

ASSIGNMENT 10

NAME – JYOTHI CHANDANA VOLETI
BATCH – DXC-262-ANALYTICS-B12-AZURE
EMPLOYEE DOMAIN –AZURE ANALYTICS
TRAINING UNDER – MANIPAL PRO LEARN
DATE OF SUBMISSION – 15TH JUNE 2022

ROLL NUMBER – DXC-262-AB-1218
COMPANY – DXC TECHNOLOGY
TRAINER NAME – MR. AJAY KUMAR
NO.OF QUESTIONS: 6

INTRODUCTION: To complete today's task we need to create Azure Databricks workspace. To create that we need to follow the steps below.

Step 1: Login to Azure portal and search for Databricks.

The screenshot shows the Microsoft Azure portal homepage. The search bar at the top contains the text "databr". Below the search bar, there are several navigation links: "All", "Services (26)", "Marketplace (9)", "Documentation (4)", "Resources (0)", and "Resource Groups (0)". A sub-section titled "Azure Active Directory (0)" is also visible. The main content area is titled "Services" and lists several services under "See all": Azure Databricks, Azure Database Migration Projects, SQL databases, Azure Database for MySQL servers, Managed databases, Azure Database for MariaDB servers, and Azure Database for PostgreSQL servers. Another section titled "Marketplace" lists Unravel Image for Azure Databricks, Lakehouse Monitor, Trifactor for Azure, and NASH Video Analytics on Azure. On the left sidebar, there are sections for "Create a resource", "More services", "Recent" (with items like dxcdb2, dxc1218, dxcsr12, dxcdb1218, dxcdf12, dxblob1218), and "Name" (with items like dxcdb2 (dx2/dx)). The bottom right corner shows a note: "47 minutes ago" and "Give feedback".

Step 2: Click on Azure Databricks and click on "+create".

The screenshot shows the Azure Databricks service list page. The URL is "portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.Databricks%2Fworkspaces". The top navigation bar includes "Home > Azure Databricks" and "Manipal Pro Learn (manipalazure.onmicrosoft.com)". There are various filtering options: "Subscription == all", "Resource group == all", "Location == all", and a "Create" button. The main content area displays a message: "No azure databricks services to display" with a small icon of three stacked squares. Below this, a sub-message reads: "Unlock insights from all your data and build artificial intelligence (AI) solutions with Azure Databricks, set up your Apache Spark environment in minutes, autoscale, and collaborate on shared projects in an interactive workspace." At the bottom, there are buttons for "Create azure databricks service", "Learn more", and "Give feedback".

Step 3: Fill all the required fields.

The screenshot shows the 'Create an Azure Databricks workspace' page. Under 'Project Details', the subscription is set to 'Azure-DXC262AB12Lab' and the resource group is 'dxc1218'. Under 'Instance Details', the workspace name is 'dxcdb1', located in the 'East US' region with a 'Standard (Apache Spark, Secure with Azure AD)' pricing tier. Navigation buttons at the bottom include 'Review + create' (highlighted in blue), '< Previous', and 'Next : Networking >'.

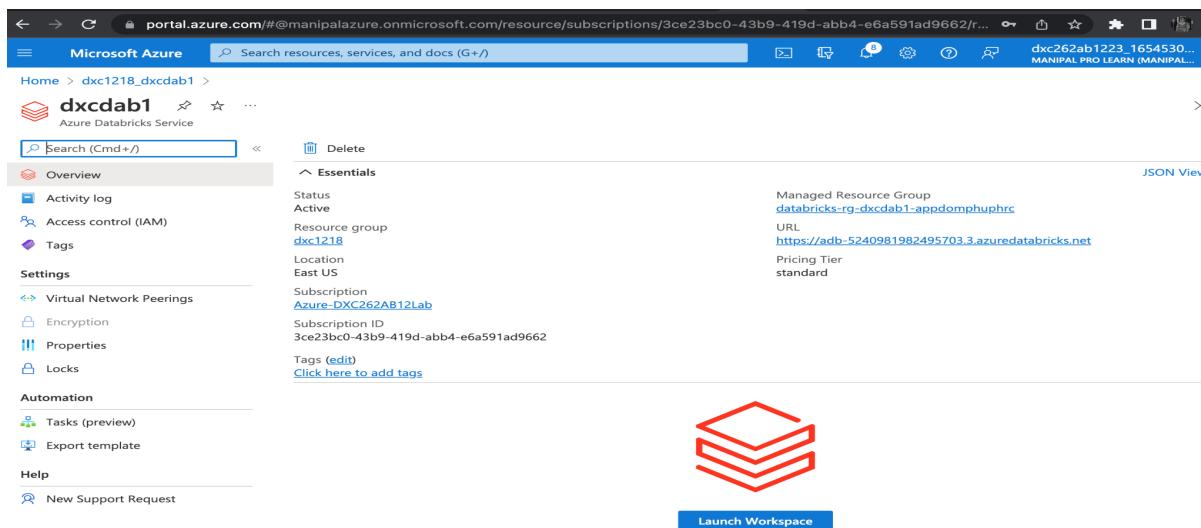
Step 4: Now click on “Review+create” to validate and then click on create at the left bottom to deploy the databricks.

The screenshot shows the 'Create an Azure Databricks workspace' page after validation. A green banner at the top indicates 'Validation Succeeded'. The 'Review + create' tab is selected. Deployment details are listed: Workspace name 'dxcdb1', Subscription 'Azure-DXC262AB12Lab', Resource group 'dxc1218', Region 'East US', and Pricing Tier 'standard'. Under 'Networking', it shows 'Deploy Azure Databricks workspace with Secure Cluster Connectivity (No Public IP)' is 'No'. 'Deploy Azure Databricks workspace in your own Virtual Network (VNet)' is also 'No'. Navigation buttons at the bottom include 'Create' (highlighted in blue), '< Previous', and 'Download a template for automation'.

Step 5: Deployment completed.

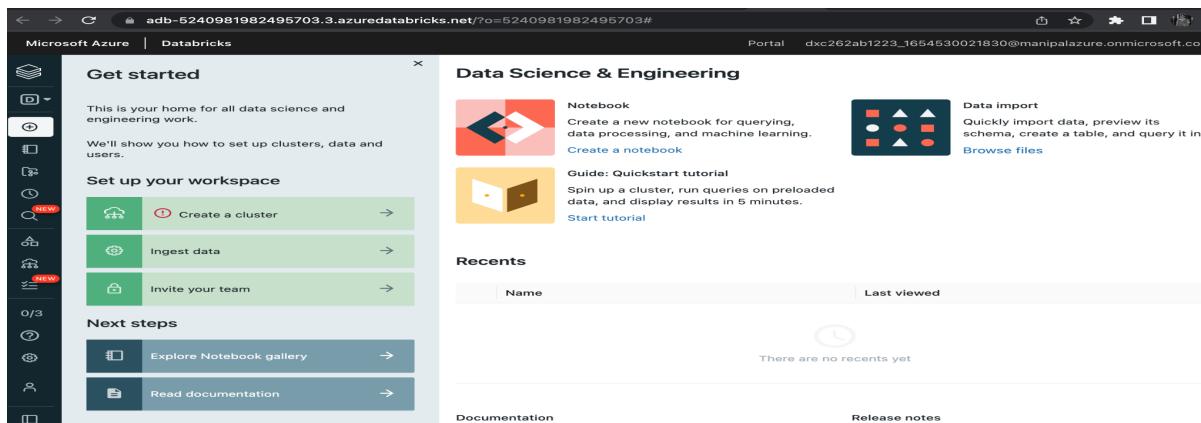
The screenshot shows the 'dxc1218_dxcdb1 | Overview' page. It displays a message: 'Your deployment is complete' with deployment details: name 'dxc1218_dxcdb1', start time '6/12/2022, 12:42:48 PM', subscription 'Azure-DXC262AB12...', and correlation ID '1030b4a1-fa56-47f0-b20f-42107e3337ab'. It includes sections for 'Deployment details' (with a download link) and 'Next steps' (with a 'Go to resource' button). A sidebar on the right offers 'Cost Management', 'Microsoft Defender for Cloud', 'Free Microsoft tutorials', and 'Work with an expert'.

Step 6: Now click on “Go to Resource” and click on “launch workspace”.



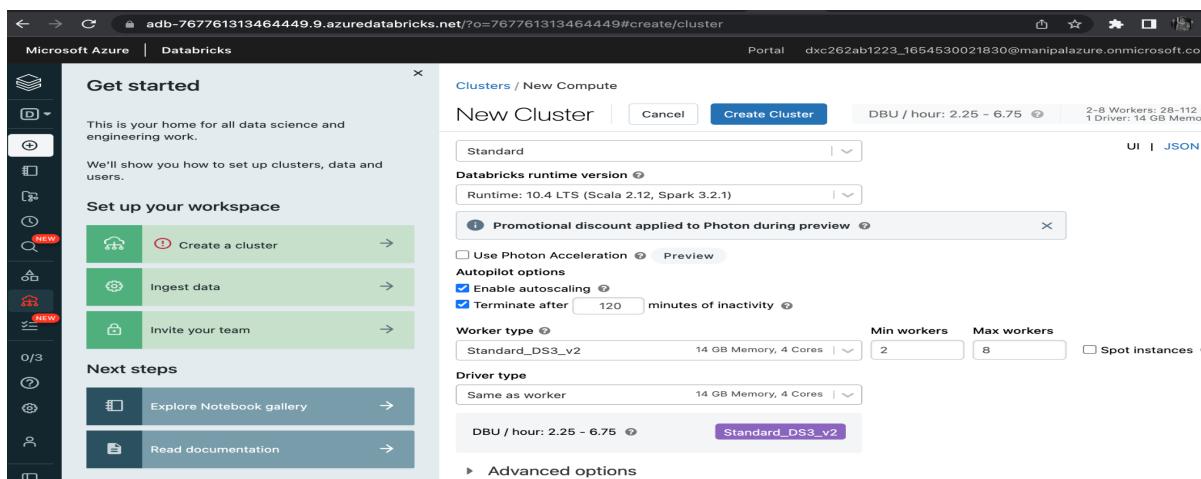
The screenshot shows the Microsoft Azure portal interface for the 'dxc1218_dxcdb1' Azure Databricks Service. The left sidebar includes sections for Overview, Activity log, Access control (IAM), Tags, Settings (Virtual Network Peering, Encryption, Properties, Locks), Automation (Tasks (preview), Export template), Help, and New Support Request. The main content area displays the 'Essentials' section with details such as Status (Active), Resource group (dxc1218), Location (East US), Subscription (Azure-DXC262AB12Lab), Subscription ID (3ce23bc0-43b9-419d-abb4-e6a591ad9662), and Tags (edit). A large red icon of three stacked cubes is centered, and below it is a blue 'Launch Workspace' button.

Step 7: You will sign into Azure Databricks and the Azure Databricks is created.



The screenshot shows the Azure Databricks Data Science & Engineering workspace. On the left, there's a sidebar with a 'Get started' section, 'Set up your workspace' (Create a cluster, Ingest data, Invite your team), and 'Next steps' (Explore Notebook gallery, Read documentation). The main area is titled 'Data Science & Engineering' and contains sections for 'Notebook' (Create a new notebook for querying, data processing, and machine learning), 'Data import' (Quickly import data, preview its schema, create a table, and query it in...), and 'Recents' (which is currently empty). There are also 'Documentation' and 'Release notes' links at the bottom.

Step 8: Now create a cluster using the option compute. Give the necessary credentials for the cluster and click on create.



The screenshot shows the 'Clusters / New Compute' dialog for creating a new cluster. It includes fields for 'Compute type' (Standard), 'Databricks runtime version' (Runtime: 10.4 LTS (Scala 2.12, Spark 3.2.1)), 'Promotional discount applied to Photon during preview' (checkbox), 'Use Photon Acceleration' (checkbox), 'Autopilot options' (checkbox), 'Enable autoscaling' (checkbox), 'Terminate after 120 minutes of inactivity' (checkbox), 'Worker type' (Standard_DS3_v2, 14 GB Memory, 4 Cores), 'Min workers' (2), 'Max workers' (8), 'Driver type' (Same as worker, 14 GB Memory, 4 Cores), 'DBU / hour: 2.25 - 6.75', and 'Spot instances' (checkbox). At the bottom, there are 'Advanced options' and a 'Create Cluster' button.

Step 9: Cluster is created.

The screenshot shows the Microsoft Azure Databricks 'Get started' page. On the left, there's a sidebar with various icons and a progress bar indicating '1/3 Tasks Completed'. The main area displays a 'Get started' section with a green 'Create a cluster' button, a green 'Ingest data' button, and a green 'Invite your team' button. Below this is a 'Set up your workspace' section with a blue 'Explore Notebook gallery' button and a blue 'Read documentation' button. At the top right, the cluster 'cluster1' is listed with a green status indicator and a yellow warning icon. The 'Notebooks (0)' tab is selected. The 'Configuration' tab is also visible. A table below shows 'No notebooks are attached to this cluster'.

