Reading data with Pyspark Dataframes in Databricks

Table of Contents (you can navigate directly by clicking on this points)

Note → Arrow ⬆ is used to go back to "table of contents".

Let's start.

Introduction to Dataframes and Various File Formats ⬆

Dataframe - Dataframe is the most common structured API of Spark, it is simply a table with rows and columns.

Schema of a Dataframe - The list of columns and their types.

Why we need Dataframes? (When we have already various options, e.g. - Microsoft Excel) – Spark Dataframe can span over thousands of computers, the reason behind this is –

- The data is either very large to get fit on a single machine. Or
- The time taken to perform the computation on a single machine is too long.

So, we simply put the data on multiple machines.

Let's talk about various formats (from which we are reading data using Dataframes) –

1) CSV - Values are separated by comma.
   Example –
   Name, Age

Chinky, 5
Minky, 7

2) JSON - Data is stored in key-value pairs.
   Example –
   {'Name': 'Chinky', 'Age': 5}
   {'Name': 'Minky', 'Age': 7}
3) Parquet - Columnar storage format and provides efficient data compression. Also, query performance is increased over row-based data stores. It supports limited schema evolution. It is a split-able file format.
   Example – Column based data store

Name	Chinky	Minky
Age	5	7

4) Text – Data where each record is delineated by a newline character.
   Example –
   My name is Chinky.
   I am 5 years old.
5) Tables – The collection of rows and columns, where each column represents the field name and row contains the values of those fields.
   Example –

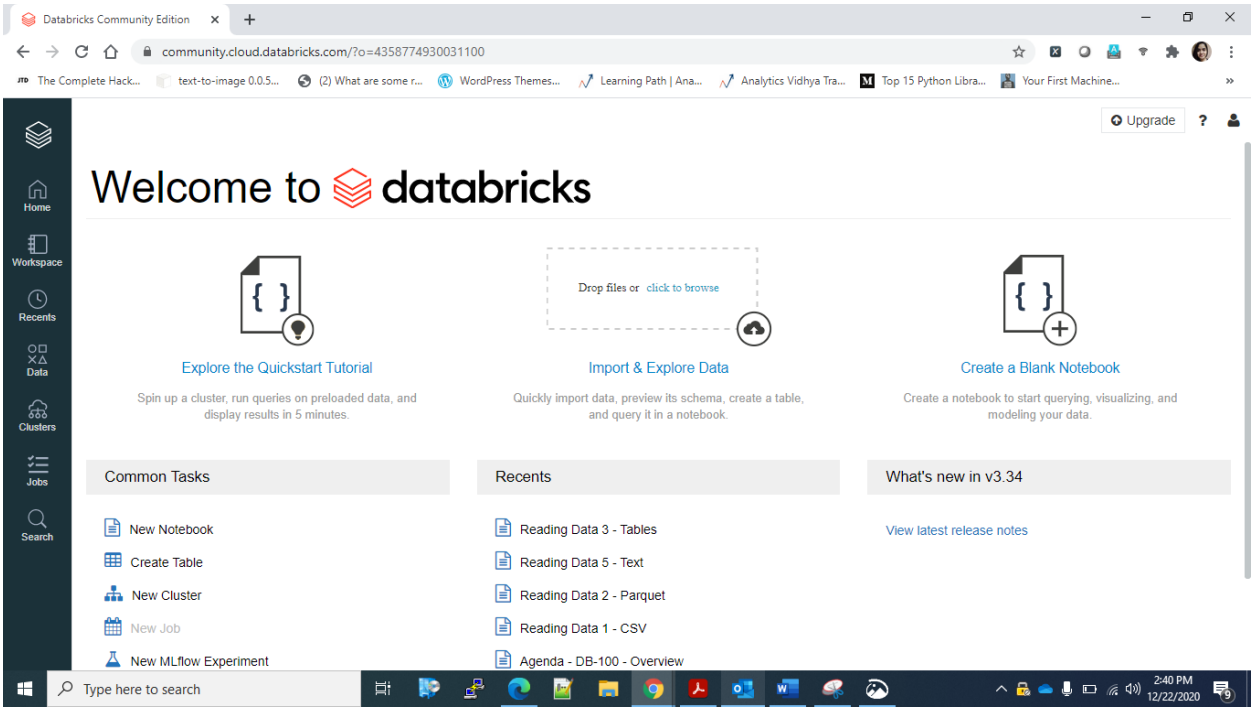
Name	Age
Chinky	5
Minky	7

Databricks UI

Let's discuss few things about databricks UI –

- How to create cluster in Databricks?
- How to upload files in Databricks?
- How to create tables in Databricks with UI?
- How to create notebook in Databricks?

Before going into these points, just make sure you have a databricks account (or go for a community edition). Once you have successfully created your account, the UI will look like this –

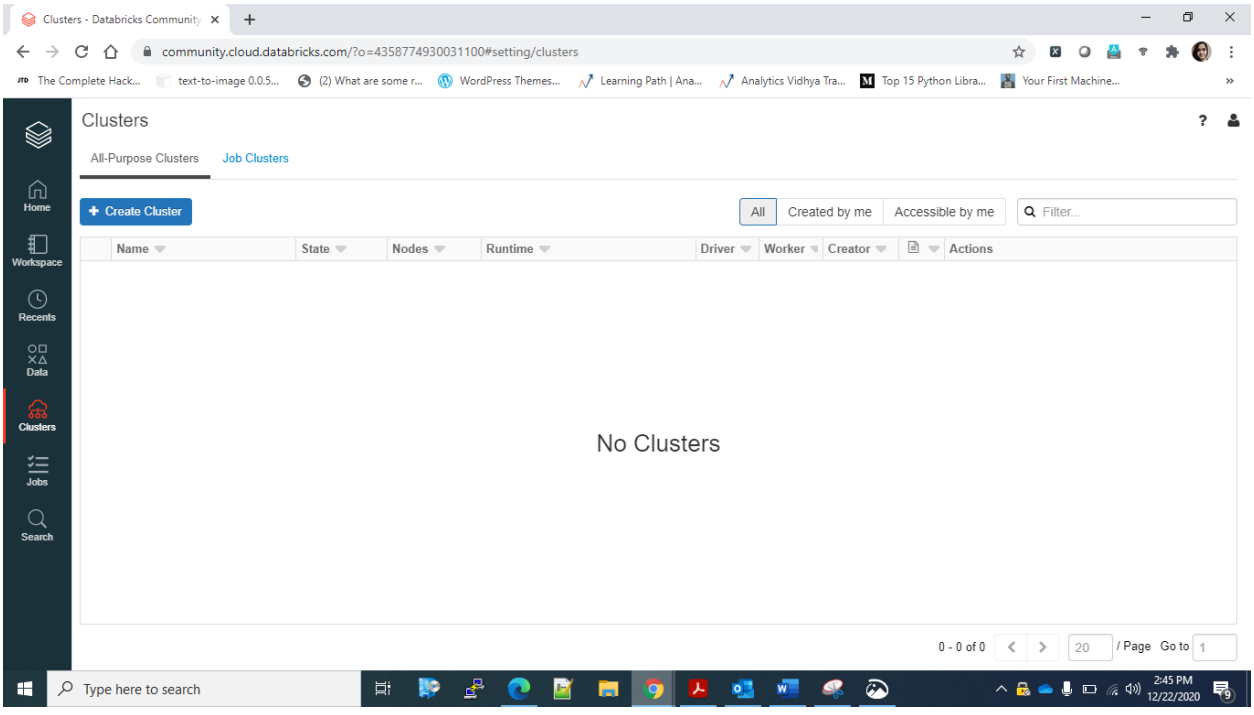


(Note – Your UI might look different than me, because my account is in community edition).

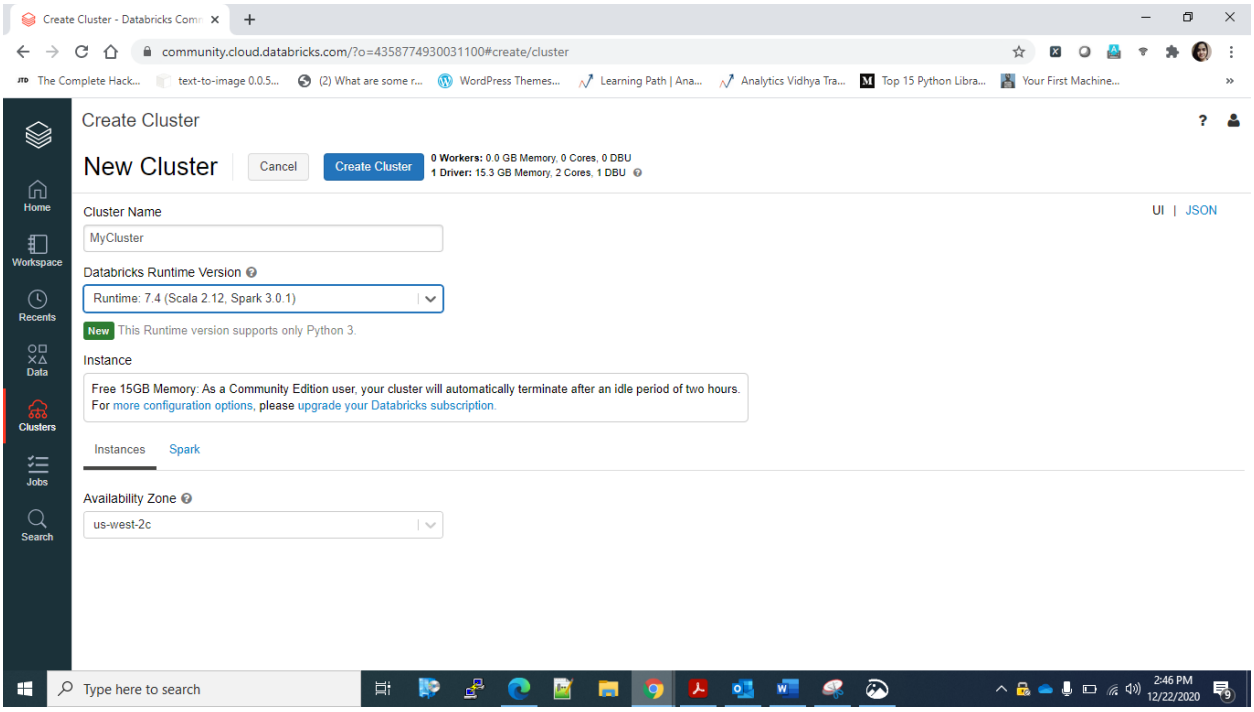
Let's start answering to our first question.

