

## **OBJECTIVE**

The objective of this lab is to introduce Apache Spark, using the Databricks cloud Spark environment. Together, we will learn how to:

- Set up the Databricks Community Edition
- Upload data to the Spark environment
- Create tables and DataFrames using SQL, Python, and Scala
- Run queries against the tables and DataFrames using SQL, Python, and Scala

## **PREREQUISITES**

Before attempting this lab, it is best to read the textbook and lecture material covering the objectives listed above. While this lab shows you how to create and use these constructs in SQL, the lab does not explain in full the theory behind the constructs, as does the lecture and textbook.

## **REQUIRED SOFTWARE**

The examples in this lab will execute in the Databricks cloud environment. The screenshots in this lab display execution of Apache Spark in the Databricks environment.

## **LAB COMPLETION**

Use the submission template provided in the assignment inbox to complete this lab.

# SECTION ONE

## OVERVIEW

In Section One, we cover some introductory Apache Spark information and we will set up the Databricks Community Edition cloud environment for Apache Spark. This includes a single node with 6 GB of storage and online notebooks for submitting commands.

### Apache Spark Introduction

Apache Spark is a framework capable of running code in parallel across many computers. Developers at UC Berkeley's [AMPLab](#), an approximately six-year private and academic partnership, created Spark.

The Spark Ecosystem consists of the Spark Core Engine, Spark Libraries and APIs, applications, environments, and data sources. The Spark Core Engine has several functions: storage system interaction, memory management, fault recovery, and application distribution and monitoring. Spark provides programming interfaces for Scala, Python, Java, SQL, and R. Spark users interact with these programming interfaces through notebooks like Jupyter. Spark provides libraries for SQL (Spark SQL), streaming applications (Spark Streaming), machine learning (MLlib), and graph databases (GraphX). A Spark application refers to the code submitted to the Spark environment, either a single machine or a cluster. Environment refers to where Spark is hosted, Apache Mesos, Hadoop YARN, Docker, Kubernetes, or others. Data sources refers to data format and location. Some common compatible formats are CSV, JSON, MySQL, Hadoop HDFS, Hive, Parquet, and Amazon S3.

Spark runtime architecture consists of several components. These roles differ slightly depending on the environment that hosts Spark.

- The Spark Master manages the standalone installation or cluster unless Spark is installed in a YARN or Mesos cluster. In that case, the YARN or Mesos master manages the cluster and allocates resources.
- The Spark Application is the code submitted by a Spark user to run on Spark.
- The Spark Driver node runs the Spark Driver process. Spark users interact with the Spark Driver to launch their Spark application. The Spark Driver responds to the Spark user's application. The Spark Driver breaks the job into stages and the stages into tasks, and distributes and schedules the tasks on Spark Executors.
- The Spark Worker launches Spark Executor JVMs as directed by the Spark Master or other cluster manager.
- The Spark Executor is a JVM container on a Spark Worker. The Spark Executor is allocated a certain number of processor cores and memory. The Spark Executor executes tasks received from the driver and caches data partitions.

The SparkSession provides a command line interface for submitting Spark commands and applications. The Databricks environment, which will be described in more detail below, provides a notebook interface with a pre-configured SparkSession.

Resilient Distributed Datasets (RDDs) are the original Spark data structure. RDDs are immutable, partitioned, distributed, fault-tolerant collections of data. RDDs provide fault tolerance by tracking their lineage instead of relying on replication. RDDs are able to rebuild lost partitions using information in other partitions.

RDDs are for unstructured data. Spark developers added two additional data abstractions, DataFrames and Datasets, for structured data. Uppercase 'D' Dataset is a specific term in Spark, as opposed to the general lowercase 'd' dataset term. DataFrames and Datasets are built on top of RDDs and they inherit RDD functionality.

Developers can deploy Spark on a single computer or in a cluster of up to thousands of interconnected computers. Spark users submit jobs or applications, essentially blocks of code, to Spark. Spark infrastructure manages the application and either returns the results to the calling computer or writes the results to durable storage. Spark can use its own built-in resource scheduler to divide and distribute a Spark job in a cluster. Spark can also rely on Apache Mesos, Hadoop YARN, or other cluster managers.