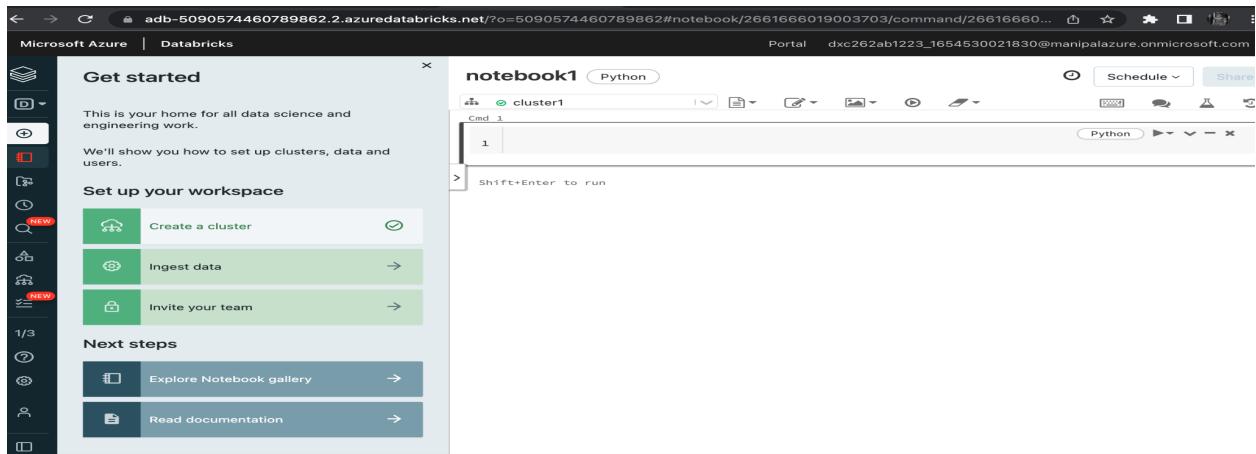
Step 10: Now create 6 notebooks for 6 questions. To create a notebook click on workspace then on create and then click on notebook.

The screenshot shows the Microsoft Azure Databricks workspace page. The left sidebar has a 'Create' menu open, with 'Workspace' selected. The main area shows the 'Notebooks' tab of the 'cluster1' cluster. A context menu is open over the 'Notebooks (0)' tab, with options like 'Create', 'Import', 'Export', 'Permissions', 'Copy Link Address', and 'Sort'. The table below shows 'No notebooks are attached to this cluster'.

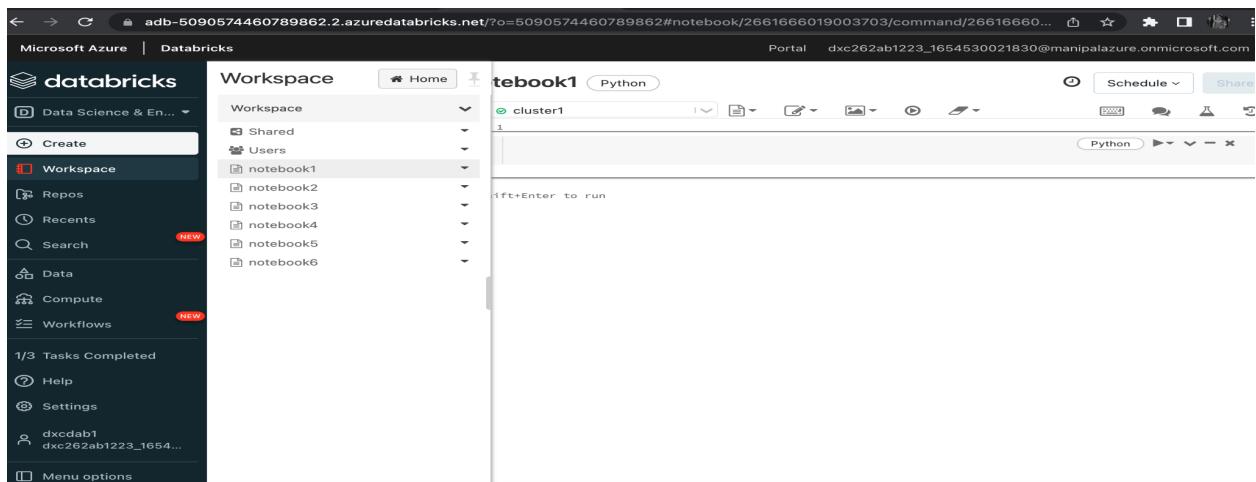
Step 11: Fill all the fields and click on create.

The screenshot shows the Microsoft Azure Databricks 'Create Notebook' dialog box. It has fields for 'Name' (set to 'notebook1'), 'Default Language' (set to 'Python'), and 'Cluster' (set to 'cluster1'). There are 'Cancel' and 'Create' buttons at the bottom. The background shows the workspace page with the 'Notebooks' tab selected.

Step 12: Notebook is created.

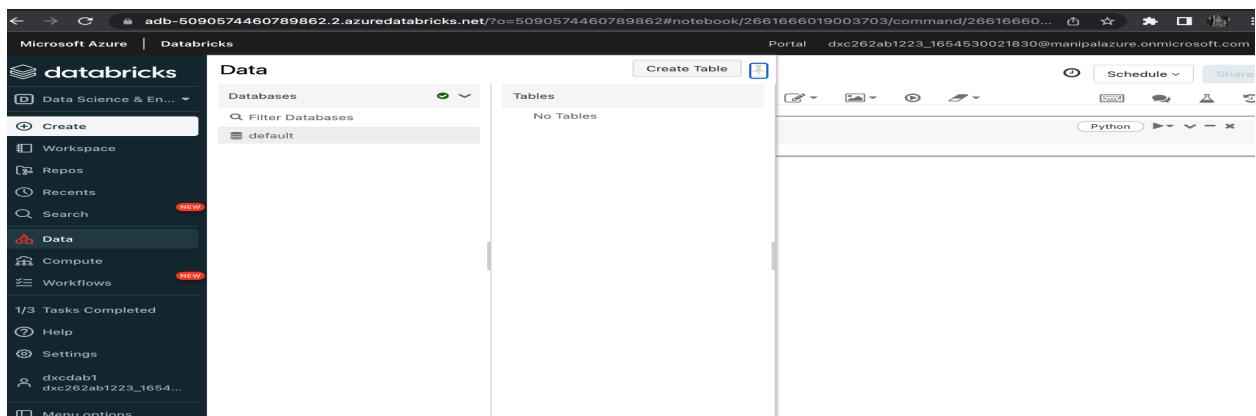


Step 13: Similarly create the remaining 5 notebooks.



All the 6 notebooks are created.

Step 14: Now upload the csv files. To upload the csv files click on data and then click on create table.



Step 15: Now browse and upload the files.

For Case 1: country-codes.csv.

For Case 2: nces330.20.csv.

For Case 3: final_data.csv.

For Case 4: SEntFiN-v1.1.csv.

For Case 5: cancer-death-rates.csv.

For Case 6: inflation-consumer.csv.

The screenshot shows the Databricks UI for creating a new table. The 'DBFS' tab is selected under 'Data source'. The left sidebar has a 'NEW' badge. The main area shows a file browser with 'FileStore' and 'tables' sections. On the right, a list of CSV files is displayed:

- SEntFIN_v1_1.csv
- cancer_death_rates.csv
- country_codes.csv
- final_data.csv
- inflation_consumer.csv
- nces330_20.csv

Now perform the actions that are needed in the following 6 cases.

Case 1. Using archive1.zip file - please ingest data into databricks DBFS path & query the data, redesign columns accordingly using dataframe commands - display with notebooks accordingly

Ans: The file archive1.zip contains a csv file named country-codes.csv.

Ingest data into databricks DBFS path.

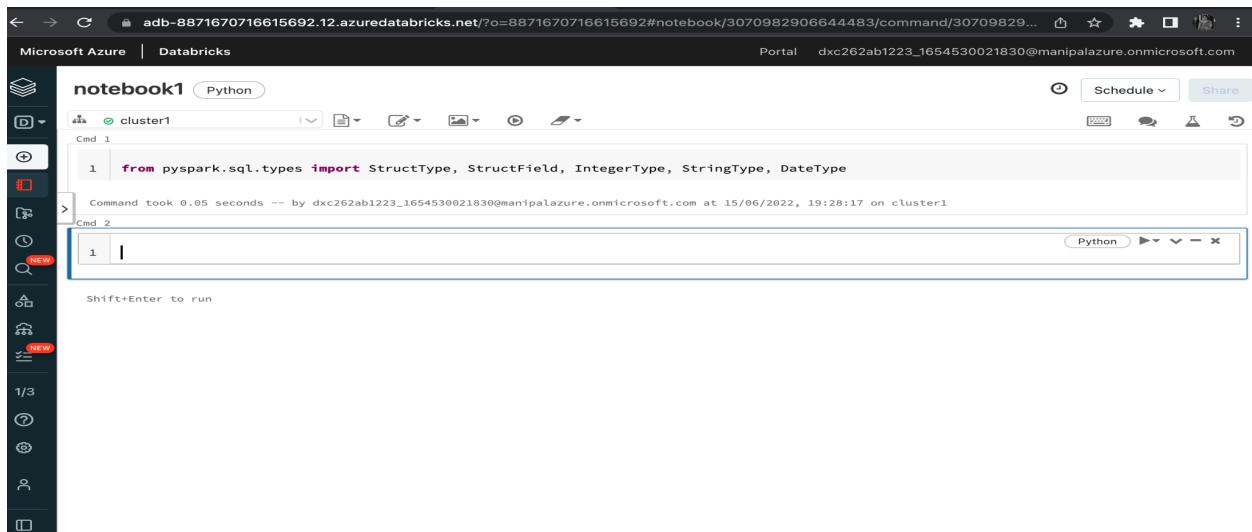
Click on data then on DBFS and click Filestore and tables and there you can view the table country-codes.csv.

The screenshot shows the Databricks UI for creating a new table. The 'DBFS' tab is selected under 'Data source'. The left sidebar has a 'NEW' badge. The main area shows a file browser with 'FileStore' and 'tables' sections. On the right, a list of CSV files is displayed:

- SEntFIN_v1_1.csv
- cancer_death_rates.csv
- country_codes.csv
- final_data.csv
- inflation_consumer.csv
- nces330_20.csv

Step 1: Import the required fields and features from pyspark.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DateType
```



```
notebook1 Python
cluster1
Cmd 1
1 from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DateType
Command took 0.05 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:28:17 on cluster1
Cmd 2
1 |
```

Step 2:

```
country_codes_schema = StructType(fields=[StructField("FIFA", StringType(), False),
                                           StructField("Dial", StringType(), True),
                                           StructField("ISO3166-1-Alpha-3", StringType(), True),
                                           StructField("MARC", StringType(), True),
                                           StructField("is_independent", StringType(), True),
                                           StructField("ISO3166-1-numeric", IntegerType(), True),
                                           StructField("GAUL", IntegerType(), True),
                                           StructField("FIPS", StringType(), True),
                                           StructField("WMO", StringType(), True),
                                           StructField("ISO3166-1-Alpha-2", StringType(), True),
                                           StructField("ITU", StringType(), True),
                                           StructField("IOC", StringType(), True),
                                           StructField("DS", StringType(), True),
                                           StructField("UNTERM Spanish Formal", StringType(), True),
                                           StructField("Global Code", StringType(), True),
                                           StructField("Intermediate Region Code", IntegerType(), True),
                                           StructField("official_name_fr", StringType(), True),
                                           StructField("UNTERM French Short", StringType(), True),
                                           StructField("ISO4217-currency_name", StringType(), True),
                                           StructField("Developed / Developing Countries",
                                                       StringType(), True),
                                           StructField("UNTERM Russian Formal", StringType(), True),
                                           StructField("UNTERM English Short", StringType(), True),
                                           StructField("ISO4217-currency_alphabetic_code", StringType(), True),
                                           StructField("Small Island Developing States
(SIDS)", StringType(), True),
```

```

StructField("UNTERM Spanish Short",StringType(),True),
StructField("ISO4217-currency_numeric_code",IntegerType(),True),
    StructField("UNTERM Chinese Formal",StringType(),True),
    StructField("UNTERM French Formal",StringType(),True),
    StructField("UNTERM Russian Short",StringType(),True),
    StructField("M49",IntegerType(),True),
    StructField("Sub-region Code",IntegerType(),True),
    StructField("Region Code",IntegerType(),True),
    StructField("official_name_ar",StringType(),True),
    StructField("ISO4217-currency_minor_unit",IntegerType(),True),
    StructField("UNTERM Arabic Formal",StringType(),True),
    StructField("UNTERM Chinese Short",StringType(),True),
    StructField("Land Locked Developing Countries
(LLDC)",StringType(),True),
        StructField("Intermediate Region Name",StringType(),True),
        StructField("official_name_es",StringType(),True),
        StructField("UNTERM English Formal",StringType(),True),
        StructField("official_name_cn",StringType(),True),
        StructField("official_name_en",StringType(),True),

StructField("ISO4217-currency_country_name",StringType(),True),
    StructField("Least Developed Countries
(LDC)",StringType(),True),
        StructField("Region Name",StringType(),True),
        StructField("UNTERM Arabic Short",StringType(),True),
        StructField("Sub-region Name",StringType(),True),
        StructField("official_name_ru",StringType(),True),
        StructField("Global Name",StringType(),True),
        StructField("Capital",StringType(),True),
        StructField("Continent",StringType(),True),
        StructField("TLD",StringType(),True),
        StructField("Languages",StringType(),True),
        StructField("Geoname ID",IntegerType(),True),
        StructField("CLDR display name",StringType(),True),
        StructField("EDGAR",StringType(),True)
])

```

```

notebook1 Python
cluster1
26 StructField("ISO4217-currency_minor_unit", IntegerType(), True),
27 StructField("UNTERM Arabic Formal", StringType(), True),
28 StructField("UNTERM Chinese Short", StringType(), True),
29 StructField("Land Locked Developing Countries (LLDC)", StringType(), True),
30 StructField("Intermediate Region Name", StringType(), True),
31 StructField("official_name_es", StringType(), True),
32 StructField("UNTERM English Formal", StringType(), True),
33 StructField("official_name_cn", StringType(), True),
34 StructField("official_name_en", StringType(), True),
35 StructField("ISO4217-currency_country_name", StringType(), True),
36 StructField("Least Developed Countries (LDC)", StringType(), True),
37 StructField("Region Name", StringType(), True),
38 StructField("UNTERM Arabic Short", StringType(), True),
39 StructField("Sub-region Name", StringType(), True),
40 StructField("official_name_ru", StringType(), True),
41 StructField("Global Name", StringType(), True),
42 StructField("Capital", StringType(), True),
43 StructField("Continent", StringType(), True),
44 StructField("TLD", StringType(), True),
45 StructField("Languages", StringType(), True),
46 StructField("Geoname ID", IntegerType(), True),
47 StructField("CLDR display name", StringType(), True),
48 StructField("EDGAR", StringType(), True)
49 ]
50

```

Command took 0.04 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:35:24 on cluster1

Step 3: Click on DBFS and select the file that you have dropped. This will give you the file path and copy that.