How to create cluster in Databricks? -

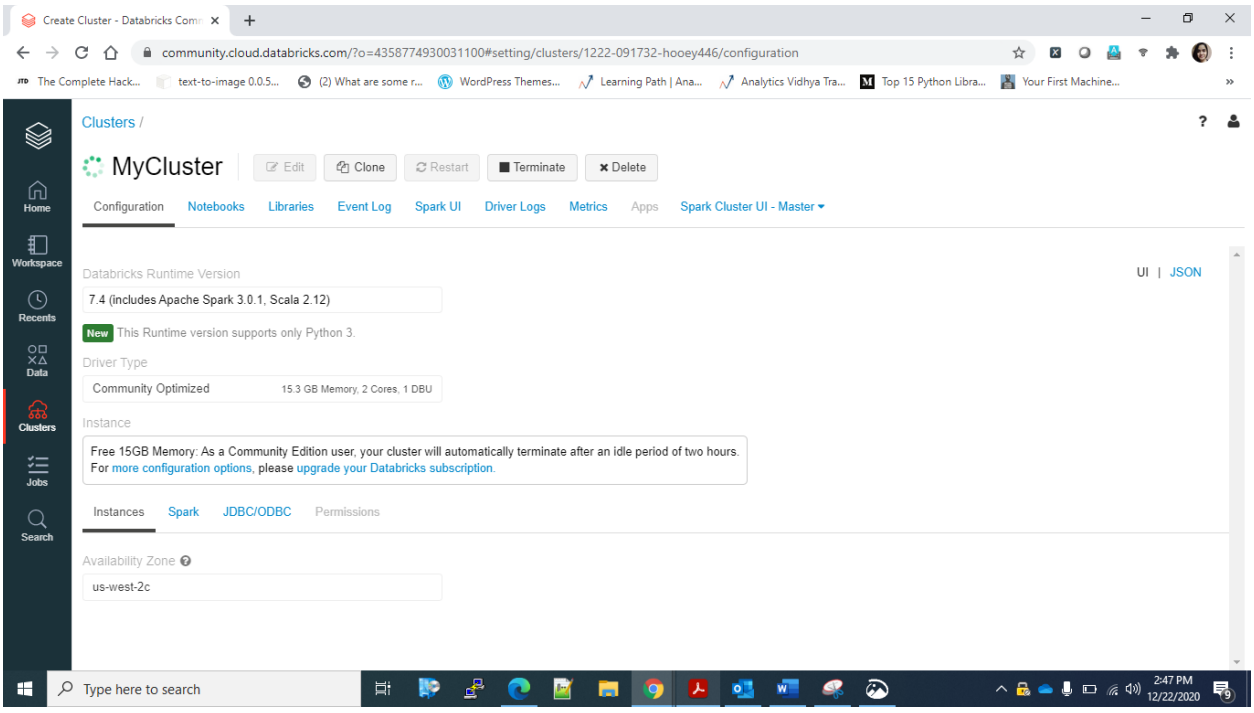
In the right side of the screen, you will find "Clusters" option, click on that. The page will look like this, once you clicked on that button –



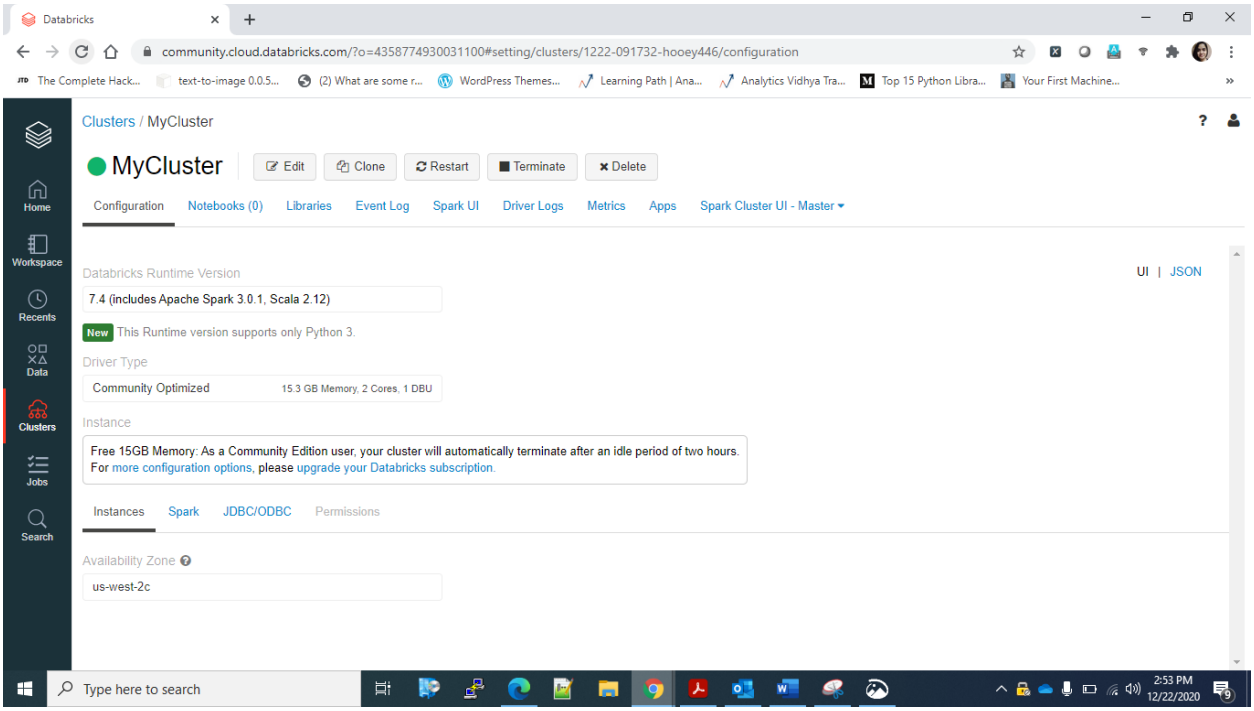
Now, click on “+ Create Cluster” button.



Fill your cluster name and hit the button “Create Cluster”.



Wait for that green “dot-dot” circle to become a full round circle. Just like the below screen.

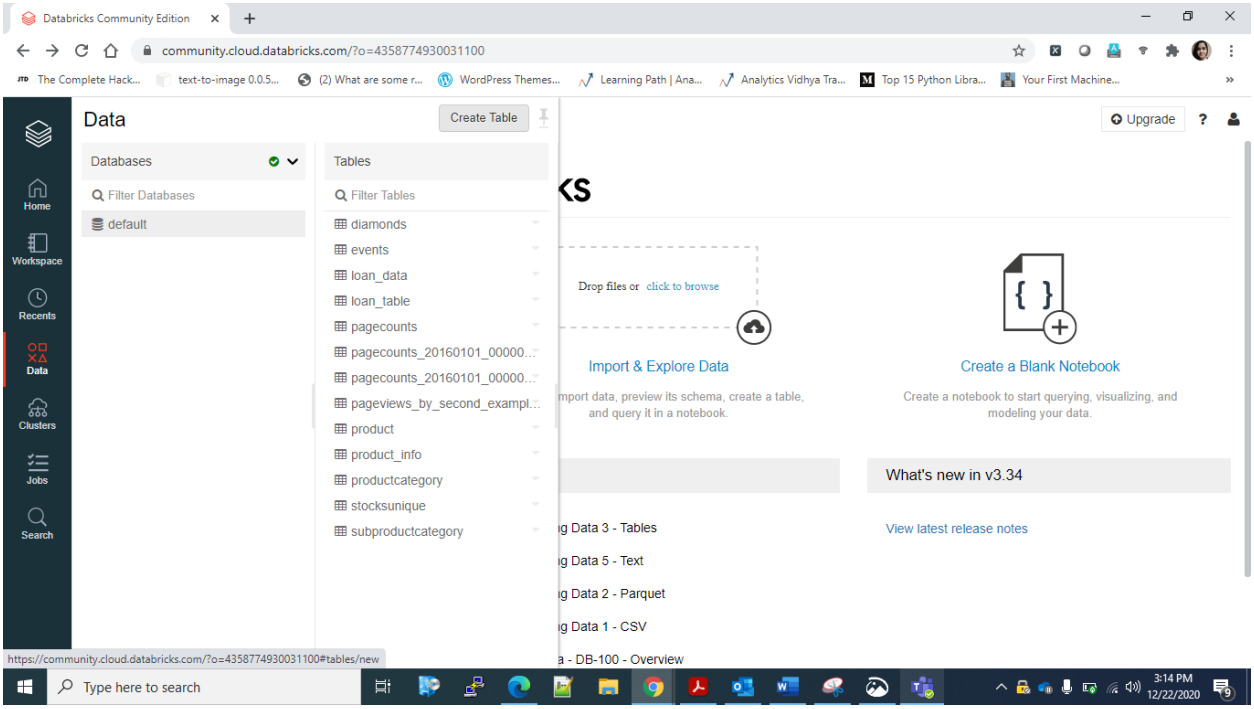


Now you are ready to use your cluster.

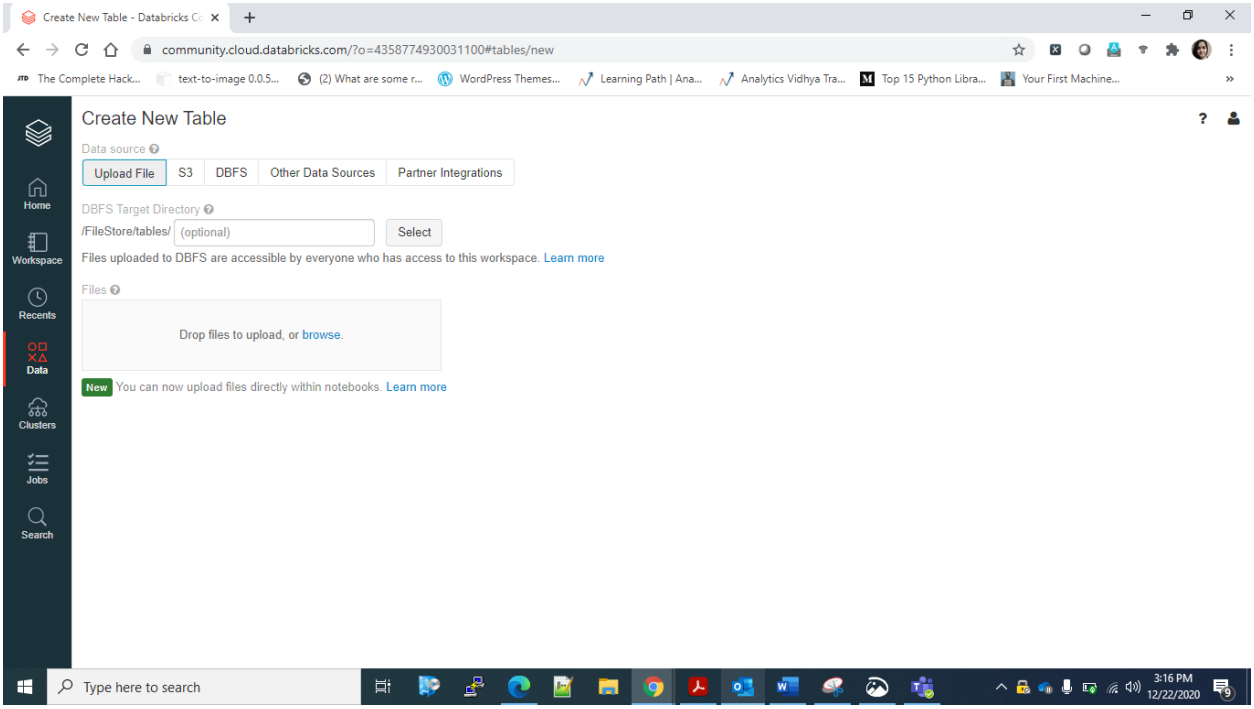
Let's go to second question.

