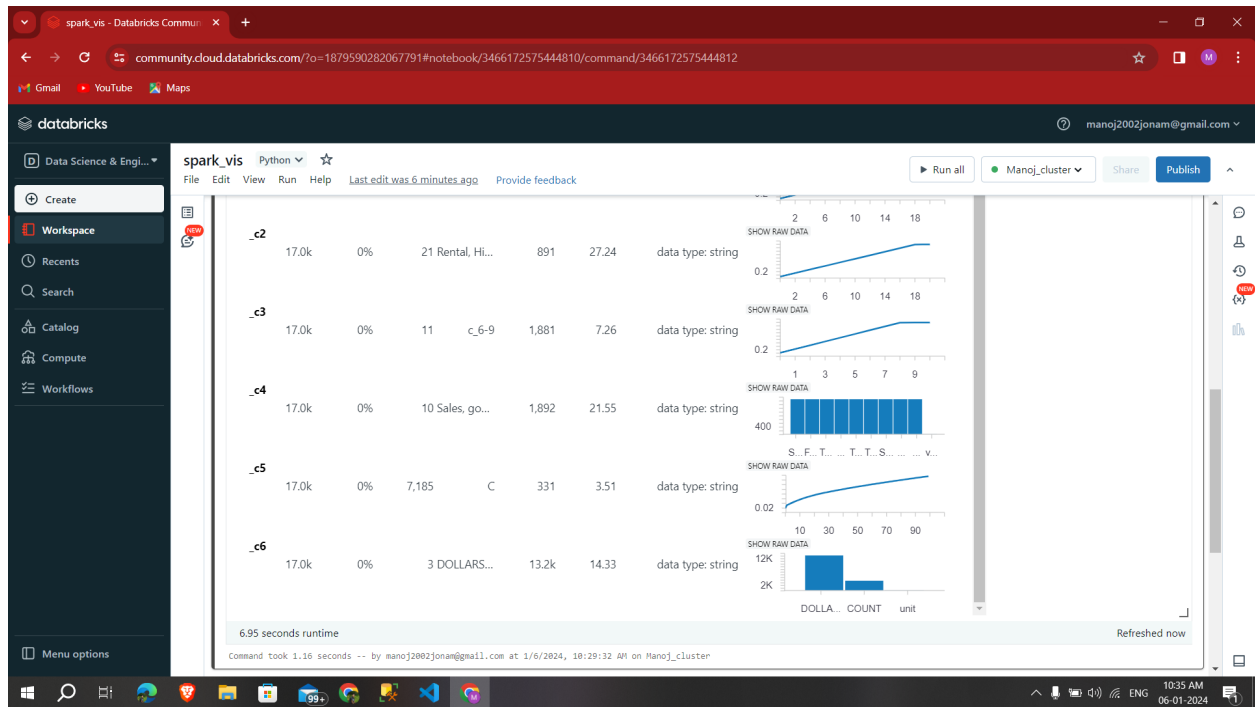
Step 12: Then adding a data profile of that particular table