```

country_codes_df = spark.read \
.option("header", True) \
.schema(country_codes_schema) \
.csv("/FileStore/tables/country_codes.csv")

```

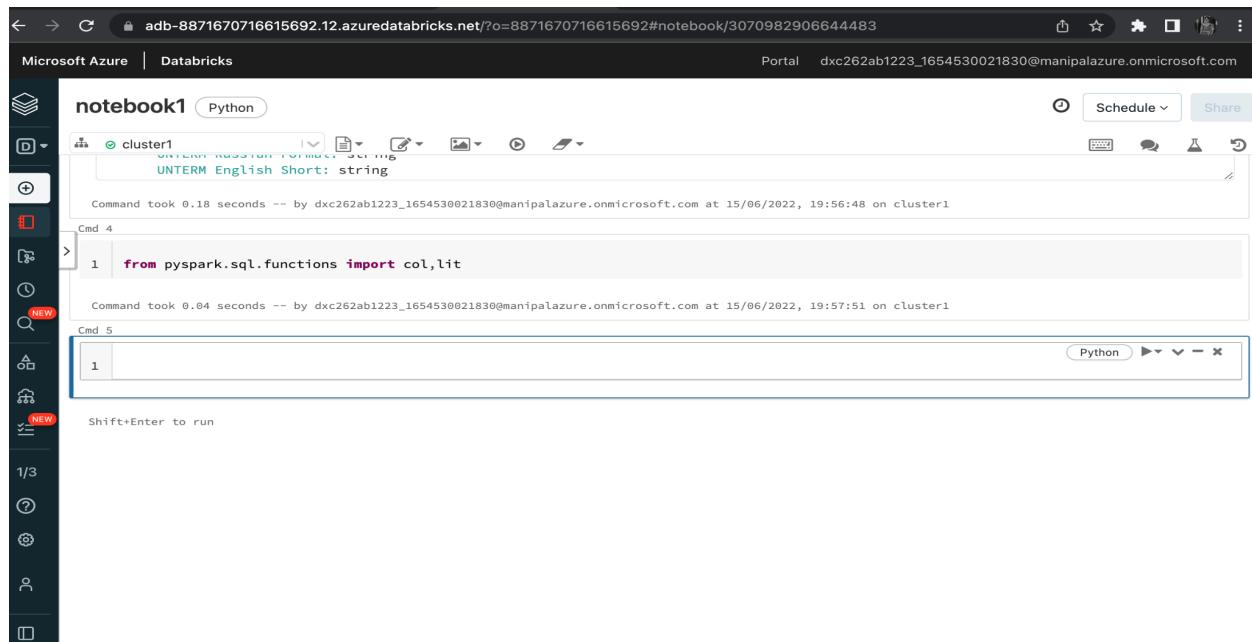
```

notebook1 Python
cluster1
Command took 0.04 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:35:24 on cluster1
Cmd 3
1 country_codes_df = spark.read \
2 .option("header", True) \
3 .schema(country_codes_schema) \
4 .csv("/FileStore/tables/country_codes.csv")
5
country_codes_df: pyspark.sql.dataframe.DataFrame
FIFA: string
DIAL: string
ISO3166-1-Alpha-3: string
MARC: string
is_independent: string
ISO3166-1-numeric: integer
GAUL: integer
FIPS: string
WMO: string
ISO3166-1-Alpha-2: string
ITU: string
IOC: string
DS: string
UNTERM_Spanish_Formal: string
Global_Code: string
Intermediate_Region_Code: integer
official_name_fr: string
UNTERM_French_Short: string
ISO4217-currency_name: string

```

Step 4:

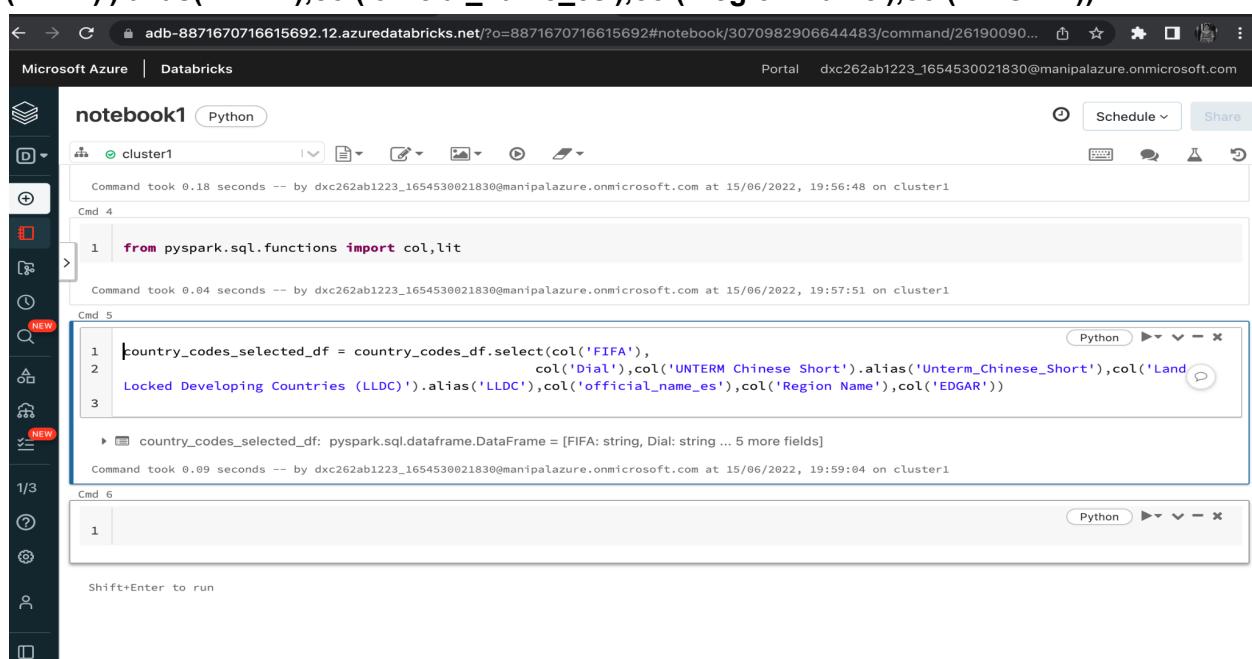
```
from pyspark.sql.functions import col,lit
```



```
notebook1 Python
cluster1
UNTERM English Short: string
Command took 0.18 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:56:48 on cluster1
Cmd 4
1 from pyspark.sql.functions import col,lit
Command took 0.04 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:57:51 on cluster1
Cmd 5
1
Shift+Enter to run
```

Step 5:

```
country_codes_selected_df = country_codes_df.select(col('FIFA'),  
                                                 col('Dial'),col('UNTERM Chinese  
Short').alias('Unterm_Chinese_Short'),col('Land Locked Developing Countries  
(LLDC)').alias('LLDC'),col('official_name_es'),col('Region Name'),col('EDGAR'))
```



```
notebook1 Python
cluster1
Command took 0.18 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:56:48 on cluster1
Cmd 4
1 from pyspark.sql.functions import col,lit
Command took 0.04 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:57:51 on cluster1
Cmd 5
1 country_codes_selected_df = country_codes_df.select(col('FIFA'),  
                                                 col('Dial'),col('UNTERM Chinese Short').alias('Unterm_Chinese_Short'),col('Land  
Locked Developing Countries (LLDC)').alias('LLDC'),col('official_name_es'),col('Region Name'),col('EDGAR'))
2
3
▶ country_codes_selected_df: pyspark.sql.dataframe.DataFrame = [FIFA: string, Dial: string ... 5 more fields]
Command took 0.09 seconds -- by dxc262ab1223_1654530021830@manipalazure.onmicrosoft.com at 15/06/2022, 19:59:04 on cluster1
Cmd 6
1
Shift+Enter to run
```

Step 6:

```
display(country_codes_selected_df)
```

The screenshot shows a Databricks notebook titled "notebook1" running on "cluster1". The code cell contains the command `display(country_codes_selected_df)`. The output of the command is a table titled "Table Data Profile" showing the following data:

| | FIFA | Dial | Unterm_Chinese_Short | LLDC | official_name_es | Region Name |
|---|------|-------|----------------------|------|------------------|-------------|
| 1 | TPE | 886 | null | null | null | null |
| 2 | AFG | 93 | 阿富汗 | x | Afganistán | Asia |
| 3 | ALB | 355 | 阿尔巴尼亚 | null | Albania | Europe |
| 4 | ALG | 213 | 阿尔及利亚 | null | Argelia | Africa |
| 5 | ASA | 1-684 | null | null | Samoa Americana | Oceania |
| 6 | AND | 376 | 安道尔 | null | Andorra | Europe |

Showing all 250 rows.

Case 2. Using archive2.zip file - please ingest data into databricks DBFS path & query the data, redesign columns accordingly using dataframe commands - display with notebooks accordingly

Ans: The file archive2.zip contains a csv file named nces330.20.csv.

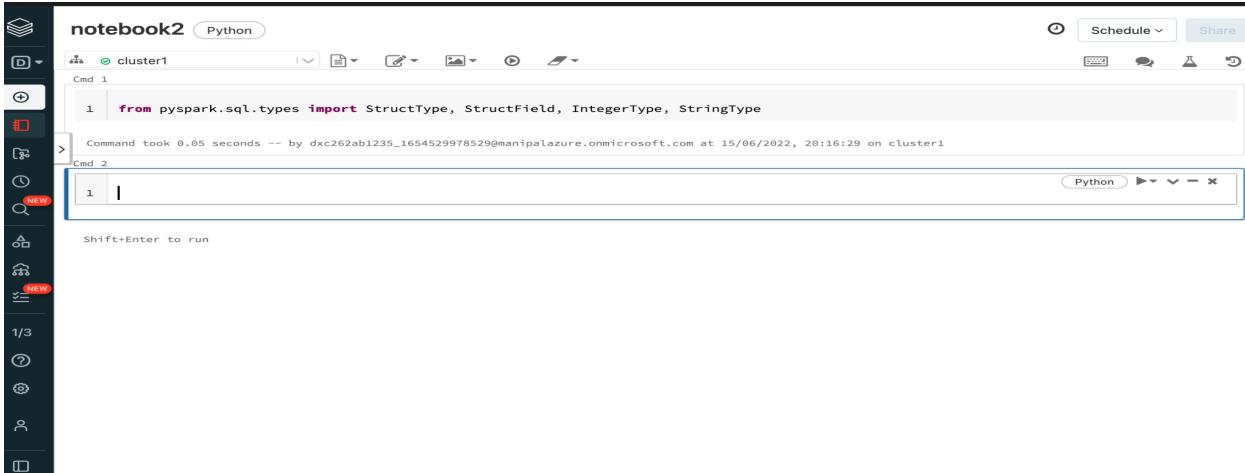
Ingest data into databricks DBFS path.

Click on data then on DBFS and click Filestore and tables and there you can view the table nces330.20.csv.

The screenshot shows the "Create New Table" interface in the Databricks UI. The "Data source" dropdown is set to "DBFS". The "Upload File" and "Other Data Sources" options are also visible. The "Select a file from DBFS" section shows a tree view of the "FileStore" directory, which contains several CSV files: SEntFIN_v1_1.csv, cancer_death_rates.csv, country_codes.csv, final_data.csv, inflation_consumer.csv, and nces330_20.csv. The URL at the bottom of the screen is `/FileStore/tables`.

Step 1: Import the required fields and features from pyspark.

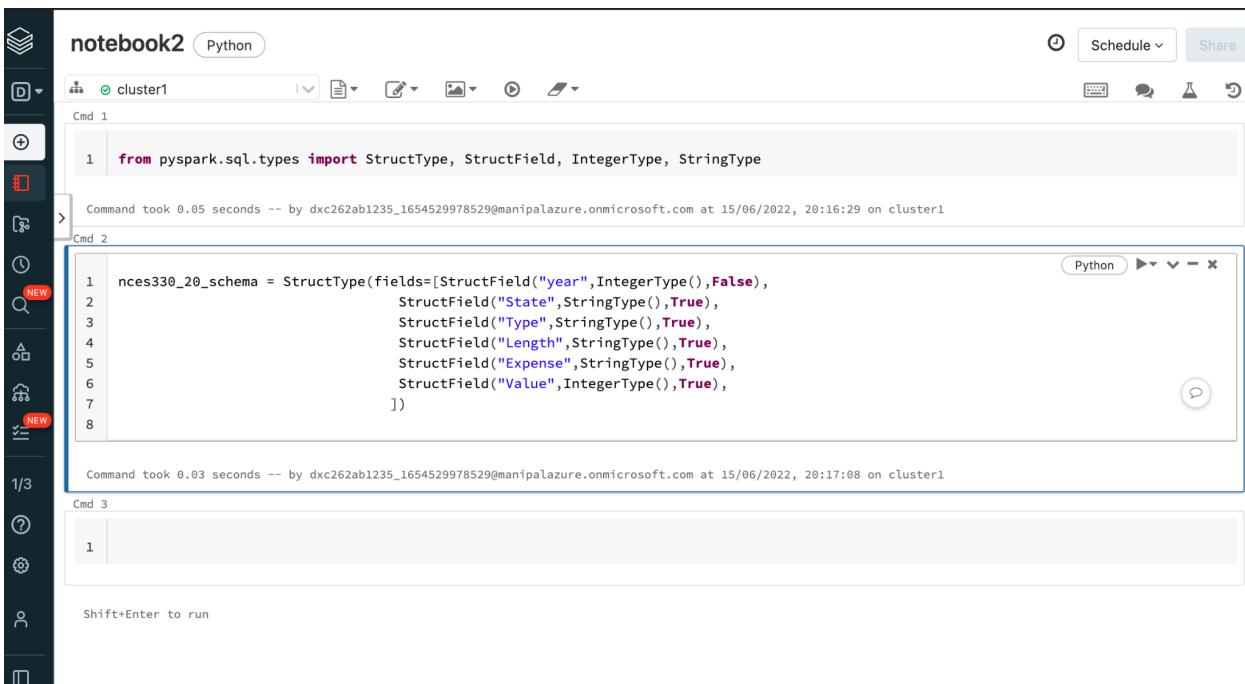
```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
```



A screenshot of a Jupyter Notebook interface titled "notebook2" in Python mode. The sidebar shows a cluster named "cluster1". In the main area, there are two command cells labeled "Cmd 1" and "Cmd 2". Cell "Cmd 1" contains the code "from pyspark.sql.types import StructType, StructField, IntegerType, StringType". The output of this cell is "Command took 0.05 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:16:29 on cluster1". Cell "Cmd 2" is currently empty, indicated by a single character "1" in its input field. A status bar at the bottom right shows "Python" and other icons.

Step 2:

```
nces330_20_schema = StructType(fields=[StructField("year",IntegerType(),False),  
                                      StructField("State",StringType(),True),  
                                      StructField("Type",StringType(),True),  
                                      StructField("Length",StringType(),True),  
                                      StructField("Expense",StringType(),True),  
                                      StructField("Value",IntegerType(),True),  
                                      ])
```



A screenshot of a Jupyter Notebook interface titled "notebook2" in Python mode. The sidebar shows a cluster named "cluster1". In the main area, there are three command cells labeled "Cmd 1", "Cmd 2", and "Cmd 3". Cell "Cmd 1" contains the code "from pyspark.sql.types import StructType, StructField, IntegerType, StringType". The output of this cell is "Command took 0.05 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:16:29 on cluster1". Cell "Cmd 2" contains the code "nces330_20_schema = StructType(fields=[StructField("year",IntegerType(),False), StructField("State",StringType(),True), StructField("Type",StringType(),True), StructField("Length",StringType(),True), StructField("Expense",StringType(),True), StructField("Value",IntegerType(),True),])" and has a small circular icon with a question mark next to it. The output of this cell is "Command took 0.03 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:17:08 on cluster1". Cell "Cmd 3" is currently empty, indicated by a single character "1" in its input field. A status bar at the bottom right shows "Python" and other icons.

Step 3: Click on DBFS and select the file that you have dropped. This will give you the file path and copy that.

```
nces330_20_df = spark.read \
.option("header", True) \
.schema(nces330_20_schema) \
.csv("/FileStore/tables/nces330_20.csv")
```

The screenshot shows a Jupyter Notebook titled "notebook2" running on a Python kernel. The code cell contains the following Python code:

```
nces330_20_df = spark.read \
.option("header", True) \
.schema(nces330_20_schema) \
.csv("/FileStore/tables/nces330_20.csv")
```

The output of the command is displayed in the next cell, showing the schema of the DataFrame:

```
1 nces330_20_df: pyspark.sql.dataframe.DataFrame
  year: integer
  State: string
  Type: string
  Length: string
  Expense: string
  Value: integer
```

Below the schema, a message indicates the command took 1.69 seconds to execute.

Step 4:

```
from pyspark.sql.functions import col,lit
```

The screenshot shows a Jupyter Notebook titled "notebook2" running on a Python kernel. The code cell contains the following Python code:

```
nces330_20_df = spark.read \
.option("header", True) \
.schema(nces330_20_schema) \
.csv("/FileStore/tables/nces330_20.csv")
```

The output of the command is displayed in the next cell, showing the schema of the DataFrame:

```
1 nces330_20_df: pyspark.sql.dataframe.DataFrame
  year: integer
  State: string
  Type: string
  Length: string
  Expense: string
  Value: integer
```

Below the schema, a message indicates the command took 1.69 seconds to execute.

The code cell below contains the following Python code:

```
1 from pyspark.sql.functions import col,lit
```

The output of this command is displayed in the next cell, showing the message "Command took 0.04 seconds".

Step 5:

```
nces330_20_selected_df = nces330_20_df.select(col('Year'),  
                                              col('State'),col('Expense'))
```

The screenshot shows a Jupyter Notebook titled "notebook2" running on a Python kernel connected to a cluster named "cluster1". The notebook interface includes a sidebar with various icons for file operations like new files, copy, paste, and search. The main area contains two command cells labeled "Cmd 5" and "Cmd 6".

Cmd 5 contains the following code:

```
1 nces330_20_selected_df = nces330_20_df.select(col('Year'),  
                                              col('State'),col('Expense'))
```

Cmd 6 shows the output of the previous command:

```
1  
Shift+Enter to run
```

Below the notebook area, there is a message indicating the command took 0.09 seconds to execute.

Step 6:

```
display(nces330_20_selected_df)
```

The screenshot shows a Jupyter Notebook titled "notebook2" running on a Python kernel connected to a cluster named "cluster1". The sidebar and command cells are identical to the previous screenshot. The output of Cmd 6 is displayed in a "Table" view, showing the first 1000 rows of the selected DataFrame.

The table has four columns: "Year", "State", and "Expense". The data is as follows:

| | Year | State | Expense |
|---|------|---------|--------------|
| 1 | 2013 | Alabama | Fees/Tuition |
| 2 | 2013 | Alabama | Room/Board |
| 3 | 2013 | Alabama | Fees/Tuition |
| 4 | 2013 | Alabama | Fees/Tuition |
| 5 | 2013 | Alabama | Room/Board |
| 6 | 2013 | Alabama | Fees/Tuition |

A note at the bottom of the table states: "Truncated results, showing first 1000 rows. Click to re-execute with maximum result limits."

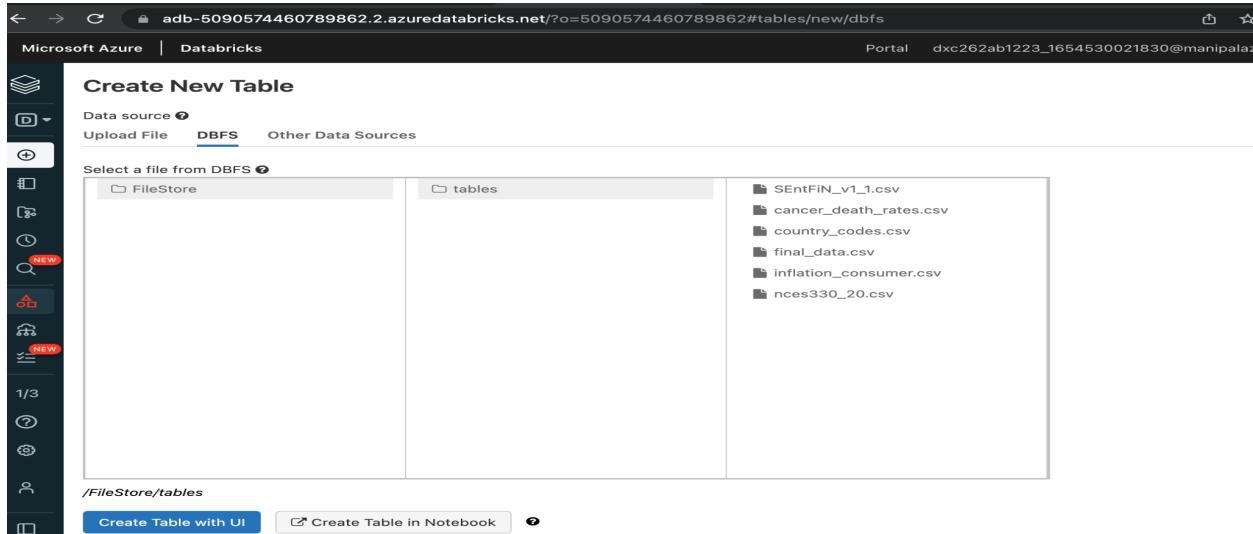
Below the table, another command cell "Cmd 7" is visible with the message "Shift+Enter to run".

Case 3. Using archive3.zip file - please ingest data into databricks DBFS path & query the data, redesign columns accordingly using dataframe commands - display with notebooks accordingly

Ans: The file archive3.zip contains a csv file named final_data.csv.

Ingest data into databricks DBFS path.

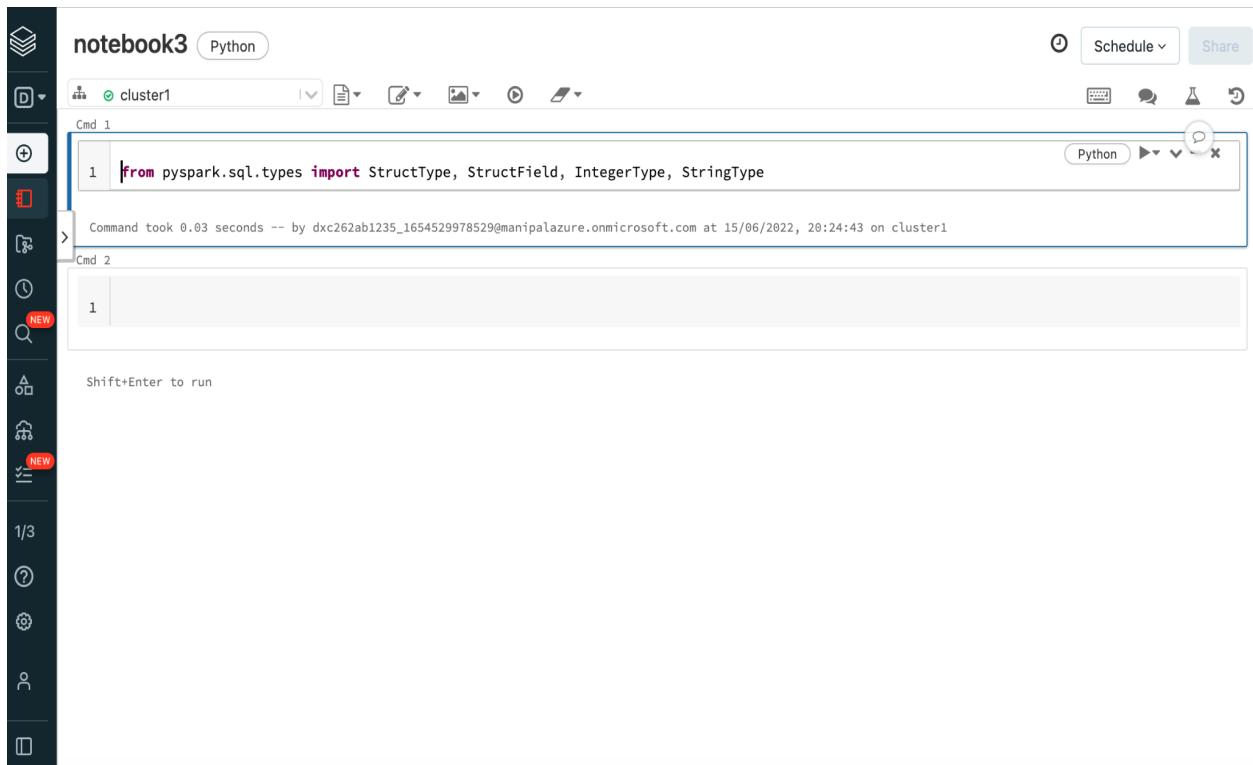
Click on data then on DBFS and click Filestore and tables and there you can view the table final_data.csv.



The screenshot shows the 'Create New Table' interface in Databricks. The 'Data source' dropdown is set to 'DBFS'. Under 'Select a file from DBFS', the 'FileStore' tab is selected, showing a list of CSV files: SEntFIN_v1_1.csv, cancer_death_rates.csv, country_codes.csv, final_data.csv, inflation_consumer.csv, and nces330_20.csv. Below the list, the URL /FileStore/tables is visible. At the bottom, there are two buttons: 'Create Table with UI' (highlighted in blue) and 'Create Table in Notebook'.

Step 1: Import the required fields and features from pyspark.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
```



The screenshot shows the Databricks notebook3 interface in Python mode. In the first command cell (Cmd 1), the user has typed the import statement: `from pyspark.sql.types import StructType, StructField, IntegerType, StringType`. The cell has been run, and the output shows the command took 0.03 seconds. In the second command cell (Cmd 2), the user has typed the number '1' and is preparing to run the cell by pressing Shift+Enter. The sidebar on the left shows various cluster and workspace options.