## Databricks Spark Environment Introduction

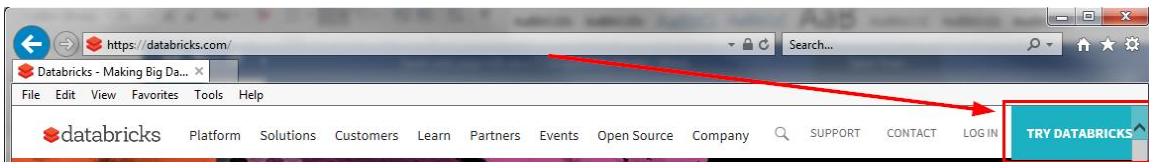
The Databricks Community Edition includes several tutorials for learning the Databricks environment and for running Spark.

1. From the [Apache Spark Getting Started](#) tutorial, review the following sections. They provide quick illustrations of Spark functionality. They don't require a deep dive, only a recognition of what each command accomplished.
  - a. **Apache Spark Quickstart** tutorial. This tutorial reviews Spark interfaces (RDD, DataFrame, Dataset) and demonstrates some commands. These commands can be copied to or re-typed in your notebook.
    - i. NOTE: Spark includes a concept called Magic Functions to tell the interpreter which language is being used. When a workbook is created, it is assigned a language (SQL, Java, Scala, Python, R, etc). It is still possible to run other languages in the notebook, using Magic Functions (%sql, %python, %java, %scala, %r).
    - ii. Another confusing aspect of Spark is the name of the Dataset interface, used in conjunction with the general term, datasets. Beware capital Dataset versus lowercase dataset.
  - b. **DataFrames** (the notebook illustrated in this example was created with SQL as the specified language)
    - i. Load sample data

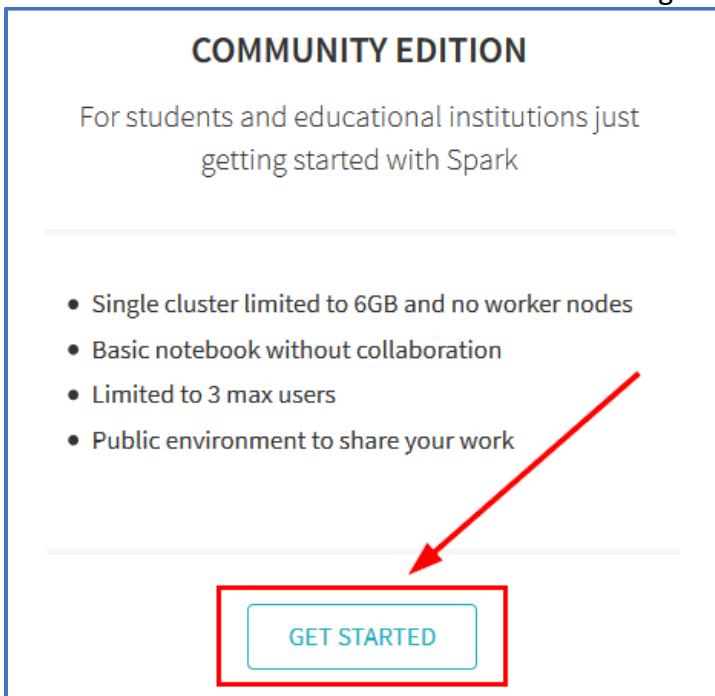
1. Note the use of the Magic Function
2. Note also the options for reading in a CSV file
  - ii. View the DataFrame
  - iii. Run SQL queries
- c. **Datasets** (the notebook illustrated in this example was created with Scala as the specified language)
  - i. Create sample data
  - ii. Load sample data
  - iii. View the Dataset

## STEPS FOR ENABLING DATABRICKS ACCESS

1. Open URL: <https://databricks.com/>
2. Click TRY DATABRICKS



3. Click "Get Started" to create a free account using the Community Edition



4. Fill out the Sign Up form
5. Acknowledge the Terms of Service

6. Databricks will send a confirmation message to your email address similar to the message below.

## Welcome to Databricks Community Edition!

Databricks Community Edition provides you with access to a free micro-cluster as well as a cluster manager and a notebook environment - ideal for developers, data scientists, data engineers and other IT professionals to get started with Spark.