How to upload files in Databricks? -

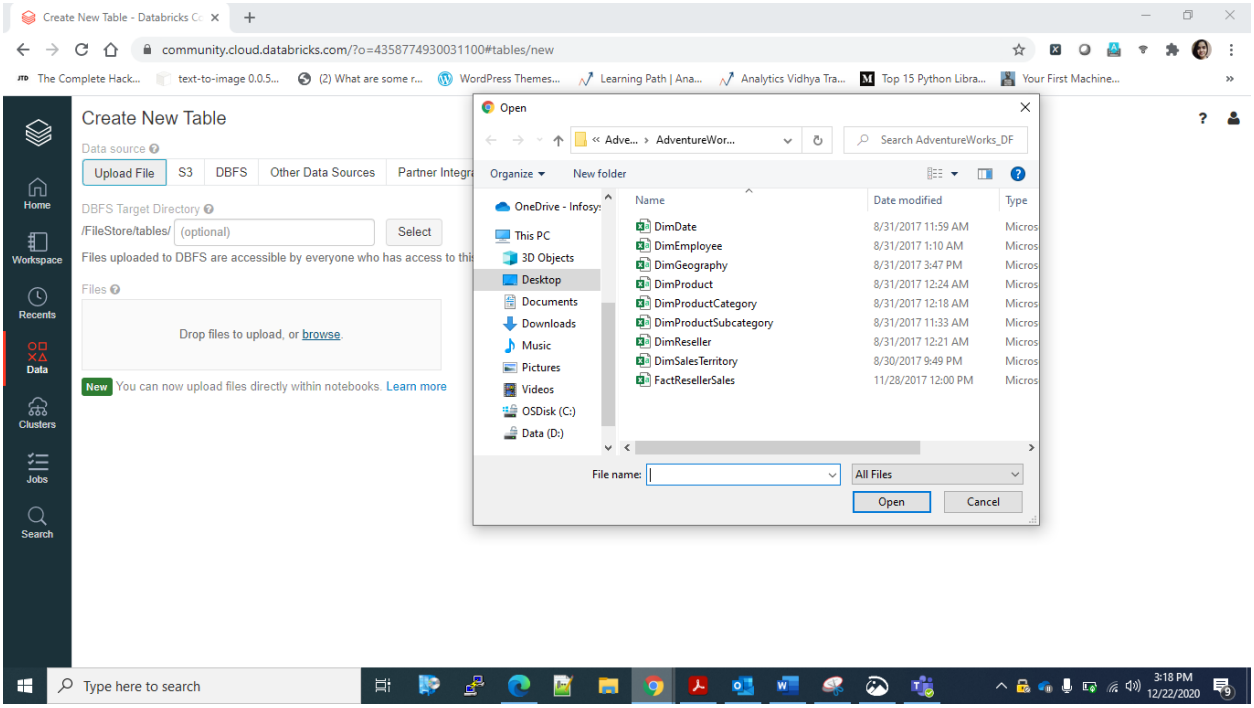
In the right side of the screen, you will find "Data" option, click on that.



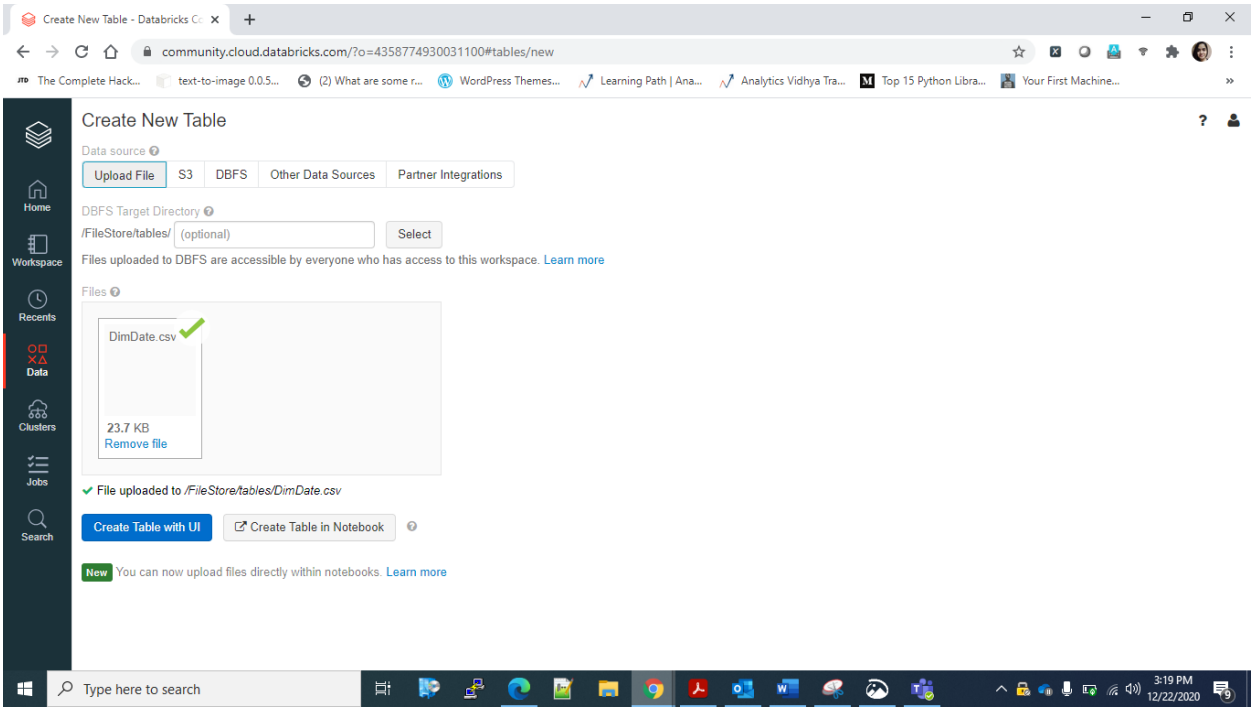
Click on “Create Table” button, the UI will look like this –



Don’t get confused with the title “Create New Table”, here we are only uploading files by clicking on “Drop files to upload or browse”.



Select your file(s) and click “Open”.

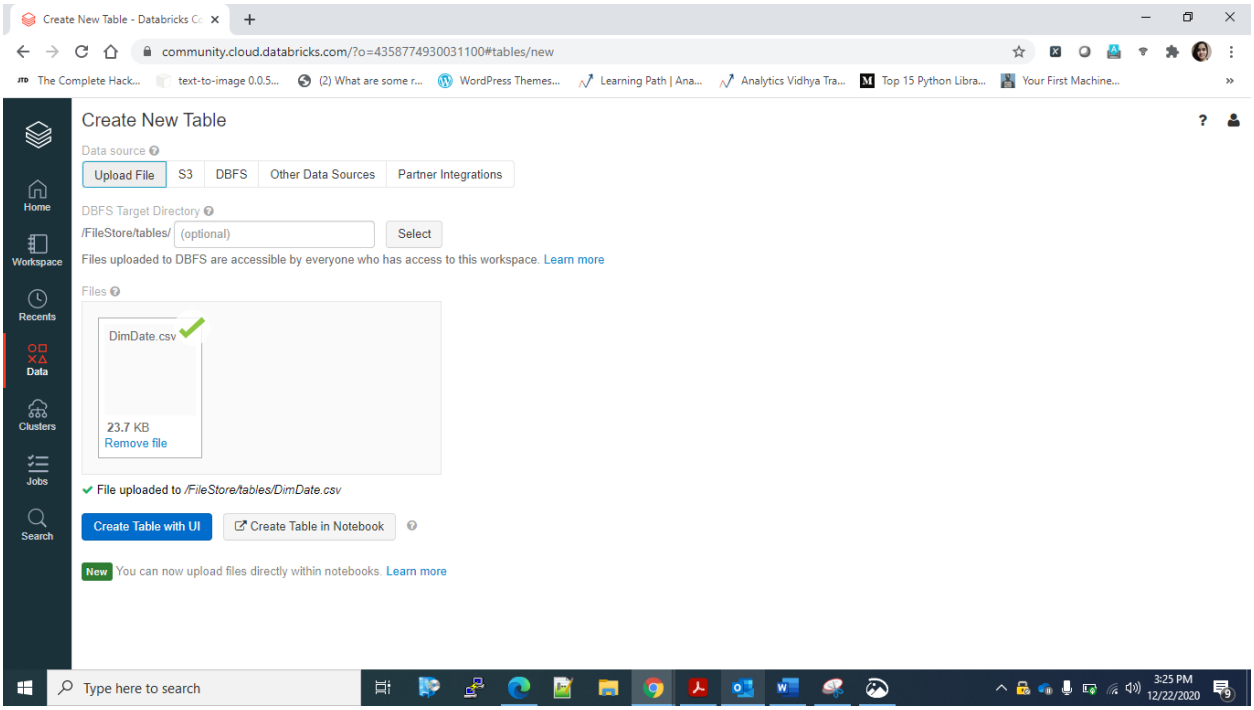


You can see that our file is successfully uploaded to "/FileStore/tables/DimDate.csv"

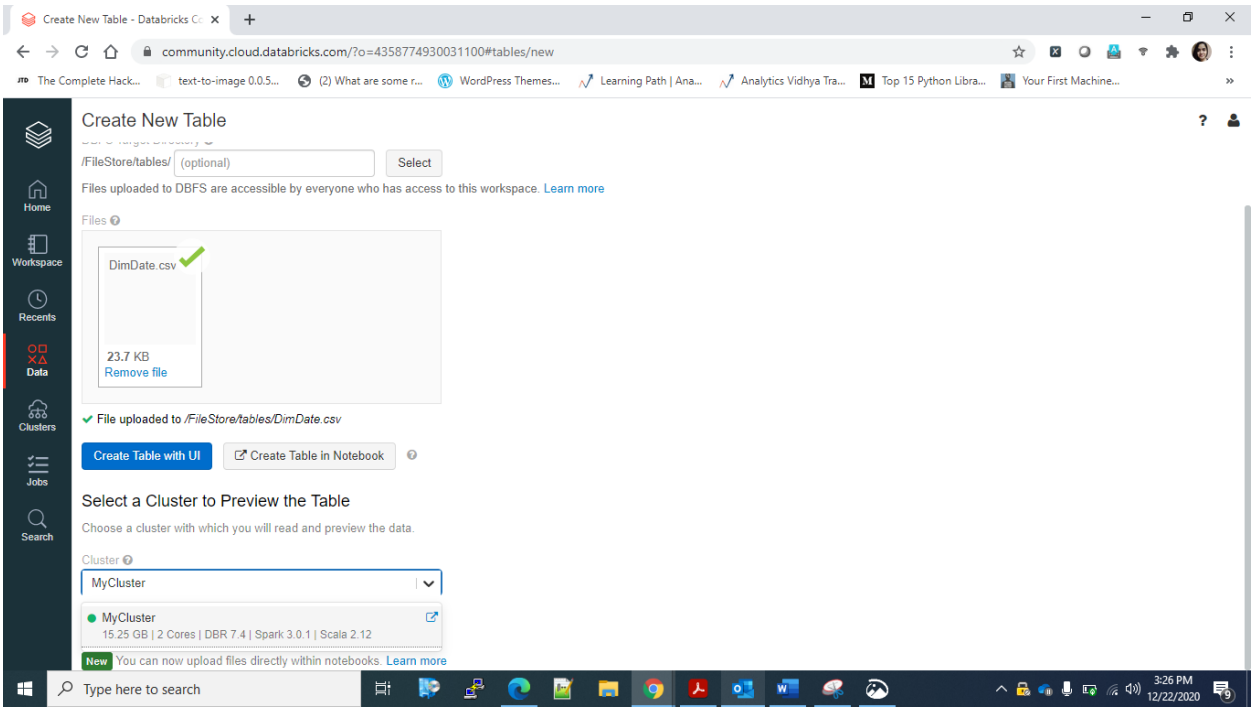
From the above image, I think you have got the idea how will we be going to create tables with UI. But if not, let me answer the next question.