Step 2:

```
final_data_schema = StructType(fields=[StructField("tweet_text",StringType(),False),
                                         StructField("emotion_in_tweet_is_directed_at",StringType(),True),
```

```
                                         StructField("is_there_an_emotion_directed_at_a_brand_or_product",StringType(),True),
                                         ])
```

The screenshot shows a Jupyter Notebook titled "notebook3" running on a Python kernel. The sidebar on the left has a "NEW" button highlighted. The main area contains three command cells. Cell 1 imports the StructType and StructField classes from pyspark.sql.types. Cell 2 defines the final_data_schema variable with the provided schema. Cell 3 is empty and has a note "Shift+Enter to run". The status bar at the bottom indicates the command took 0.03 seconds and was run on cluster1.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
final_data_schema = StructType(fields=[StructField("tweet_text",StringType(),False),
                                         StructField("emotion_in_tweet_is_directed_at",StringType(),True),
                                         StructField("is_there_an_emotion_directed_at_a_brand_or_product",StringType(),True),
                                         ])

```

Step 3: Click on DBFS and select the file that you have dropped. This will give you the file path and copy that.

```
final_data_df = spark.read \
.option("header" , True) \
.schema(final_data_schema) \
.csv("/FileStore/tables/final_data.csv")
```

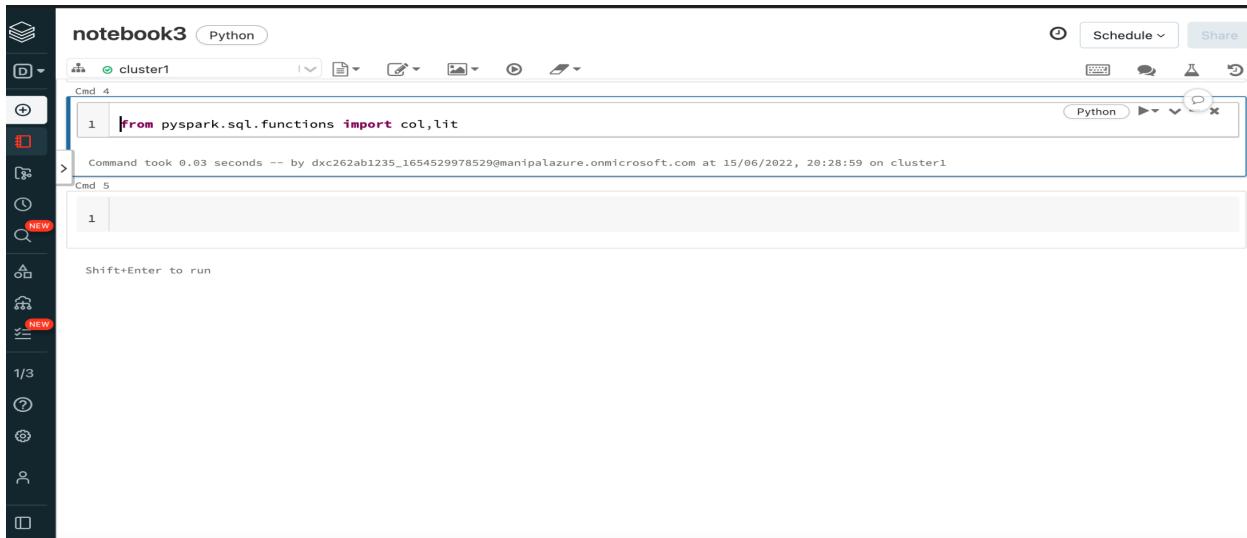
The screenshot shows a Jupyter Notebook titled "notebook3" running on a Python kernel. The sidebar on the left has a "NEW" button highlighted. The main area contains four command cells. Cells 1-3 define the final_data_schema and create the final_data_df DataFrame. Cell 4 displays the schema of final_data_df, which includes columns tweet_text, emotion_in_tweet_is_directed_at, and is_there_an_emotion_directed_at_a_brand_or_product, all of type string. Cell 5 is empty. The status bar at the bottom indicates the command took 0.16 seconds and was run on cluster1.

```
final_data_schema = StructType(fields=[StructField("tweet_text",StringType(),False),
                                         StructField("emotion_in_tweet_is_directed_at",StringType(),True),
                                         StructField("is_there_an_emotion_directed_at_a_brand_or_product",StringType(),True),
                                         ])
final_data_df = spark.read \
.option("header" , True) \
.schema(final_data_schema) \
.csv("/FileStore/tables/final_data.csv")
final_data_df: pyspark.sql.dataframe.DataFrame
    tweet_text: string
    emotion_in_tweet_is_directed_at: string
    is_there_an_emotion_directed_at_a_brand_or_product: string

```

Step 4:

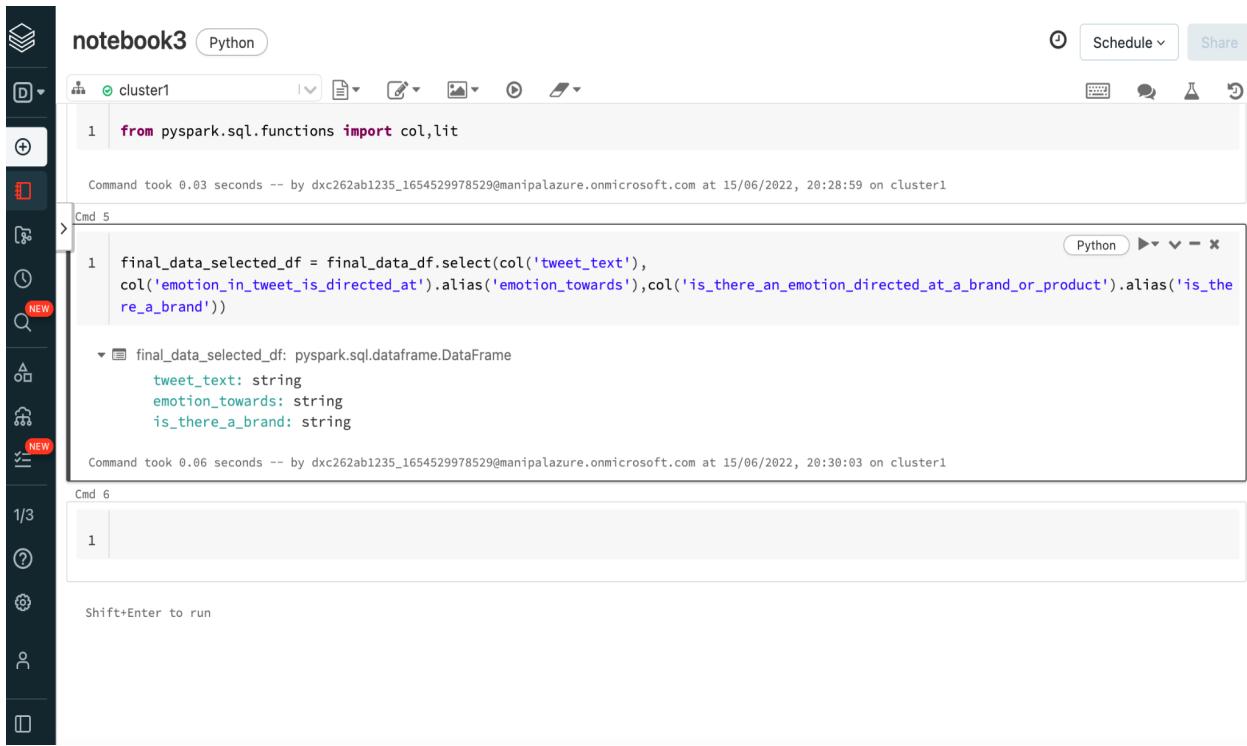
```
from pyspark.sql.functions import col,lit
```



The screenshot shows a Jupyter Notebook interface titled "notebook3" running on a Python kernel. The sidebar on the left includes icons for file operations like "New", "Save", and "Run". The main area has a toolbar with buttons for file, edit, cell, and help. A command cell labeled "Cmd 4" contains the code "from pyspark.sql.functions import col,lit". Below the code, the output shows the command took 0.03 seconds and was run by a specific user on cluster1 at a specific date and time. A second command cell, "Cmd 5", is present but empty.

Step 5:

```
final_data_selected_df = final_data_df.select(col('tweet_text'),  
col('emotion_in_tweet_is_directed_at').alias('emotion_towards'),col('is_there_an_emotion  
_directed_at_a_brand_or_product').alias('is_there_a_brand'))
```



The screenshot shows a Jupyter Notebook interface titled "notebook3" running on a Python kernel. The sidebar on the left includes icons for file operations like "New", "Save", and "Run". The main area has a toolbar with buttons for file, edit, cell, and help. A command cell labeled "Cmd 4" contains the code "from pyspark.sql.functions import col,lit". Below the code, the output shows the command took 0.03 seconds and was run by a specific user on cluster1 at a specific date and time. A second command cell, "Cmd 5", contains the code for defining the "final_data_selected_df" DataFrame. The output shows the definition of the DataFrame and its schema, which includes columns for "tweet_text" (string), "emotion_towards" (string), and "is_there_a_brand" (string). A third command cell, "Cmd 6", is present but empty.

Step 6:
display(final_data_selected_df)

| tweet_text | emotion_towards | is_spark |
|---|--------------------|----------|
| 1 @wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW. | iPhone | No |
| 2 @jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW | iPad or iPhone App | Yes |
| 3 @swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW. | iPad | Yes |
| 4 @sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw | iPad or iPhone App | No |
| 5 @sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress) | Google | Yes |
| 6 @teachntech00 New iPad Apps For #SpeechTherapy And Communication Are Showcased At The #SXSW Conference http://ht.ly/49n4M #ear #edchat #asd | null | No |

Truncated results, showing first 1000 rows.
Click to re-execute with maximum result limits.

Command took 0.61 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:31:03 on cluster1

Case 4. Using archive4.zip file - please ingest data into databricks DBFS path & query the data, redesign columns accordingly using dataframe commands - display with notebooks accordingly

Ans: The file archive4.zip contains a csv file named SEntFiN-v1.1.csv.

Ingest data into databricks DBFS path.

Click on data then on DBFS and click Filestore and tables and there you can view the table SEntFiN-v1.1.csv.

Create New Table

Data source

Upload File **DBFS** Other Data Sources

Select a file from DBFS

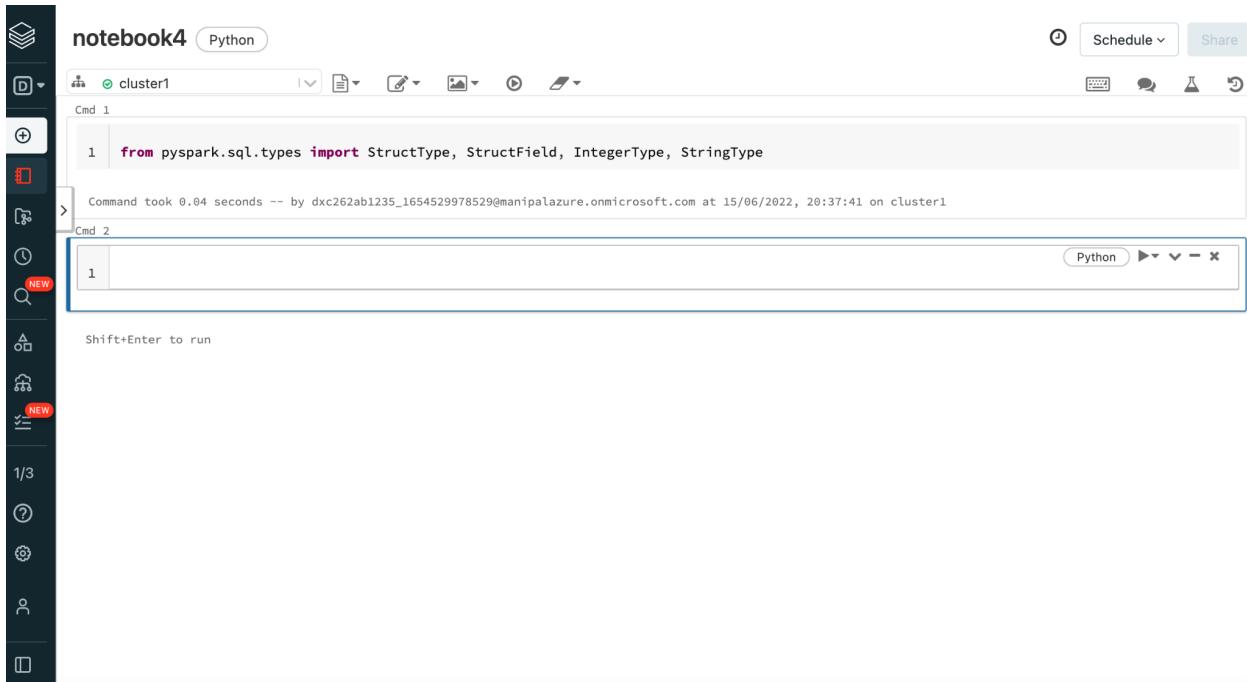
/FileStore/tables

Create Table with UI Create Table in Notebook

- FileStore
- tables
- SEntFiN_v1_1.csv
- cancer_death_rates.csv
- country_codes.csv
- final_data.csv
- inflation_consumer.csv
- nces330_20.csv

Step 1: Import the required fields and features from pyspark.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
```