We need you to verify your email address by clicking on [this link](#). You will then be redirected to Databricks Community Edition!

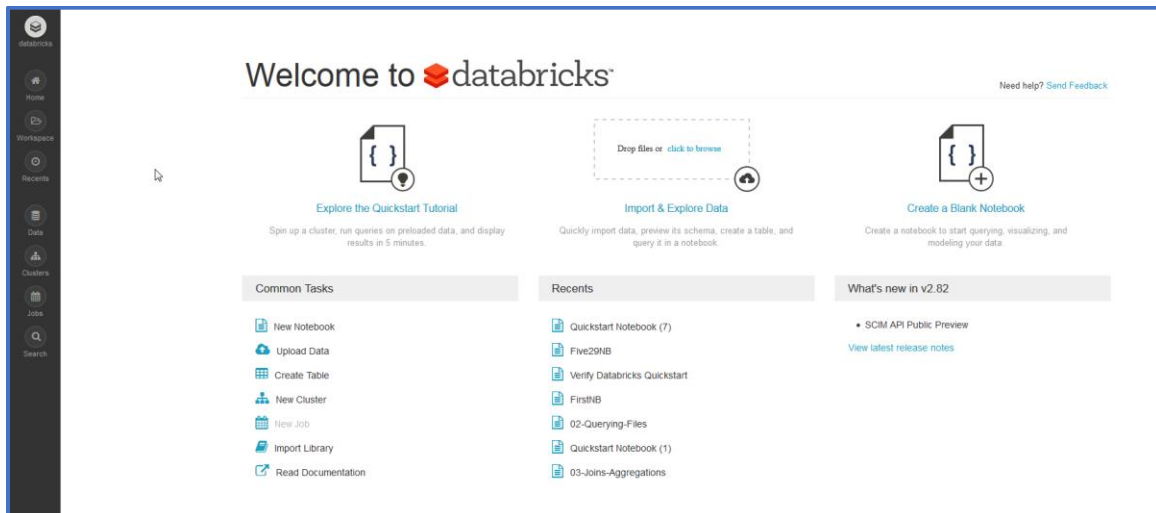
**Get started by visiting:** <https://accounts.cloud.databricks.com/signup/validate?emailToken=b202f81b0f3db30a7b7a4014df18a670&ce=true>

If you have any questions, please contact [feedback@databricks.com](mailto:feedback@databricks.com).

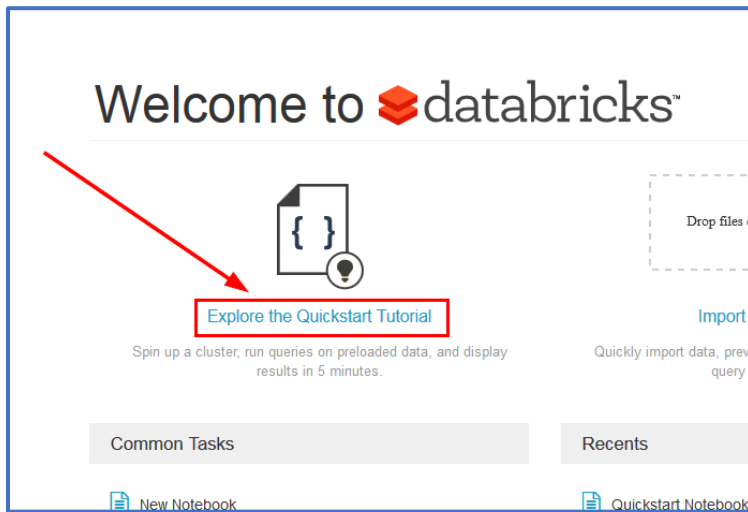
- The Databricks Team

7. Click “this link” in the message and the Databricks Community Edition login page will open. Set up your password.

8. Log in to the Databricks Community Edition home page. You can find it by navigating to <https://community.cloud.databricks.com/>

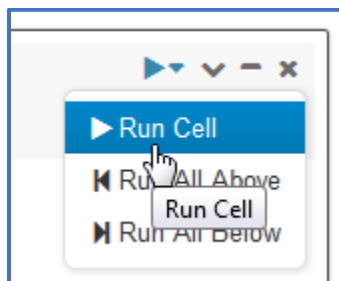


9. Click “Explore the Quickstart Tutorial”



10. Read through the tutorial with the following notes in mind

- a. It is more helpful to execute the commands one cell at a time. Review the results of each command.
- b. To run commands individually, place the cursor in the command cell and click Run Cell using the controls on the upper right side of the particular command cell. Alternatively, place cursor next to the command and press Ctrl + Enter.