How to create tables in Databricks with UI? -

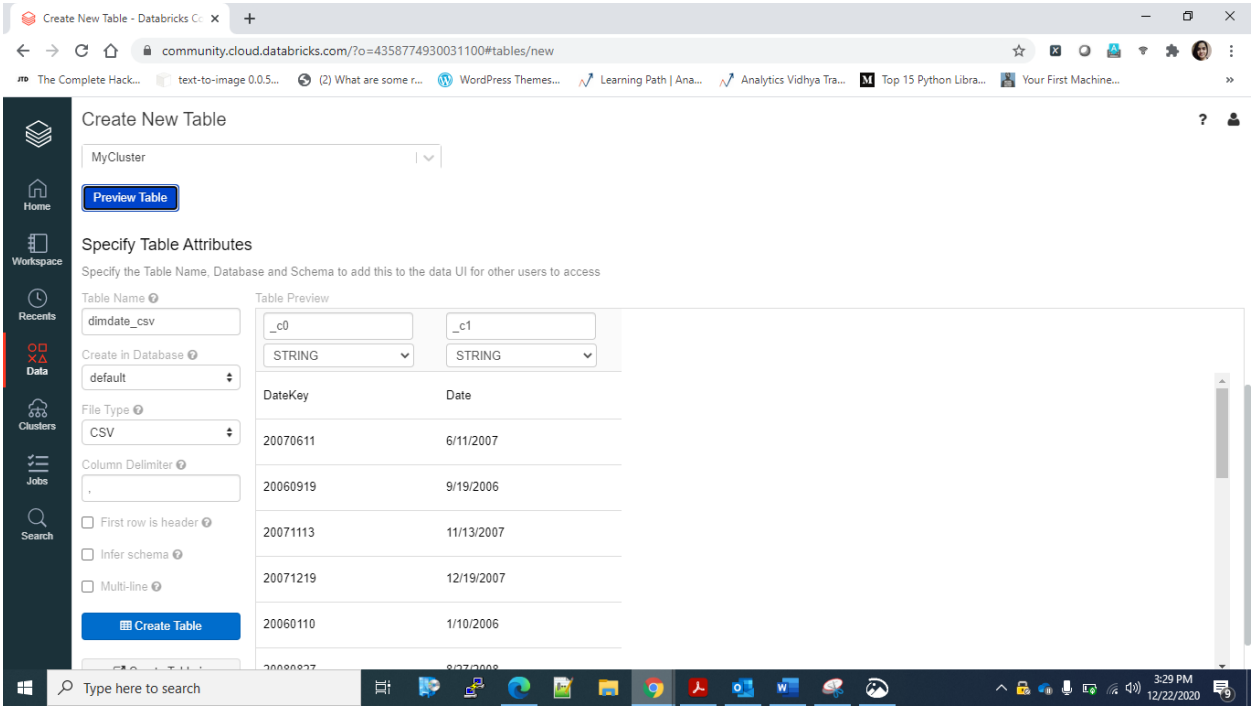
First, we will upload the file, and then create a table from it.



Now, click on “Create Table with UI” button. It will ask you to select a cluster to preview the table. Simply enter your cluster name.

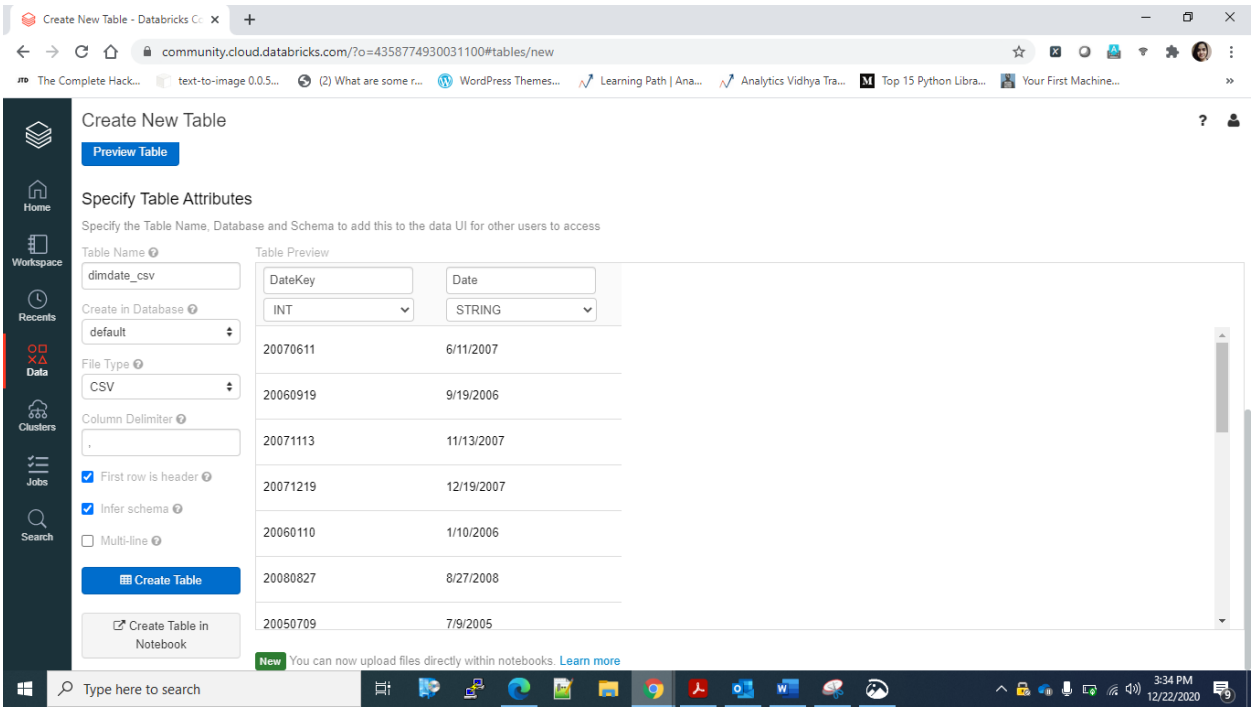


Simply enter your cluster name. Hit the button “Preview Table”.

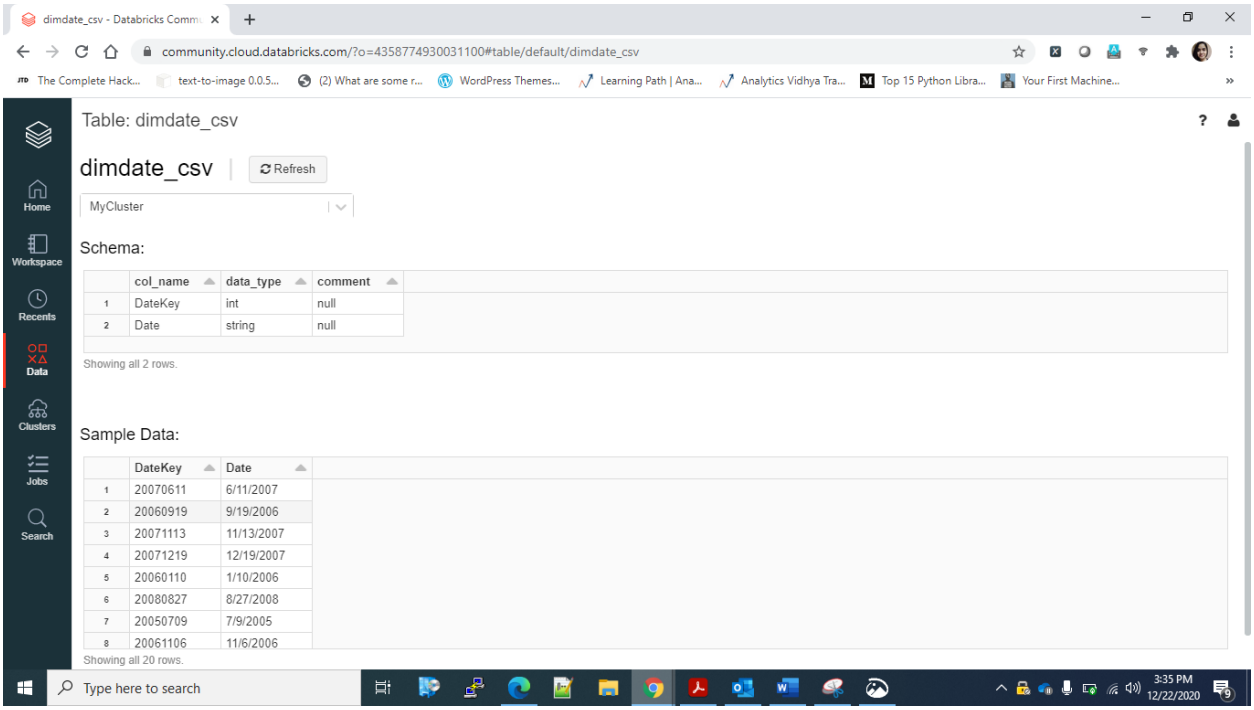


Screen will look like this. Now add the table name, database name(if you want the table in a particular database), column delimiter, then hit the checkbox if first row is a header, if you want

to infer schema and if data is expand to multi-lines. Once, you have done all these, the screen will look like this –



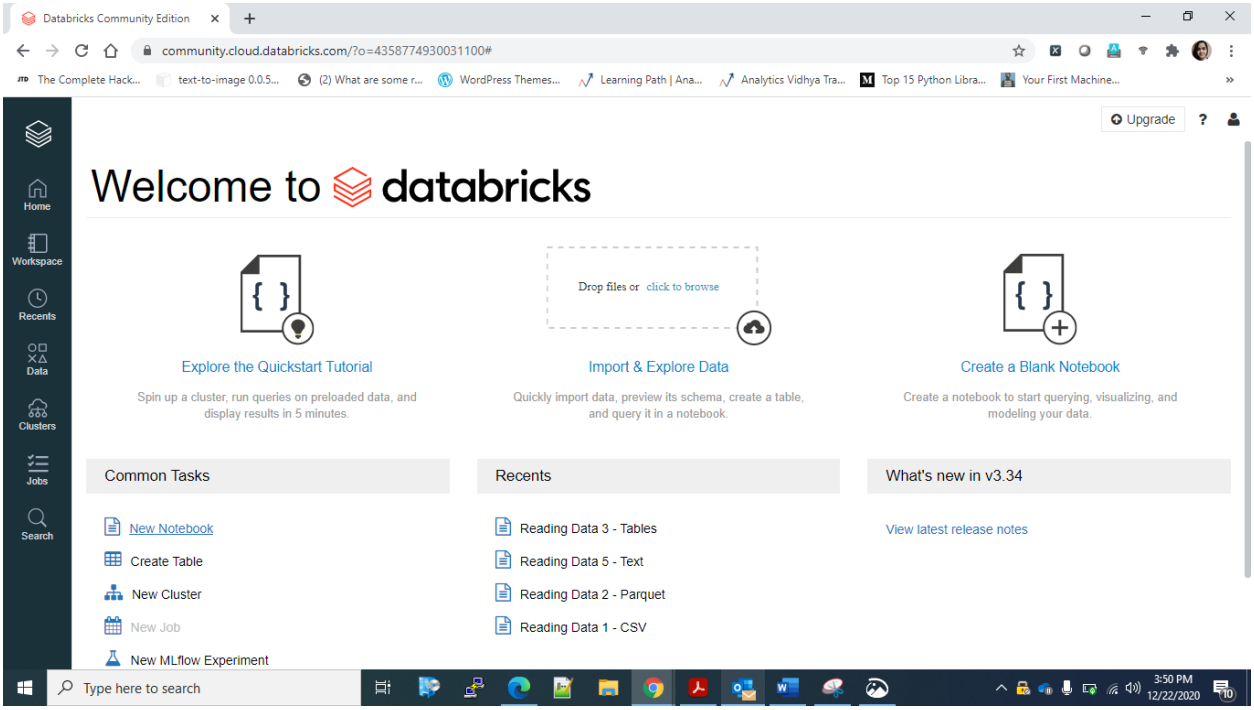
Let's done the table creation by hitting the button "Create table". You can verify the details of your table from this screen.