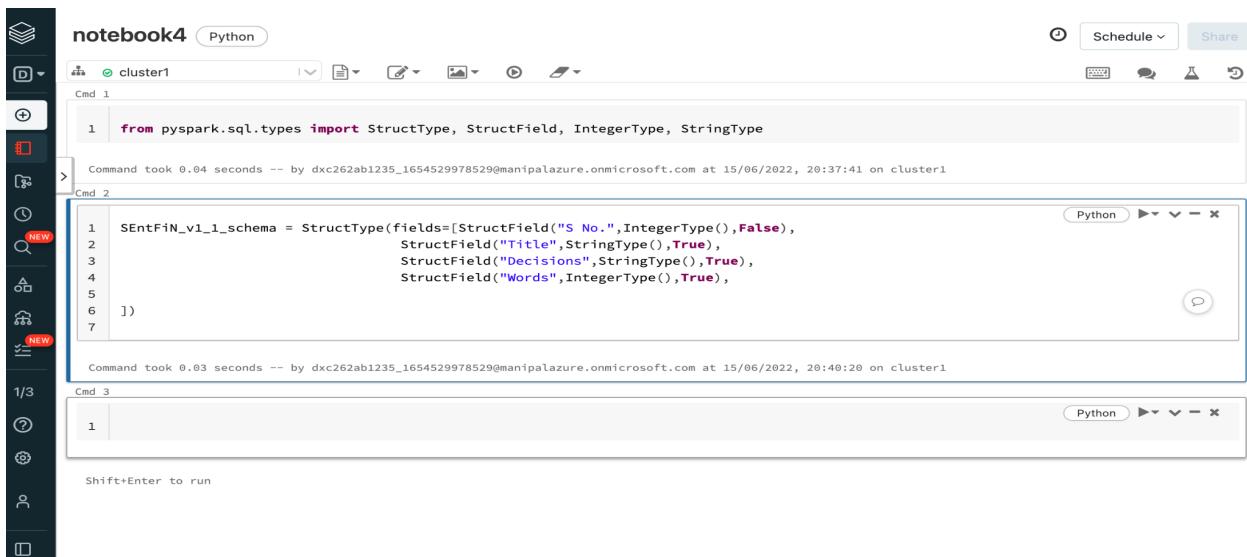


The screenshot shows a Jupyter Notebook interface titled "notebook4" in Python mode. The sidebar on the left includes icons for file operations like "New", "Save", and "Run". The main area has two command cells. Cell 1 contains the code "from pyspark.sql.types import StructType, StructField, IntegerType, StringType". Cell 2 is currently empty, indicated by the number "1" in its header. A status bar at the bottom says "Shift+Enter to run". The top right corner has buttons for "Schedule" and "Share".

Step 2:

```
SEntFiN_v1_1_schema = StructType(fields=[StructField("S No.",IntegerType(),False),  
                                         StructField("Title",StringType(),True),  
                                         StructField("Decisions",StringType(),True),  
                                         StructField("Words",IntegerType(),True),  
                                         ])
```

I)



This screenshot shows the continuation of the Jupyter Notebook session. The sidebar and top navigation are identical to the previous screenshot. The main area now includes a third command cell, Cell 3, which contains the definition of the schema variable. The code is identical to the one in Step 2. The status bar at the bottom of the notebook interface indicates "Shift+Enter to run".

Step 3: Click on DBFS and select the file that you have dropped. This will give you the file path and copy that.

```
SEntFiN_v1_1_df = spark.read \
.option("header", True) \
.schema(SEntFiN_v1_1_schema) \
.csv("/FileStore/tables/SEntFiN_v1_1-1.csv")
```

The screenshot shows a Jupyter Notebook interface with a sidebar containing various icons. The main area has a title bar "notebook4 Python". Below it is a toolbar with icons for cluster selection, file operations, and sharing. A status bar at the bottom indicates "Command took 0.03 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:40:20 on cluster1". The code cell "Cmd 3" contains the following Python code:

```
1 SEntFiN_v1_1_df = spark.read \
2 .option("header", True) \
3 .schema(SEntFiN_v1_1_schema) \
4 .csv("/FileStore/tables/SEntFiN_v1_1-1.csv")
```

Below the code cell, the output shows the DataFrame schema:

```
SEntFiN_v1_1_df: pyspark.sql.dataframe.DataFrame
  S_No.: integer
  Title: string
  Decisions: string
  Words: integer
```

Another status bar below the schema indicates "Command took 0.14 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:43:51 on cluster1". The notebook shows 1/3 cells run.

Step 4:

```
from pyspark.sql.functions import col,lit
```

The screenshot shows a Jupyter Notebook interface with a sidebar containing various icons. The main area has a title bar "notebook4 Python". Below it is a toolbar with icons for cluster selection, file operations, and sharing. A status bar at the bottom indicates "Command took 0.14 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:43:51 on cluster1". The code cell "Cmd 4" contains the following Python code:

```
1 from pyspark.sql.functions import col,lit
```

Below the code cell, the output indicates "Command took 0.02 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:44:44 on cluster1". The notebook shows 1/3 cells run.

Step 5:

```
SEntFiN_v1_1_selected_df = SEntFiN_v1_1_df.select(col('Decisions'),  
col('Title'),col('Words'))
```

The screenshot shows a Jupyter Notebook interface with a single cell containing the Python code from Step 5. The code defines a DataFrame named `SEntFiN_v1_1_selected_df` by selecting specific columns from `SEntFiN_v1_1_df`. The output pane below the cell shows the resulting DataFrame structure with columns `Decisions`, `Title`, and `Words`. A command took 0.06 seconds to run.

Step 6:

```
display(SEntFiN_v1_1_df)
```

The screenshot shows a Jupyter Notebook interface with a single cell containing the Python code from Step 6. The code uses the `display` function to show the contents of `SEntFiN_v1_1_df`. The output pane displays a table titled "Data Profile" with two columns: `S No.` and `Title`. Below the table, a truncated results message indicates that only the first 1000 rows are shown. The full table data is as follows:

| S No. | Title | Decisions |
|-------|--|-------------------------------|
| 1 | SpiceJet to issue 6.4 crore warrants to promoters | {"SpiceJet": "neutral"} |
| 2 | MMTC Q2 net loss at Rs 10.4 crore | {"MMTC": "neutral"} |
| 3 | Mid-cap funds can deliver more, stay put: Experts | {"Mid-cap funds": "positive"} |
| 4 | Mid caps now turn into market darlings | {"Mid caps": "positive"} |
| 5 | Market seeing patience, if not conviction: Prakash Diwan | {"Market": "neutral"} |
| 6 | Infosys: Will the strong volume growth sustain? | {"Infosys": "neutral"} |

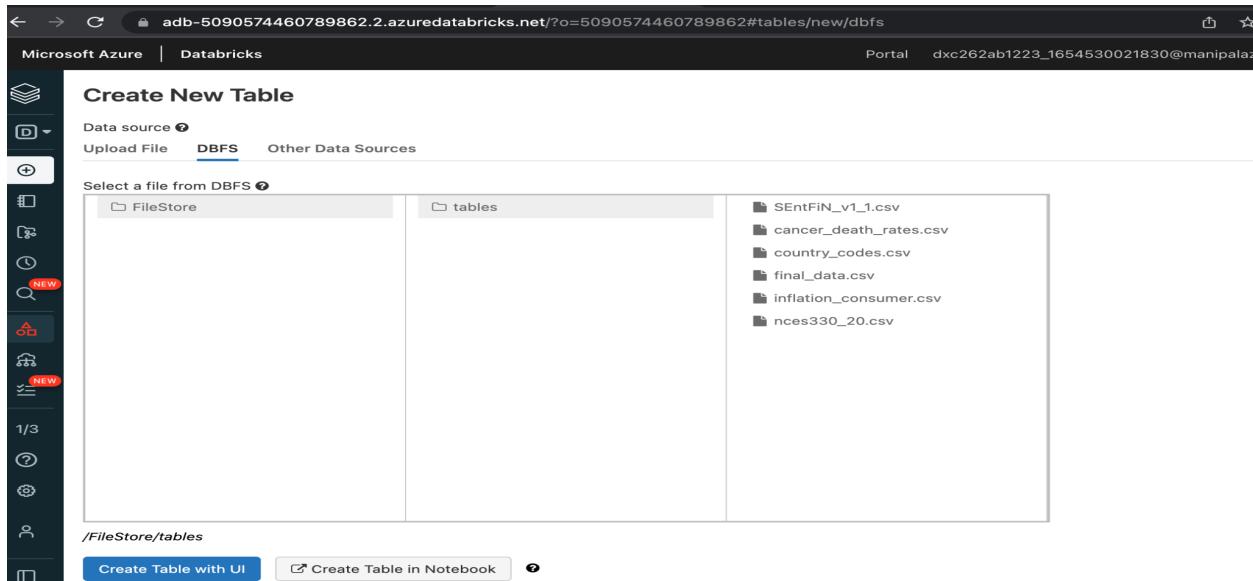
A note at the bottom of the table states: "Truncated results, showing first 1000 rows. Click to re-execute with maximum result limits." A command took 0.40 seconds to run.

Case 5. Using archive5.zip file - please ingest data into databricks DBFS path & query the data, redesign columns accordingly using dataframe commands - display with notebooks accordingly

Ans: The file archive5.zip contains a csv file named cancer-death-rates.csv.

Ingest data into databricks DBFS path.

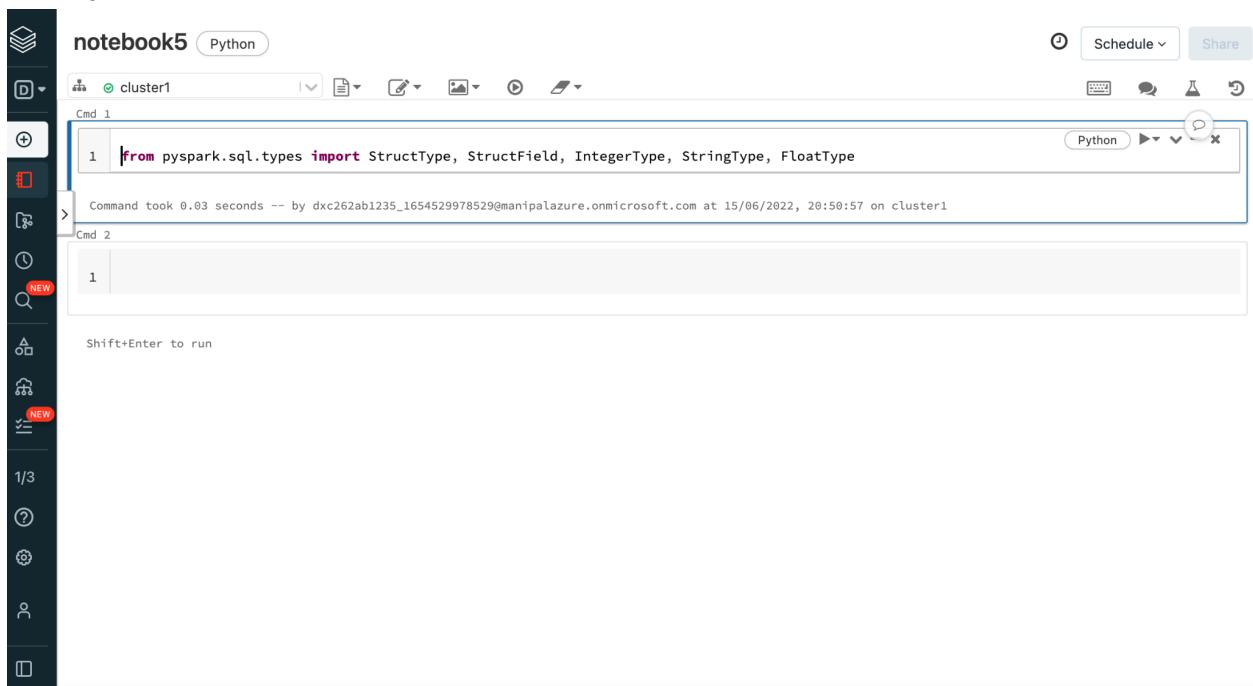
Click on data then on DBFS and click Filestore and tables and there you can view the table cancer-death-rates.csv.



The screenshot shows the 'Create New Table' interface in Databricks. The 'DBFS' tab is selected under 'Data source'. In the 'Select a file from DBFS' section, the '/FileStore/tables' directory is expanded, showing files like SEntFiN_v1_1.csv, cancer_death_rates.csv, country_codes.csv, final_data.csv, inflation_consumer.csv, and nces330_20.csv. At the bottom, there are two buttons: 'Create Table with UI' and 'Create Table in Notebook'.