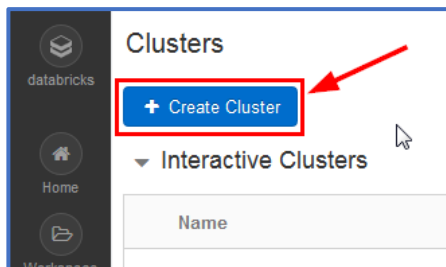
## SECTION TWO

### OVERVIEW

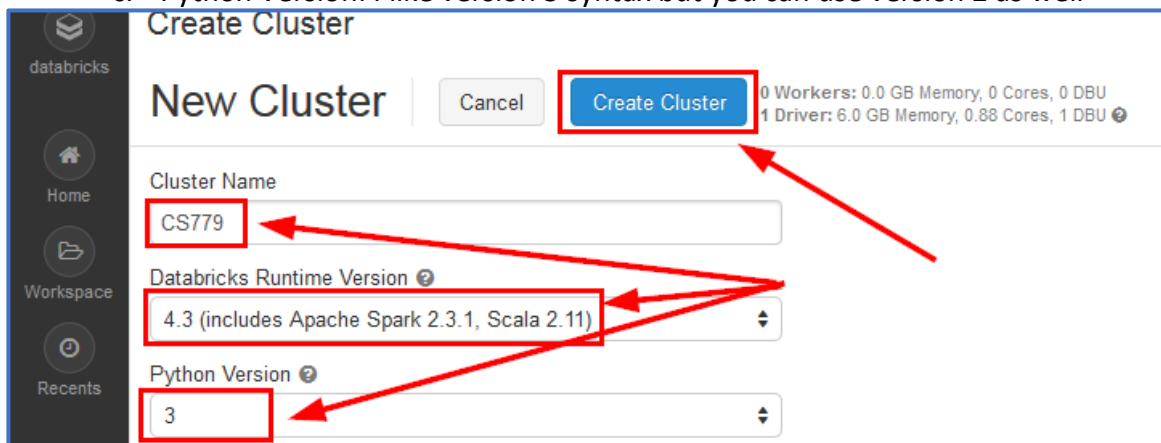
In Section Two, we will demonstrate Apache Spark in use on the Databricks platform. The demonstration will include downloading a dataset from boston.gov and uploading data to the Databricks platform, creating a table, and querying the table.

### STEPS

1. Log into your Community Databricks account. The Welcome to Databricks screen should be displayed, along with a toolbar along the left side of the screen.
2. Click the Clusters tool in the left toolbar. Make sure no other clusters are running (i.e. from the tutorial)
3. Click Create Cluster