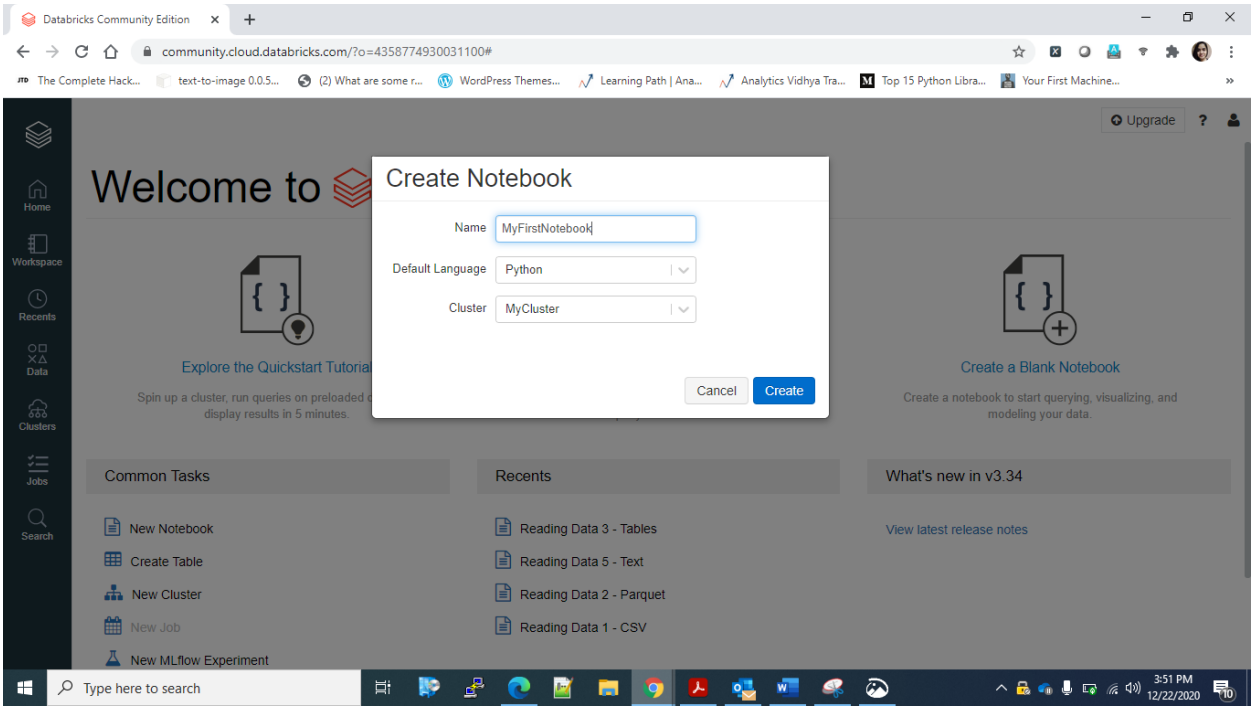
Let's go to next question.

How to create notebook in Databricks with UI? -

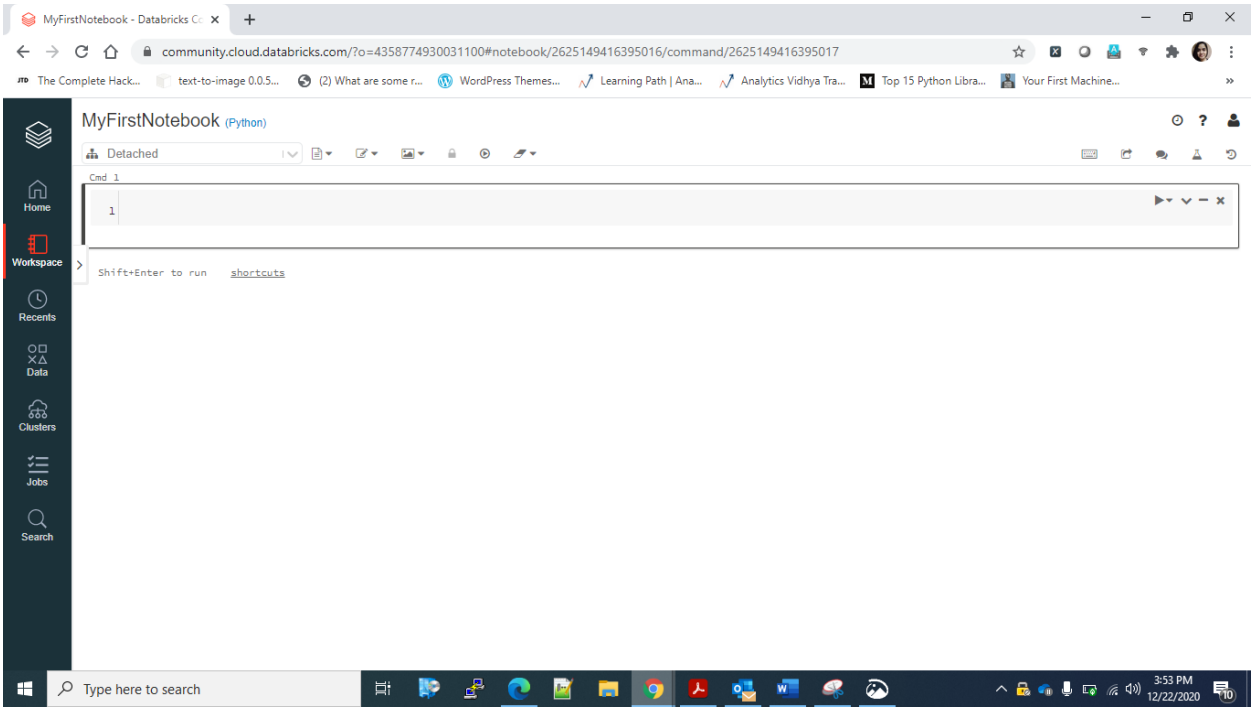
Click on “New Notebook” button.



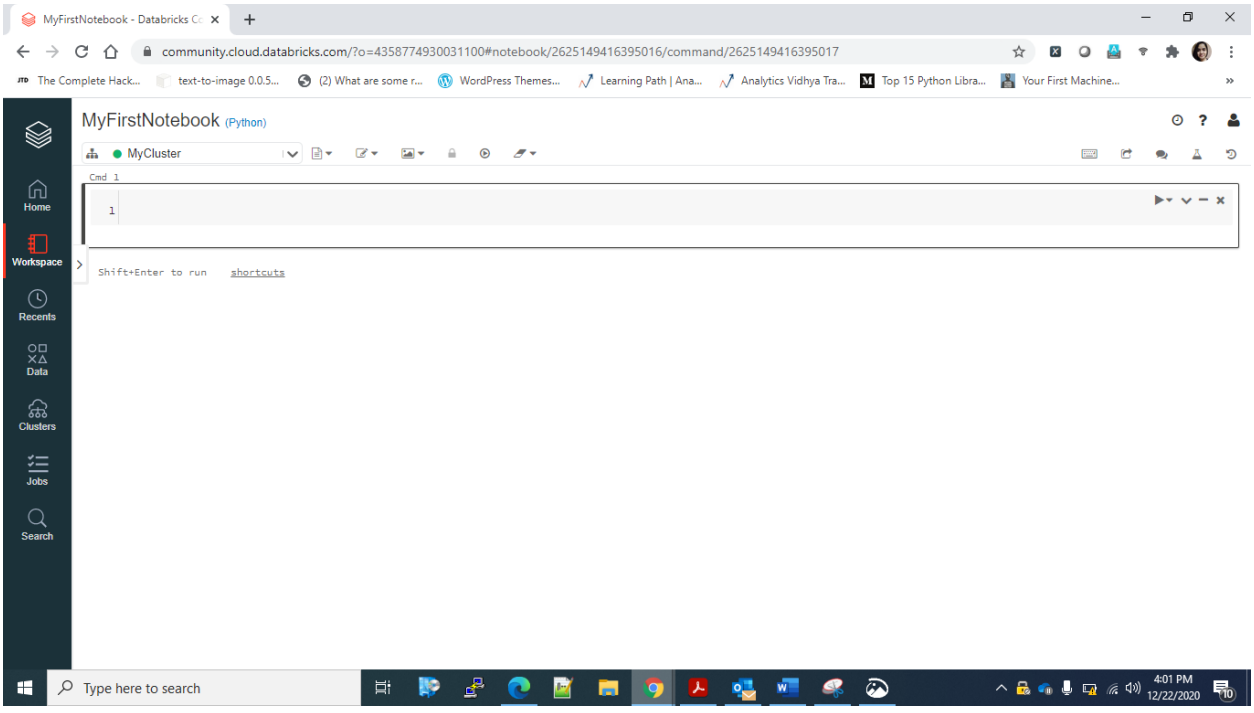
Enter your Notebook name and click the “Create” button.



Below your notebook name, you can see that our notebook is detached. In order to execute our commands, we need to attach the notebook to our cluster.



Once, we have attached our cluster, our screen will look like this –



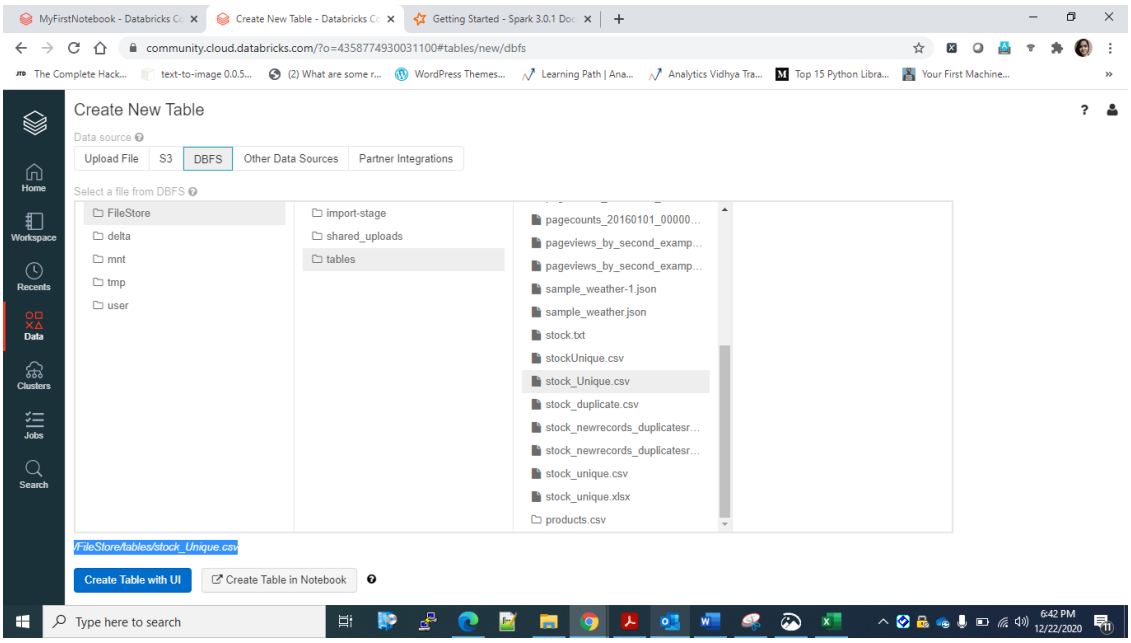
Hence, we have successfully created a cluster, uploaded our files, created a table from a file and finally created a notebook and attached it to our cluster.