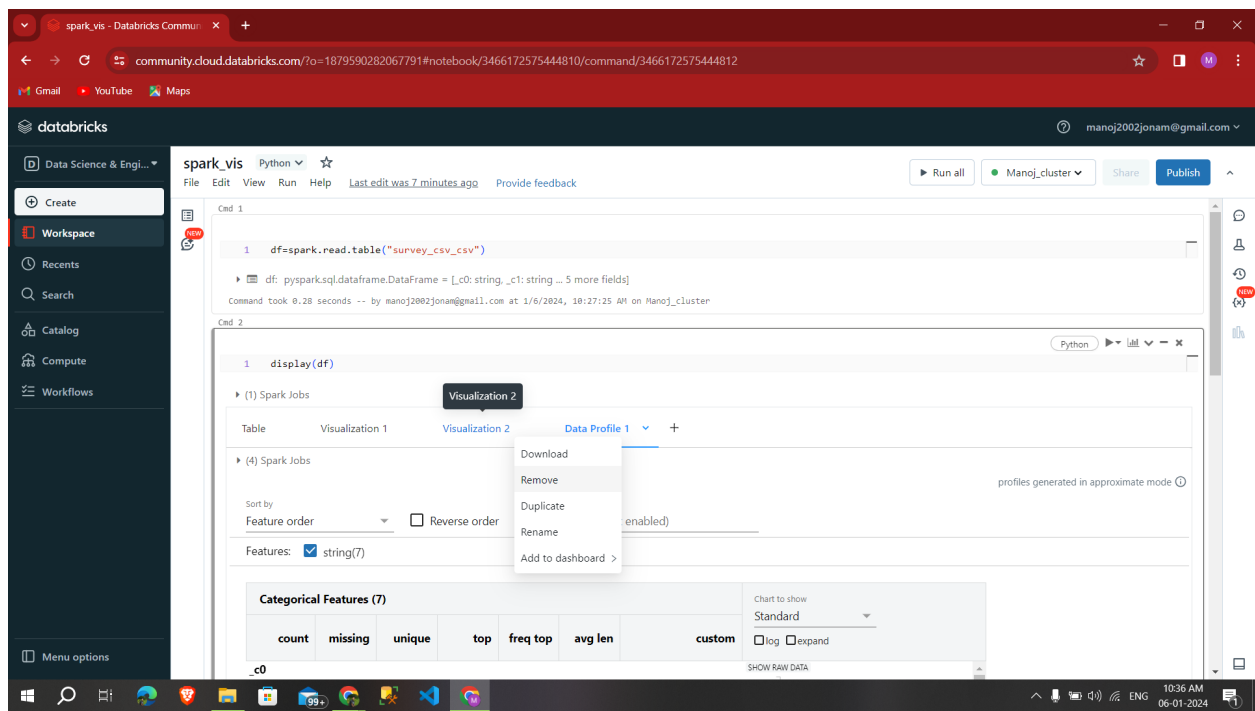
Step 13: The data profile for the table



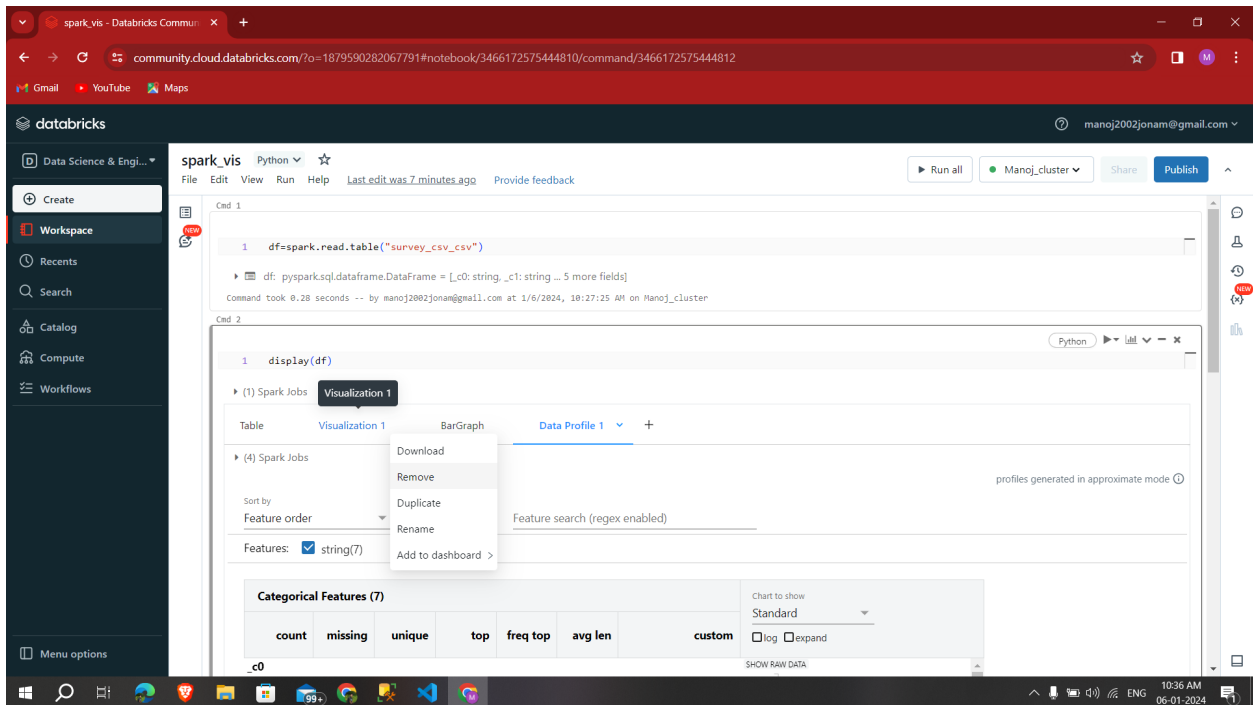
Continuation



Step 14: We can also able to **perform