Step 1: Import the required fields and features from pyspark.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType
```



The screenshot shows the 'notebook5' interface in Databricks. A code cell in 'cluster1' contains the import statement 'from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType'. The cell has run successfully, with the message 'Command took 0.03 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:50:57 on cluster1'. Below the cell is another empty cell labeled 'Cmd 2' with the number '1' in it. The interface includes a sidebar with various icons and a toolbar at the top.

Step 2:

```
cancer_death_rates_schema =  
StructType(fields=[StructField("Entity",StringType(),False),  
    StructField("Code",StringType(),True),  
    StructField("Year",IntegerType(),True),  
    StructField("Deaths - Neoplasms - Sex: Both - Age: Age-standardized  
(Rate)",FloatType(),True)],
```

])

The screenshot shows a Jupyter Notebook interface with a sidebar containing icons for file operations like New, Open, Save, and Run. The main area has three code cells labeled Cmd 1, Cmd 2, and Cmd 3. Cmd 1 contains the import statement for StructType. Cmd 2 contains the schema definition. Cmd 3 is empty. The status bar at the bottom indicates "Shift+Enter to run".

```
notebook5 Python  
cluster1  
Cmd 1  
1 from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType  
Command took 0.03 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:50:57 on cluster1  
Cmd 2  
1 cancer_death_rates_schema = StructType(fields=[StructField("Entity",StringType(),False),  
2     StructField("Code",StringType(),True),  
3     StructField("Year",IntegerType(),True),  
4     StructField("Deaths - Neoplasms - Sex: Both - Age: Age-standardized (Rate)",FloatType(),True),  
5     ])  
6  
7  
Command took 0.02 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:51:38 on cluster1  
Cmd 3  
1  
Shift+Enter to run
```

Step 3: Click on DBFS and select the file that you have dropped. This will give you the file path and copy that.

```
Cancer_death_rates_df = spark.read \  
.option("header" , True) \  
.schema(cancer_death_rates_schema) \  
.csv("/FileStore/tables/cancer_death_rates.csv")
```

The screenshot shows a Jupyter Notebook interface with a sidebar containing icons for file operations like New, Open, Save, and Run. The main area has four code cells labeled Cmd 3, Cmd 4, Cmd 5, and Cmd 6. Cmd 3 contains the code to read the CSV file. Cmd 4 shows the resulting DataFrame structure with columns Entity, Code, Year, and Deaths - Neoplasms - Sex: Both - Age: Age-standardized (Rate). Cmd 5 and Cmd 6 are empty. The status bar at the bottom indicates "Shift+Enter to run".

```
notebook5 Python  
cluster1  
Cmd 3  
1 Cancer_death_rates_df = spark.read \  
2 .option("header" , True) \  
3 .schema(cancer_death_rates_schema) \  
4 .csv("/FileStore/tables/cancer_death_rates.csv")  
Cancer_death_rates_df: pyspark.sql.dataframe.DataFrame  
  Entity: string  
  Code: string  
  Year: integer  
  Deaths - Neoplasms - Sex: Both - Age: Age-standardized (Rate): float  
Command took 0.13 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:53:03 on cluster1  
Cmd 4  
1  
Shift+Enter to run
```

Step 4:

```
from pyspark.sql.functions import col,lit
```



The screenshot shows a Jupyter Notebook titled "notebook5" running on a Python cluster. In the code editor, the following command is being run:

```
1 Cancer_death_rates_df = spark.read \
2 .option("header", True) \
3 .schema(Cancer_death_rates_schema) \
4 .csv("FileStore/tables/cancer_death_rates.csv")
```

The output shows the DataFrame structure:

```
Cancer_death_rates_df: pyspark.sql.dataframe.DataFrame
  Entity: string
  Code: string
  Year: integer
  Deaths - Neoplasms - Sex: Both - Age: Age-standardized (Rate): float
```

Execution details: Command took 0.13 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:53:03 on cluster1

In the next cell (Cmd 4), the command `from pyspark.sql.functions import col,lit` is typed.

```
1 from pyspark.sql.functions import col,lit
```

Execution details: Command took 0.03 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:54:10 on cluster1

Step 5:

```
Cancer_death_rates_selected_df = Cancer_death_rates_df.select(col('Entity'),  
col('Year'),col('Deaths - Neoplasms - Sex: Both - Age: Age-standardized  
(Rate)').alias('Deaths'))
```



The screenshot shows the continuation of the notebook. The previous command has been run, and the current cell (Cmd 5) contains the following code:

```
1 Cancer_death_rates_selected_df = Cancer_death_rates_df.select(col('Entity'), col('Year'),col('Deaths - Neoplasms - Sex: Both - Age:  
Age-standardized (Rate)').alias('Deaths'))
```

The output shows the selected DataFrame structure:

```
Cancer_death_rates_selected_df: pyspark.sql.dataframe.DataFrame
  Entity: string
  Year: integer
  Deaths: float
```

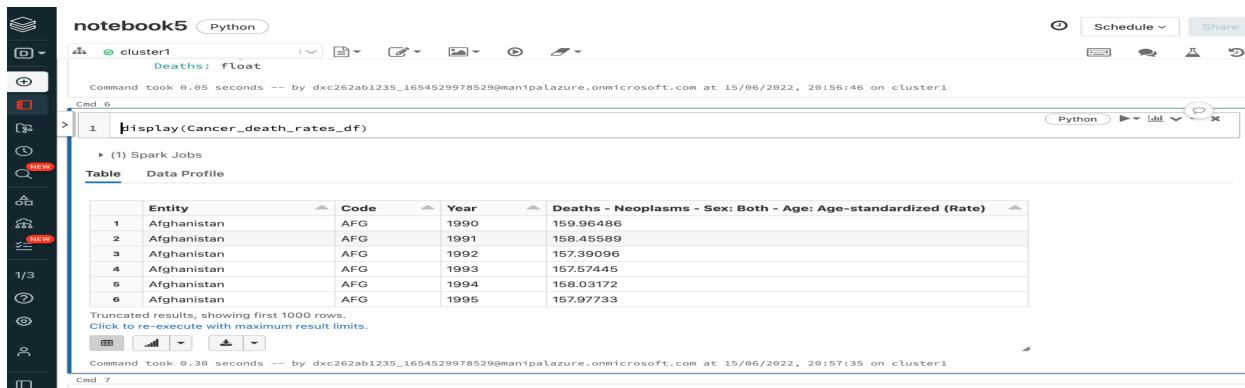
Execution details: Command took 0.05 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:56:46 on cluster1

In the next cell (Cmd 6), the command `display(Cancer_death_rates_df)` is typed.

```
1 display(Cancer_death_rates_df)
```

Step 6:

```
display(Cancer_death_rates_df)
```



The screenshot shows the execution of the `display` command. The output displays a table of data for Afghanistan:

| Entity | Code | Year | Deaths - Neoplasms - Sex: Both - Age: Age-standardized (Rate) |
|---------------|------|------|---|
| 1 Afghanistan | AFG | 1990 | 159.96486 |
| 2 Afghanistan | AFG | 1991 | 158.45589 |
| 3 Afghanistan | AFG | 1992 | 157.39096 |
| 4 Afghanistan | AFG | 1993 | 157.57445 |
| 5 Afghanistan | AFG | 1994 | 158.03172 |
| 6 Afghanistan | AFG | 1995 | 157.97733 |

Truncated results, showing first 1000 rows.
Click to re-execute with maximum result limits.

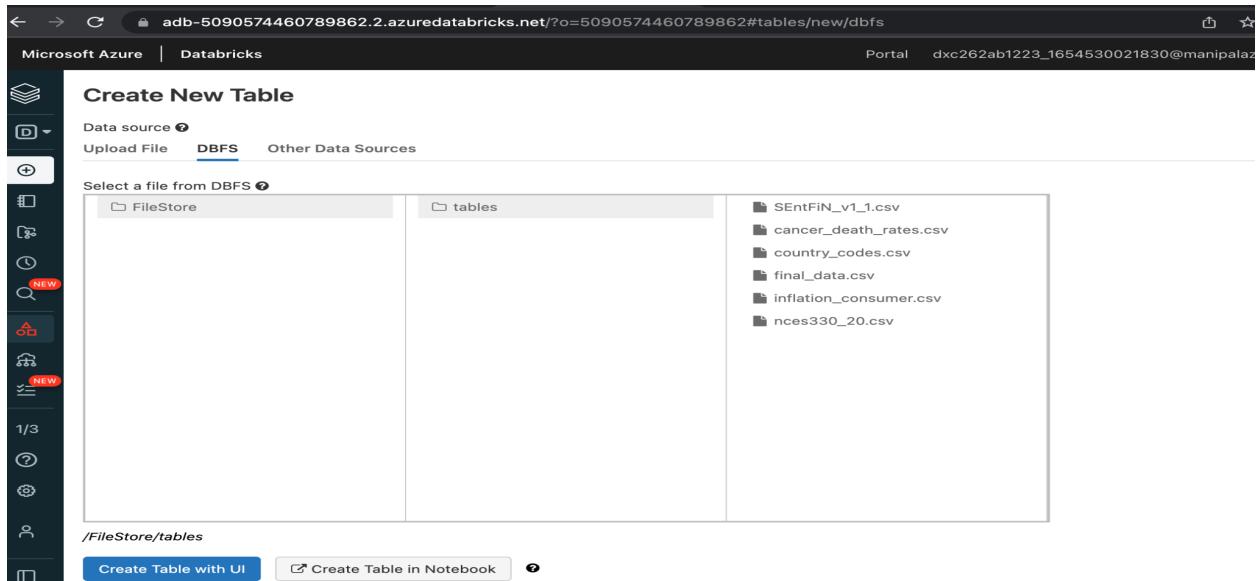
Execution details: Command took 0.30 seconds -- by dxc262ab1235_1654529978529@manipalazure.onmicrosoft.com at 15/06/2022, 20:57:35 on cluster1

Case 6. Using archive6.zip file - please ingest data into databricks DBFS path & query the data, redesign columns accordingly using dataframe commands - display with notebooks accordingly

Ans: The file archive6.zip contains a csv file named inflation-consumer.csv.

Ingest data into databricks DBFS path.

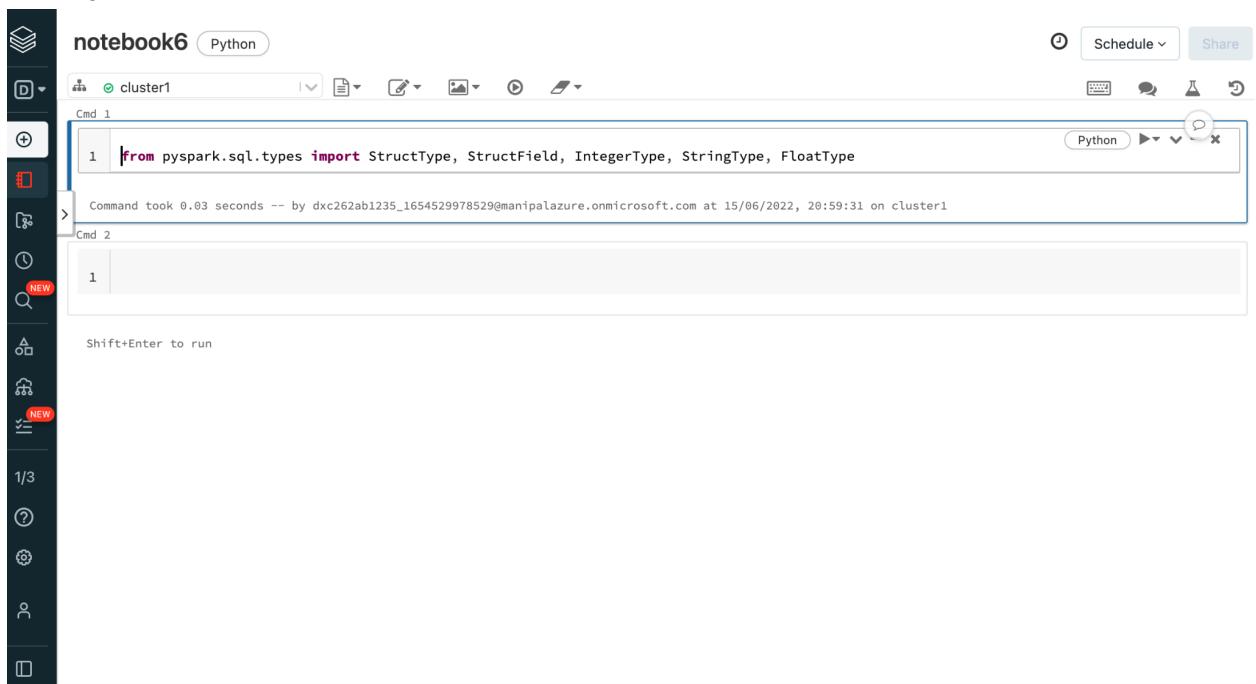
Click on data then on DBFS and click Filestore and tables and there you can view the table inflation-consumer.csv.



The screenshot shows the 'Create New Table' interface in Databricks. The 'DBFS' tab is selected under 'Data source'. A sidebar on the left shows various icons for clusters, notebooks, and data. The main area shows a tree view of 'FileStore' and 'tables' directories. Inside 'tables', several CSV files are listed: SEntFiN_v1_1.csv, cancer_death_rates.csv, country_codes.csv, final_data.csv, inflation_consumer.csv, and nces330_20.csv. At the bottom, there are buttons for 'Create Table with UI' and 'Create Table in Notebook'.

Step 1: Import the required fields and features from pyspark.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType
```