Let's get back to our main task – "Reading data using various formats in Pyspark Dataframes.

Here, we will discuss –

Reading data using csv format

We will be working on stocks.csv file in this tutorial, let's copy the address (you can check in the Data section as shown below) –



Our stocks_Unique file looks like this –

	A	B	C	D	E
1	1	11-Dec-20	1000	998	2
2	2	11-Dec-20	2000	1500	500
3	3	11-Dec-20	500	350	150
4	4	11-Dec-20	300	200	100
5	5	11-Dec-20	650	470	180
6	70	11-Dec-20	10000	9800	200
7	130	11-Dec-20	200	150	50
8	165	11-Dec-20	210	150	60
9	220	11-Dec-20	800	700	100
10	121	11-Dec-20	900	470	430
11					

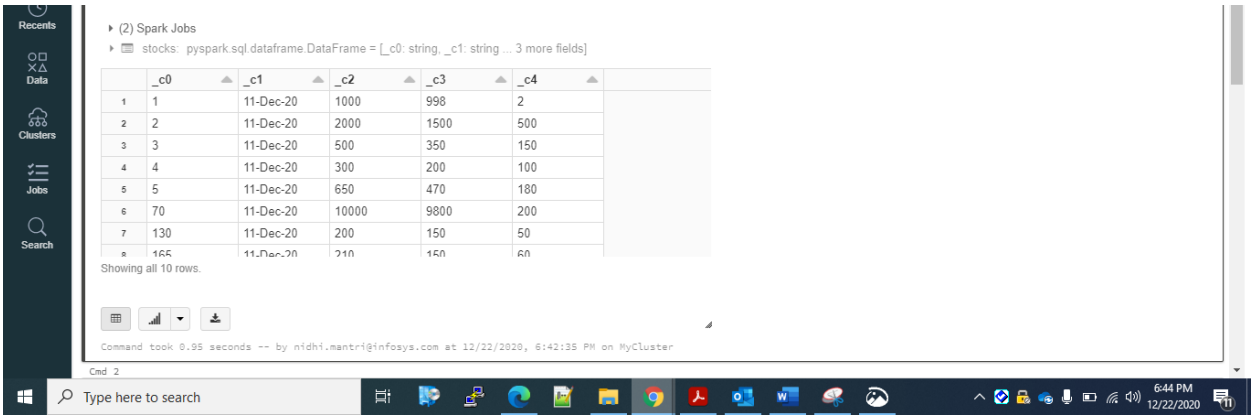
Its schema is –

- Product_id
- Sale_Date
- Total Qty
- Available Qty
- Sale Qty

Let's read the data into Dataframe. (we added "dbfs:" in the file address because our file is stored in dbfs (databricks file system))

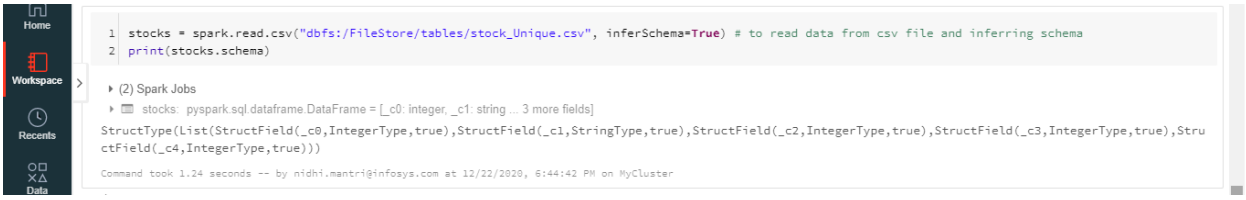
```
stocks = spark.read.csv("dbfs:/FileStore/tables/stock_Unique.csv")

display(stocks)
```



Here you can see – type of every column is string, try inferring schema now.

```
stocks = spark.read.csv("dbfs:/FileStore/tables/stock_Unique.csv", inferSchema=True)
print(stocks.schema)
```



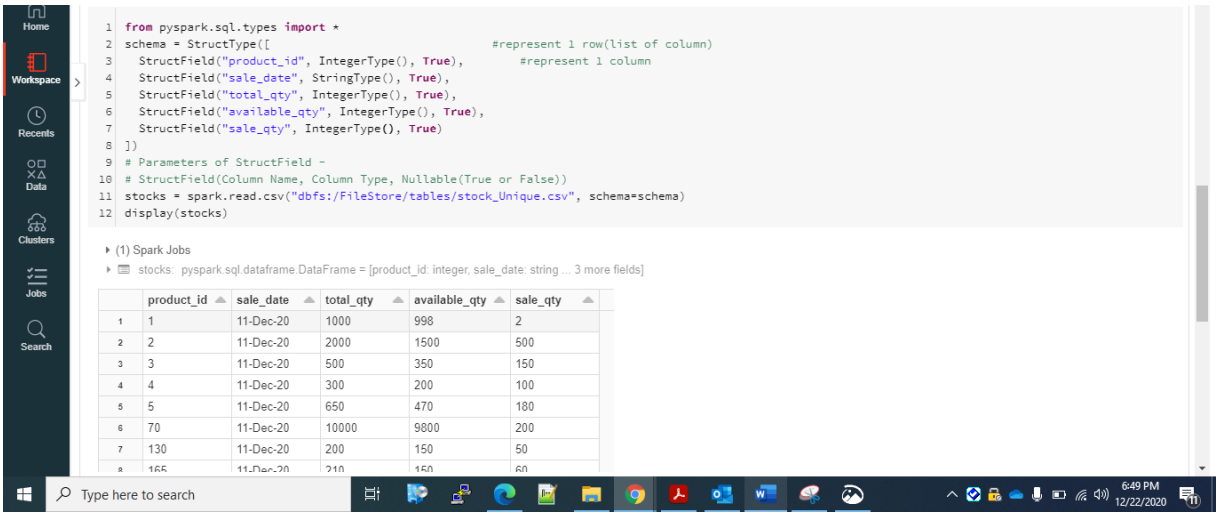
Now, we can see it inferred the schema quite well. But you might get confused what is StructType and StructField right?

Let me answer it first. Whenever we define schema of a Dataframe, we represent row by StructType and column by StructField. And as row is list of various columns, the StructType is also the list of StructField.

Note – inferring schema takes long time when we deal with large data, because it scans whole data. Hence providing schema is the best solution.

Also, there is no name of any column and it is very hard to find what our columns represent. So, now try adding schema (column names and their types) to it.

```
from pyspark.sql.types import *
schema = StructType([                          #represent 1 row(list of columns)
  StructField("product_id", IntegerType(), True),       #represent 1 column
  StructField("sale_date", StringType(), True),
  StructField("total_qty", IntegerType(), True),
  StructField("available_qty", IntegerType(), True),
  StructField("sale_qty", IntegerType(), True)
])
# Parameters of StructField -
# StructField(Column Name, Column Type, Nullable(True or False))
stocks = spark.read.csv("dbfs:/FileStore/tables/stock_Unique.csv", schema=schema)
display(stocks)
```



Pretty good output, right? Yeah!!

Note – if your file has a header, write command –

```
# stocks = spark.read.csv("file name", header=True, schema=schema)
```

So, with this, we learnt how to read data using csv format.

Reading data using json format

We will be working on sample_weather.json file in this tutorial. Our file address is – "dbfs:/*FileStore/tables/sample_file.json*". Let's start loading data in the dataframe.

Note – the JSON reader also assumes...

- That there is one JSON object per line and
- it's delineated by a new line.

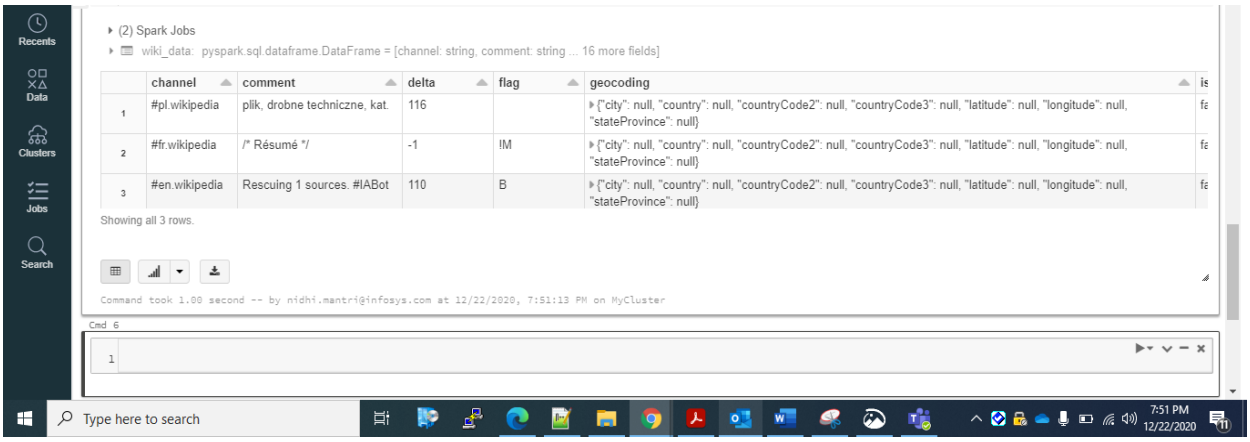
Our sample_file.json data looks like –

```
%fs head "dbfs:/FileStore/tables/sample_file.json"
```

Let's try reading the data into dataframe.

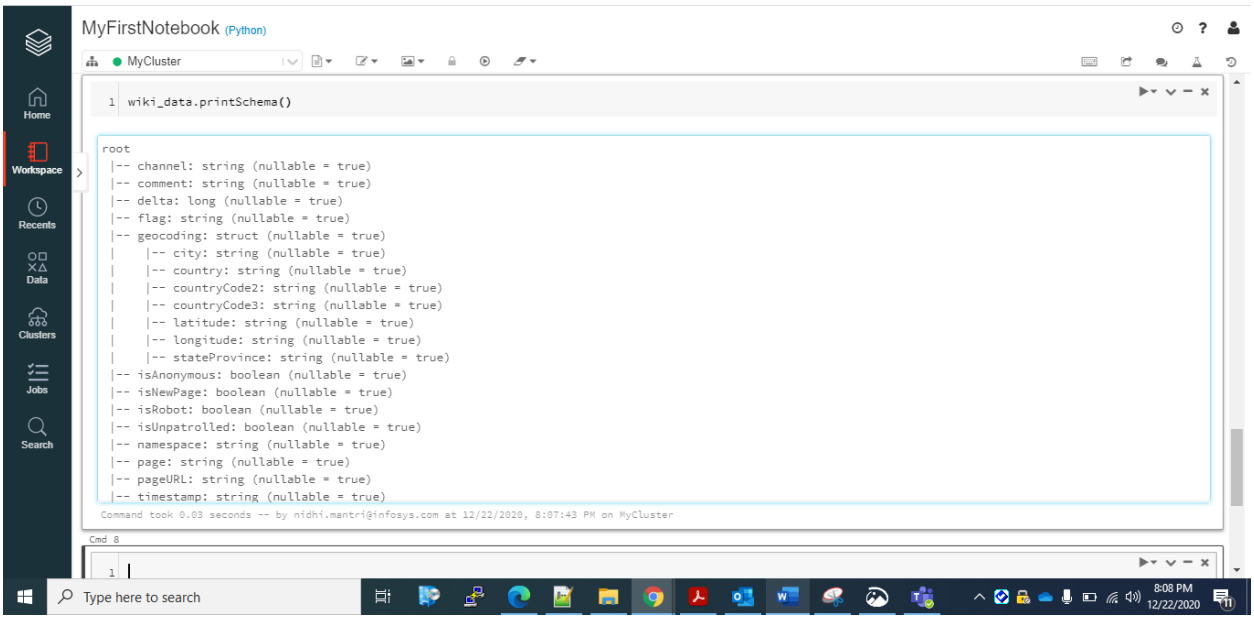
```
wiki_data = spark.read.json("dbfs:/FileStore/tables/sample_file.json")

display(wiki_data)
```