rename,duplicate,remove the visualization or dataprofile**

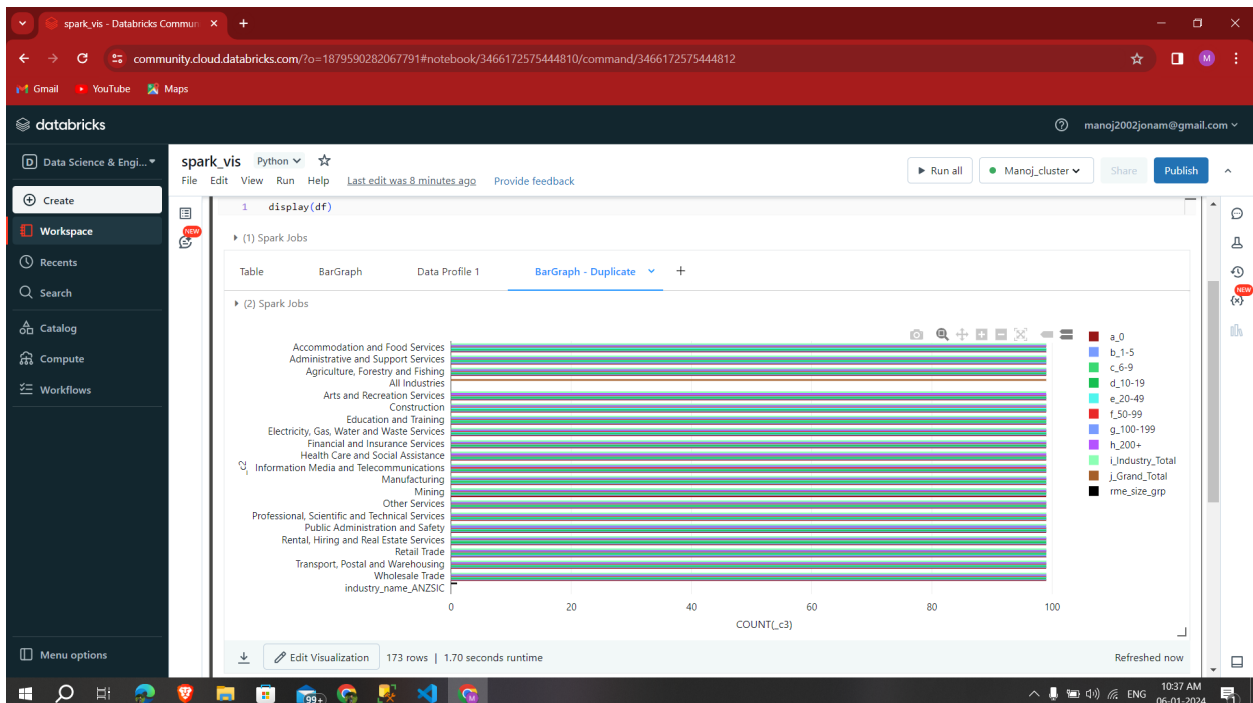


Step 15: I renamed a visualization 2 into BarGarp and removed the visualization 1