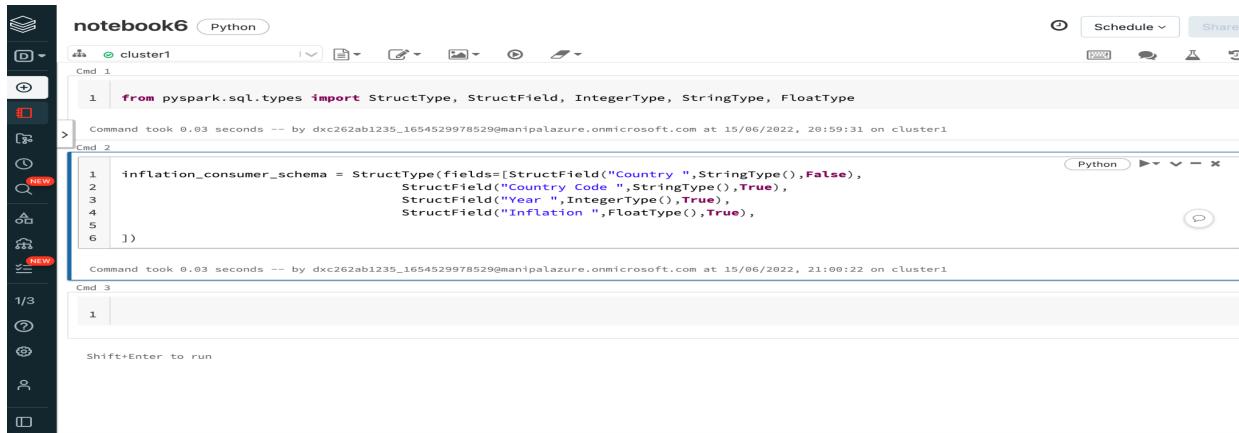


The screenshot shows a Databricks notebook titled 'notebook6' in Python mode. The sidebar on the left includes icons for clusters, notebooks, and data. The main area has a toolbar with 'cluster1', file operations, and a search bar. Below is a command cell labeled 'Cmd 1' containing the Python code: 'from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType'. The output of this command is shown in a box below, indicating it took 0.03 seconds to run on cluster1. There is also a second, empty command cell labeled 'Cmd 2'.

Step 2:

```
inflation_consumer_schema = StructType(fields=[StructField("Country", StringType(), False),  
                                              StructField("Country Code", StringType(), True),  
                                              StructField("Year", IntegerType(), True),  
                                              StructField("Inflation", FloatType(), True),
```

I)

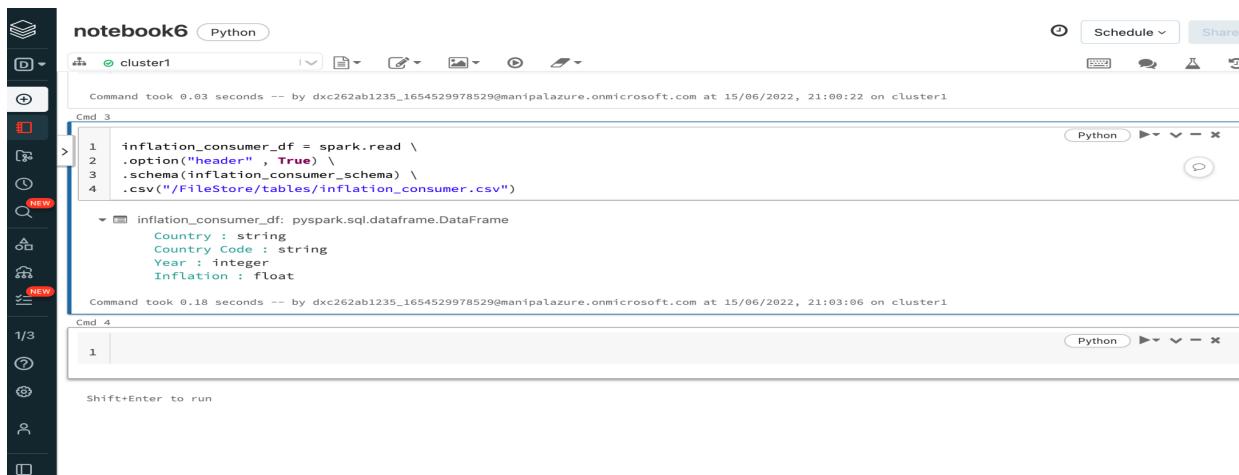


The screenshot shows a Jupyter Notebook titled "notebook6" in Python mode. It contains three command cells. The first cell imports necessary types from pyspark.sql.types. The second cell defines the schema variable with the provided code. The third cell is empty and has a placeholder "Shift+Enter to run". The sidebar on the left shows a tree view of the notebook's structure.

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, FloatType  
  
inflation_consumer_schema = StructType(fields=[StructField("Country", StringType(), False),  
                                              StructField("Country Code", StringType(), True),  
                                              StructField("Year", IntegerType(), True),  
                                              StructField("Inflation", FloatType(), True),  
                                              ])  
  
Shift+Enter to run
```

Step 3: Click on DBFS and select the file that you have dropped. This will give you the file path and copy that.

```
inflation_consumer_df = spark.read \  
.option("header", True) \  
.schema(inflation_consumer_schema) \  
.csv("/FileStore/tables/inflation_consumer.csv")
```

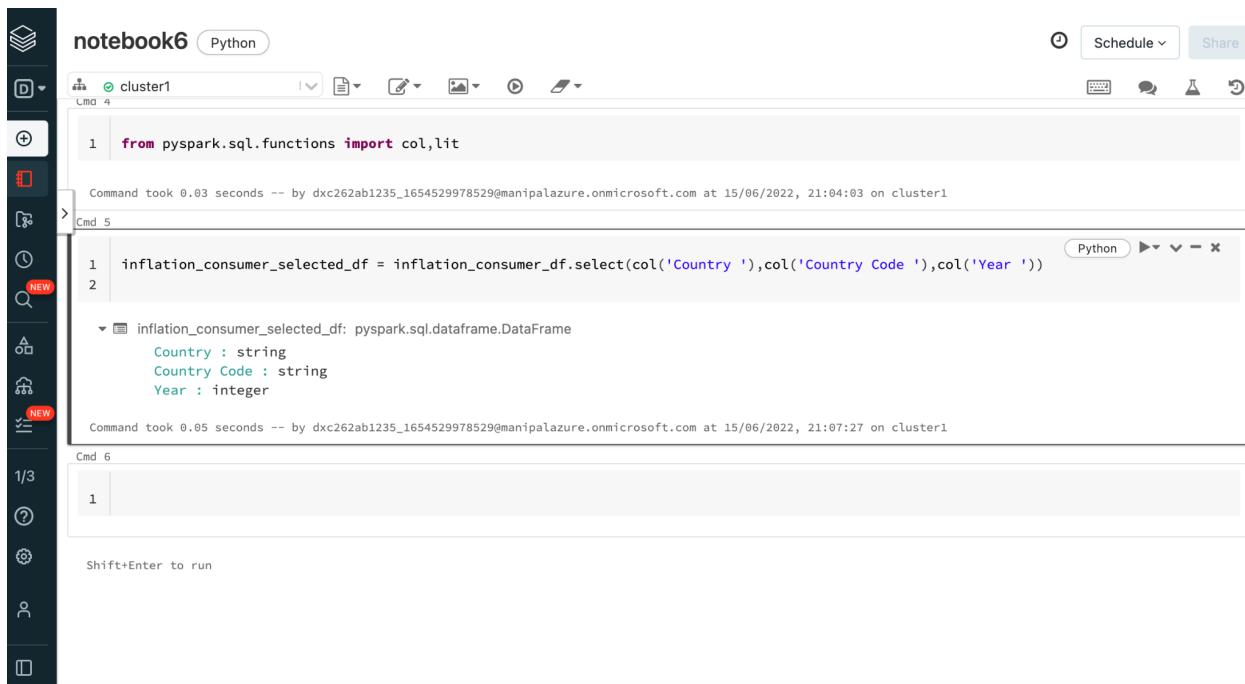


The screenshot shows a Jupyter Notebook titled "notebook6" in Python mode. It contains four command cells. The first three cells define the schema and read the CSV file into a DataFrame. The fourth cell is empty and has a placeholder "Shift+Enter to run". The sidebar on the left shows a tree view of the notebook's structure. A tooltip for the third cell indicates the DataFrame structure: Country : string, Country Code : string, Year : integer, Inflation : float.

```
inflation_consumer_df = spark.read \  
.option("header", True) \  
.schema(inflation_consumer_schema) \  
.csv("/FileStore/tables/inflation_consumer.csv")  
  
inflation_consumer_df: pyspark.sql.dataframe.DataFrame  
  Country : string  
  Country Code : string  
  Year : integer  
  Inflation : float  
  
Shift+Enter to run
```

Step 4:

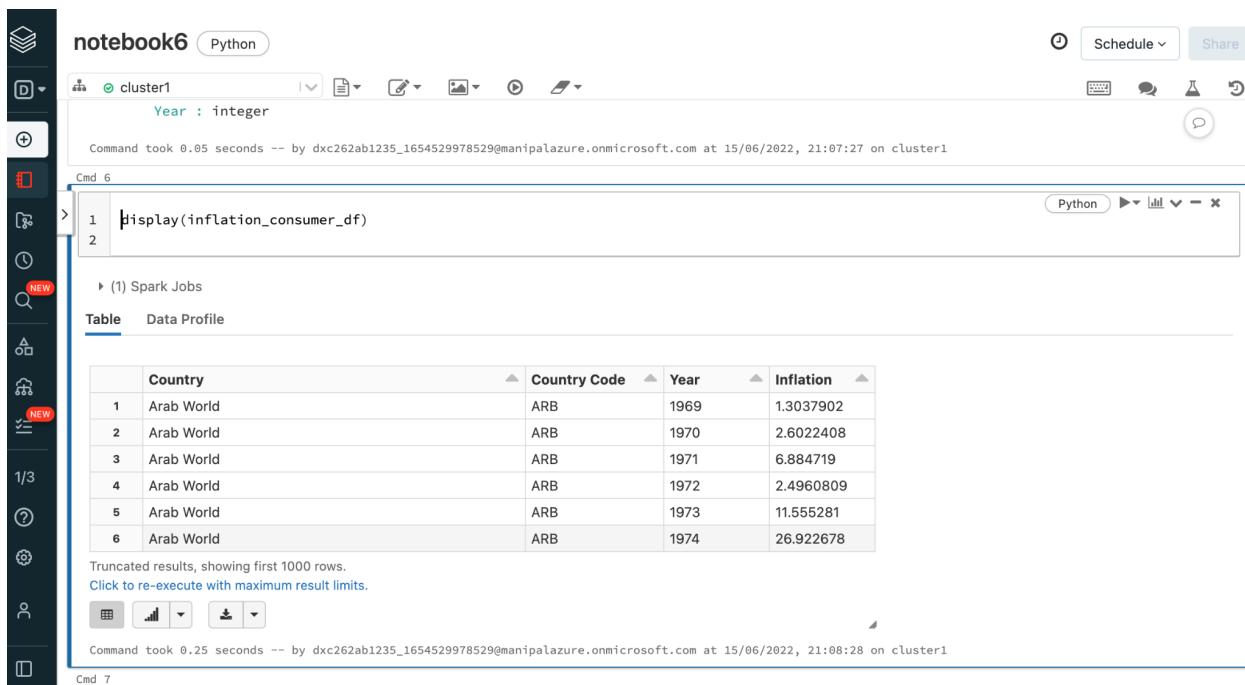
```
from pyspark.sql.functions import col,lit  
  
inflation_consumer_selected_df = inflation_consumer_df.select(col('Country'),col('Country Code'),col('Year'))
```



The screenshot shows a Jupyter Notebook titled "notebook6" running on a Python kernel connected to "cluster1". The notebook has a sidebar with various icons for file operations. The main area contains two code cells. The first cell (Cmd 4) contains the code from Step 4. The second cell (Cmd 5) shows the result of the execution, which is a DataFrame named "inflation_consumer_selected_df" with three columns: "Country" (string), "Country Code" (string), and "Year" (integer). A message indicates the command took 0.05 seconds to run.

Step 5:

```
display(inflation_consumer_df)
```



The screenshot shows the same Jupyter Notebook environment. The user has run the code from Step 5, which displays the contents of the DataFrame "inflation_consumer_df". The output shows a table with six rows, each representing a data point for "Arab World". The columns are "Country", "Country Code", "Year", and "Inflation". The "Country" column shows repeated entries for "Arab World", while the other columns provide specific values for each row. A message at the bottom indicates that the results are truncated, showing only the first 1000 rows.

| Country | Country Code | Year | Inflation |
|--------------|--------------|------|-----------|
| 1 Arab World | ARB | 1969 | 1.3037902 |
| 2 Arab World | ARB | 1970 | 2.6022408 |
| 3 Arab World | ARB | 1971 | 6.884719 |
| 4 Arab World | ARB | 1972 | 2.4960809 |
| 5 Arab World | ARB | 1973 | 11.555281 |
| 6 Arab World | ARB | 1974 | 26.922678 |