Here, we can see in the schema that it gets all the column names and their default types(string and Boolean), but it did not infer the data types correctly. (ex. - values which are of type double are inferred as strings).

```
wiki_data.printSchema()
```

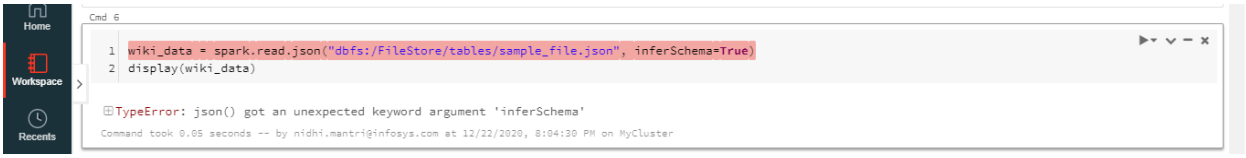


Let's infer the schema –

```
#/FileStore/tables/sample_weather.json

wiki_data = spark.read.json("dbfs:/FileStore/tables/sample_file.json", inferSchema=True)

display(wiki_data)
```



Oops!! We got an error, let's try providing user defined schema to it.

```
from pyspark.sql.types import *

jsonSchema = StructType([

  StructField("channel", StringType(), True),

  StructField("comment", StringType(), True),

  StructField("delta", IntegerType(), True),

  StructField("flag", StringType(), True),

  StructField("geocoding", StructType([

    StructField("city", StringType(), True),
```

```
    StructField("country", StringType(), True),
    StructField("countryCode2", StringType(), True),
    StructField("countryCode3", StringType(), True),
    StructField("stateProvince", StringType(), True),
    StructField("latitude", DoubleType(), True),
    StructField("longitude", DoubleType(), True)]), True),
  StructField("isAnonymous", BooleanType(), True),
  StructField("isNewPage", BooleanType(), True),
  StructField("isRobot", BooleanType(), True),
  StructField("isUnpatrolled", BooleanType(), True),
  StructField("namespace", StringType(), True),
  StructField("page", StringType(), True),
  StructField("pageURL", StringType(), True),
  StructField("timestamp", StringType(), True),
  StructField("url", StringType(), True),
  StructField("user", StringType(), True),
  StructField("userURL", StringType(), True),
  StructField("wikipediaURL", StringType(), True),
  StructField("wikipedia", StringType(), True)])
```

Now, again read the data with schema provided.

```
wiki_data = spark.read.json("dbfs:/FileStore/tables/sample_file.json", schema=jsonSchema)
wiki_data.printSchema()
```

Yeah! We have done it right.

Reading data using parquet format

I have already created a parquet file of stocks_Unique file.

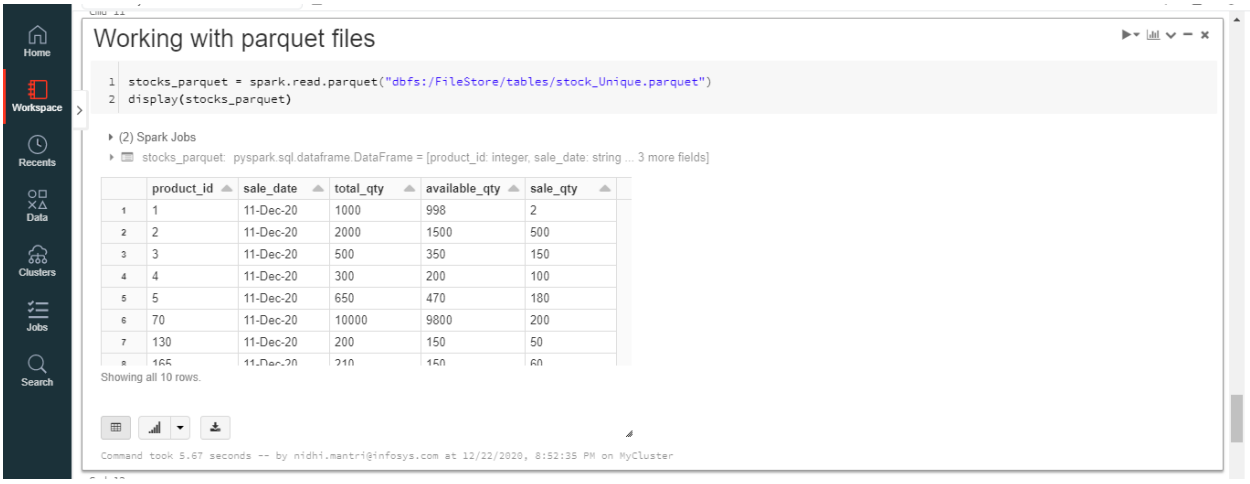
Command I used to save the stocks dataframe into parquet file is –

```
stocks.write.parquet("dbfs:/FileStore/tables/stock_Unique.parquet")
```

Let's try reading data from it.

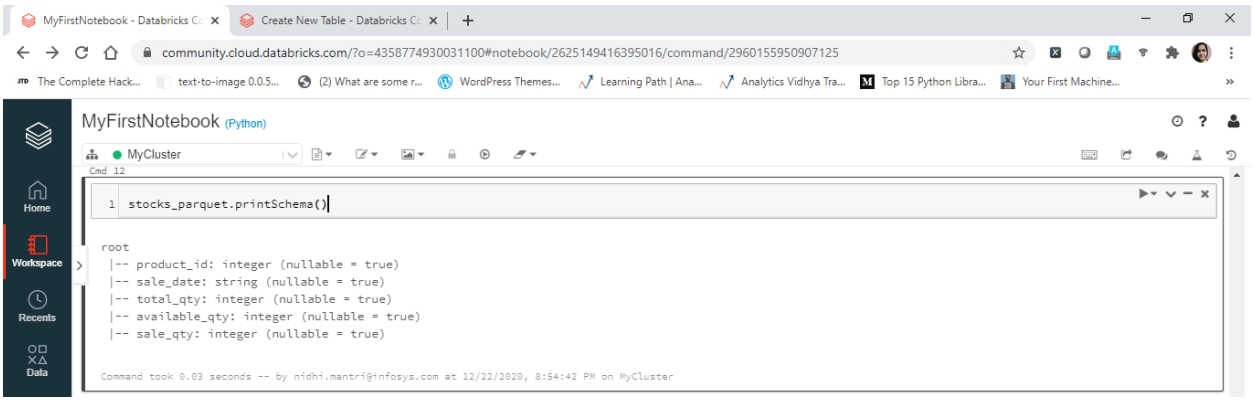
```
stocks_parquet = spark.read.parquet("dbfs:/FileStore/tables/stock_Unique.parquet")

display(stocks_parquet)
```



You can see I did not provide the schema to it. Let's try printing the schema.

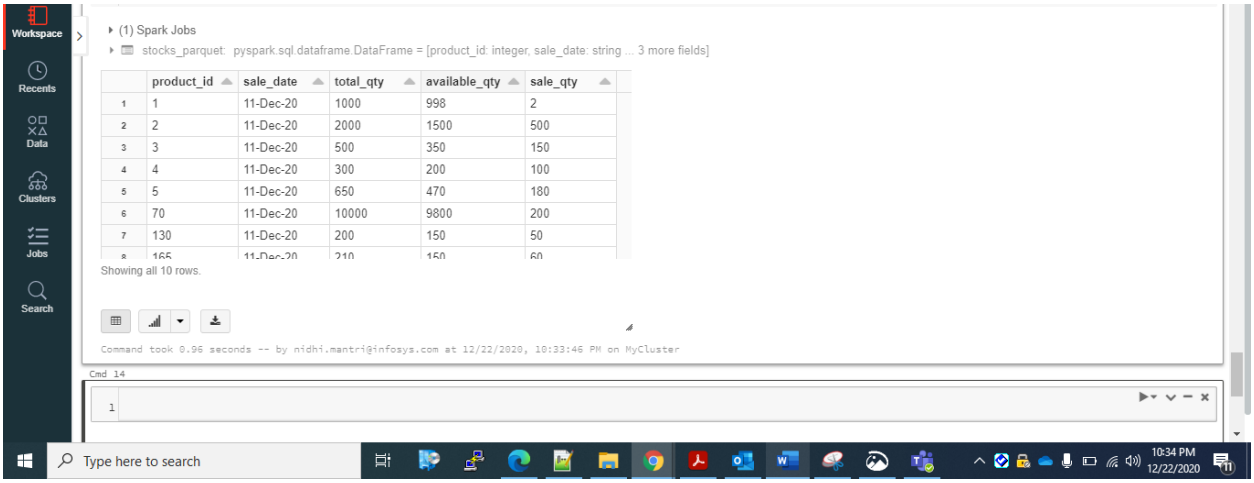
```
stocks_parquet.printSchema()
```



Here, we get the right schema, because parquet stores the schema (column names and their data types) in parquet files. It requires 1 job to read the schema from those files. And if we want to avoid that 1 job, we can provide schema (user defined) while reading data from parquet, which is shown below –

```
from pyspark.sql.types import *

schema = StructType([                    #represent 1 row(list of columns)
  StructField("product_id", IntegerType(), True),      #represent 1 column
  StructField("sale_date", StringType(), True),
  StructField("total_qty", IntegerType(), True),
  StructField("available_qty", IntegerType(), True),
  StructField("sale_qty", IntegerType(), True)
])
stocks_parquet =(spark.read
          .schema(schema)
          .parquet("dbfs:/FileStore/tables/stock_Unique.parquet"))
display(stocks_parquet)
```



Compare the spark jobs it took to run the commands (with or without schema). Here, we need to provide the schema before providing our file. (Try providing the schema as – **spark.read.parquet(filename, schema=schema))**

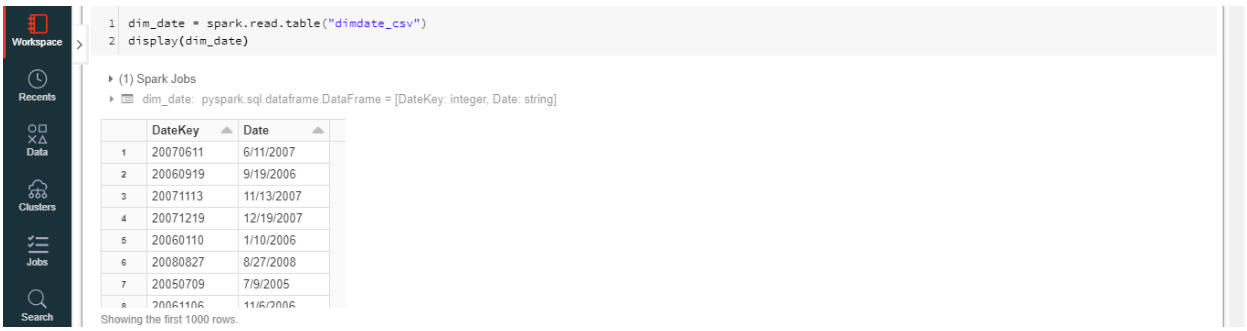
So, with that, we have successfully read the data from parquet files. As we mentioned earlier, query performance is increased when we work with parquet files.