The screenshot shows the Databricks interface with a notebook named 'spark_vis'. The notebook contains two commands. Command 1 reads a CSV file into a DataFrame. Command 2 displays the DataFrame. A context menu is open over 'Visualization 1', showing options like Download, Remove, Duplicate, Rename, and Add to dashboard. The visualization area shows a table view with columns for count, missing, unique, top, freq top, avg len, and custom.

Step 16: I duplicated the bar graph and its shown in the next to the data profile



The screenshot shows the Databricks interface with the same notebook. The visualization area now shows a duplicated bar graph next to the data profile. The bar graph displays the distribution of 'COUNT(c3)' across various industries. The legend on the right lists categories like a_0, b_1-5, c_6-9, d_10-19, e_20-49, f_50-99, g_100-199, h_200+, i_Industry_Total, j_Grand_Total, and rme_size_grp.

These are processes of **Create a cluster & Attach the notebook to the cluster and run all commands in the notebook, DataFrame from a Databricks dataset, Visualizations in Databricks notebooks, Rename, duplicate, or remove a visualization or data profile.**

2.Explain the copy activity in Azure data factory.

- ★ **Azure Data Factory** is a cloud-based data integration service provided by Microsoft Azure. It allows you to create, schedule, and manage data pipelines that can move data between supported on-premises and cloud-based data stores.
- ★ **Copy Activity** is a crucial component in Azure Data Factory, responsible for moving data between supported data stores. It allows you to copy data from one location to another.
- ★ **Copy Activity is part of a pipeline**, and it performs the actual movement of data as defined in your data pipelines.
- ★ **Pipelines are a key concept in Azure Data Factory**, representing a logical grouping of activities that together perform a task.
- ★ Pipelines define a set of **data-driven workflows for orchestrating and automating data movement and data transformation**.

Steps :

1. Source and Destination :

- **Source:** The location from which data needs to be copied. This can be a file, database, or any supported data store.
- **Destination:** The destination where the data needs to be copied. This could be another database, file system, or any supported data store.

2.Azure data factory

- Create a data factory service and launch the factory

3.Creating a pipeline:

- After launching the factory click **ingest** → **properties**
- Select **built in copy** → **run once** → **next**
- We have to select the **source data storage account**
 - Select **the storage type** → **click new connection**
 - Select **your subscription** → **source storage account** → **test connection** → **create**
- After selecting the source storage account need to **select the folder** → **next**
- We have to select the **destination data storage account**
 - Select the **storage type** → **click new connection**
 - Select your **subscription** → **source storage account** → **test connection** → **create**
- After selecting the destination storage account **need to select the folder** → **next**
- Give the name for the **pipeline in the setting** → **next**
- In summary check the **details once again** → **next**
- Once **deployment is success click** → **finish**

4. checking the destination file

- Then click on **storage account** → **your destination account** → **container** → **we can able to see the copied files from the source to destination**

5. Monitor and clean up

- Monitor progress using the data factory in the author tab → pipelines → select the pipeline
- We can able to **perform the pipeline again by validate and debug the pipeline**
- **Scheduling** is a crucial aspect that enables you to define when and how frequently your data pipelines should run.
- Clean up resources after completing the process to **avoid ongoing costs**.

- ★ Copy Activity is a tool for orchestrating data movement and transformation across various data stores and services **in a scalable and efficient manner**.
- ★ It is a fundamental building block for building **end-to-end data pipelines in the Azure cloud**.

These are the process of copy activity in Azure data factory.