Reading data from tables using table format

Let's try to read data into dataframe from dimdate_csv table we created earlier. While reading data from tables, we don't need to provide schema, because we already provided it while creating tables.

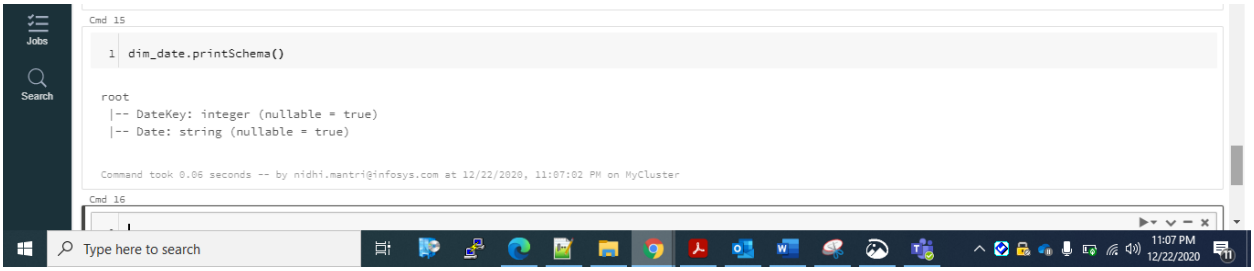
```
dim_date = spark.read.table("dimdate_csv")

display(dim_date)
```



Printing the schema to verify whether we get the right schema or not.

```
dim_date.printSchema()
```



We got the right schema; hence we have successfully loaded the data from table into dataframe.

Reading data from text files using csv and text format

Here we will discuss two types of text data. First, when our data is complete text, like there is no rows and columns, then we use text format. Second, when our data is like csv, tsv format, we use csv format.

Sample of both type of data is – (first data is simple raw text and second data is tab separated).

```
----------------------
Name - Nidhi Mantri
Emp # - 1101483
----------------------
Daily Learning - 1

1. Apache Spark Overview
    - Short History of Apache Spark
    - Who is Databricks?
    - What is Apache Spark?
    - Spark - A Unifying Engine
    - The RDDs and The DataFrames
    - Scala, Python, Java, R & SQL
    - The Cluster: Drivers, Executors, Slots & Tasks
    - Quick Note on Jobs & Stages
    - Quick Note on Cluster Management
```

orders - Notepad

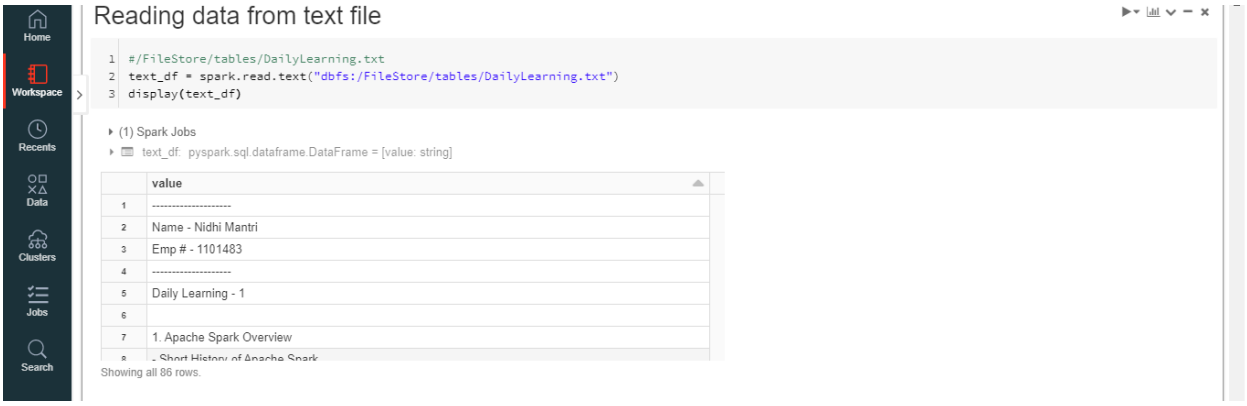
File Edit Format View Help

OrderID	OrderDate	CustomerName	PhoneNumber	DeliveryAddr
63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2 1319
63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2 1319
63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2 1319
63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2 1319
63969	2020-12-01	Tailspin Toys (Sans Souci, SC)	(803) 555-01	
63969	2020-12-01	Tailspin Toys (Sans Souci, SC)	(803) 555-01	
63969	2020-12-01	Tailspin Toys (Sans Souci, SC)	(803) 555-01	
63970	2020-12-01	Wingtip Toys (Birds, IL)	(217) 555-01	

Let's read first type of data using text format –

```
text_df = spark.read.text("dbfs:/FileStore/tables/DailyLearning.txt")

display(text_df)
```

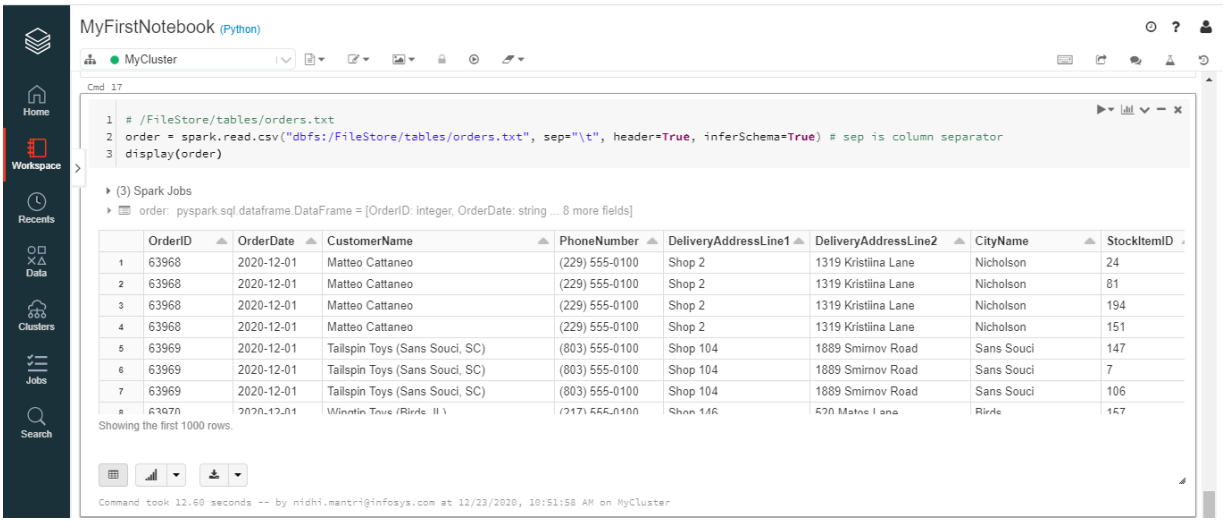


This text format will consider each new line data as a separate record, and to get some meaningful insights from such type of data, we need to process it.

Now, let's discuss second type of data – *why I mentioned earlier that we need csv format to read such type of data?* – Because we can't specify column separator type of things in text format. It only read data line by line. Therefore, we need csv format. Let's try it –

```
order = spark.read.csv("dbfs:/FileStore/tables/orders.txt", sep="\t", header=True, inferSchema=True) # sep is column separator

display(order)
```



	OrderID	OrderDate	CustomerName	PhoneNumber	DeliveryAddressLine1	DeliveryAddressLine2	CityName	StockItemID
1	63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2	1319 Kristiina Lane	Nicholson	24
2	63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2	1319 Kristiina Lane	Nicholson	81
3	63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2	1319 Kristiina Lane	Nicholson	194
4	63968	2020-12-01	Matteo Cattaneo	(229) 555-0100	Shop 2	1319 Kristiina Lane	Nicholson	151
5	63969	2020-12-01	Tailspin Toys (Sans Souci, SC)	(803) 555-0100	Shop 104	1889 Smirnov Road	Sans Souci	147
6	63969	2020-12-01	Tailspin Toys (Sans Souci, SC)	(803) 555-0100	Shop 104	1889 Smirnov Road	Sans Souci	7
7	63969	2020-12-01	Tailspin Toys (Sans Souci, SC)	(803) 555-0100	Shop 104	1889 Smirnov Road	Sans Souci	106
8	63970	2020-12-01	Wingtip Toys (Birds, IL)	(217) 555-0100	Shop 146	520 Matos Lane	Birds	157

We got the correct table from this text file using csv format. Are you Still want to know? What will be the output when I use text format? Let me try it –

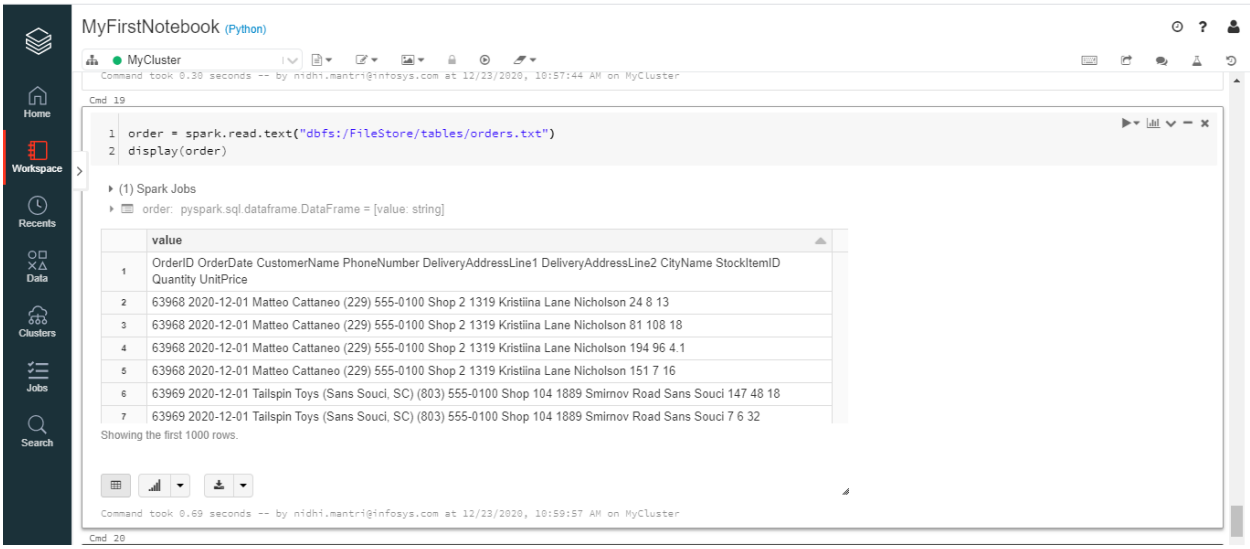
```
order = spark.read.text("dbfs:/FileStore/tables/orders.txt", sep="\t", header=True)

display(order)
```

See, we got the error. Because text data has no separator, header kind of things. Let me remove this and see the output.

```
order = spark.read.text("dbfs:/FileStore/tables/orders.txt")

display(order)
```



The output is just each new line as a separate record. So, using csv format for this type of cases is helpful and timesaving.

So, Yeahhh!!! With this we have successfully read the data using various formats(csv, json, table, parquet and text) in Pyspark.

Keep Learning!!!

- Nidhi Mantri

Specialist Programmer – Big Data