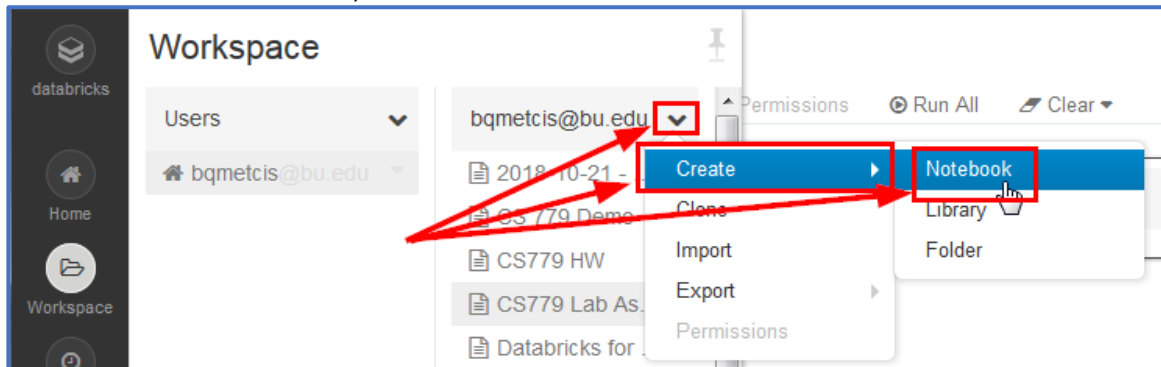


4. Set the parameters and click Create Cluster
  - a. Cluster Name: choose a name
  - b. Databricks Runtime Version: can leave as default
  - c. Python Version: I like version 3 syntax but you can use version 2 as well



Note that if the create cluster button is not available – it means that you have a running cluster.

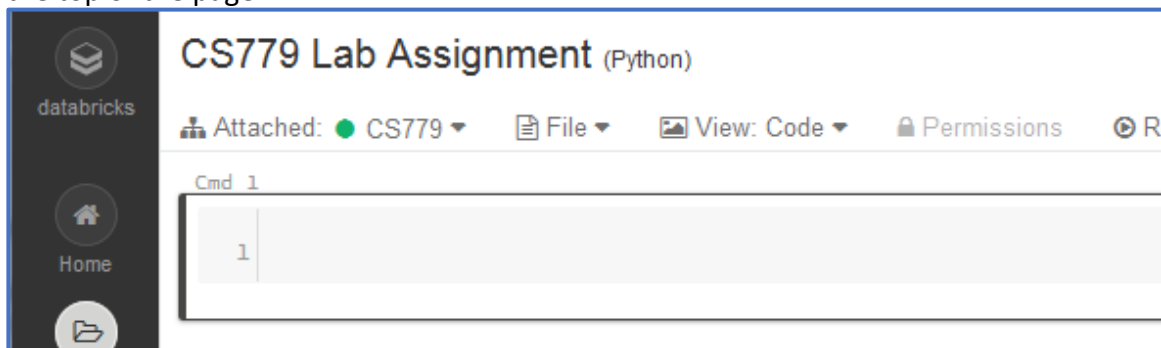
5. Click the Workspace tool in the toolbar on the left
6. Click the drop down arrow next to your account name. Click on Users first to see your account. Then click Create, Notebook.



7. Name the notebook. The Language (Python) and Cluster fields can be left as default. Click Create.

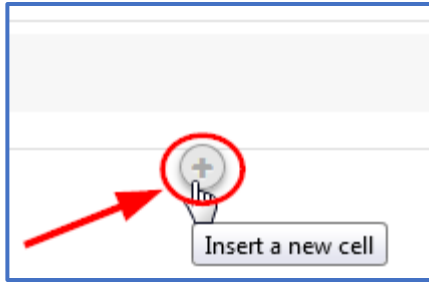
A screenshot of the 'Create Notebook' dialog box. It has three input fields: 'Name' with the value 'CS779 Lab Assignment', 'Language' with the value 'Python', and 'Cluster' with the value 'CS779 (6 GB, Running, 4.3 (incl)'. At the bottom right, there are two buttons: 'Cancel' and 'Create'. A red box highlights the 'Create' button, and a red arrow points to it from the 'Language' field.

8. Your newly created notebook will be ready to use. It will have a single blank cell at the top of the page.



9. To add new cells, hover near the bottom middle of the cell, until the '+' icon appears.





10. Download the 311 Service Request data set from the City of Boston to your local drive
11. Go to the 311 Dataset URL: <https://data.boston.gov/dataset/311-service-requests>
12. Click the CSV option for downloading (the screen-shot may look slightly different)

Dataset

Topics


Activity Stream



Showcases


## 311 SERVICE REQUESTS

This data set includes all channels of engagement in which a service request is created.  
Refer to this link to learn more about **BOS:311**: <https://www.cityofboston.gov/311/>



### DATA AND RESOURCES

 311 Service Requests 🔥

 Preview  **DOWNLOAD**

 CRM Value Codex 🔥

Helps in interpreting actual values given in columns

 Preview  **DOWNLOAD**

### TAGS

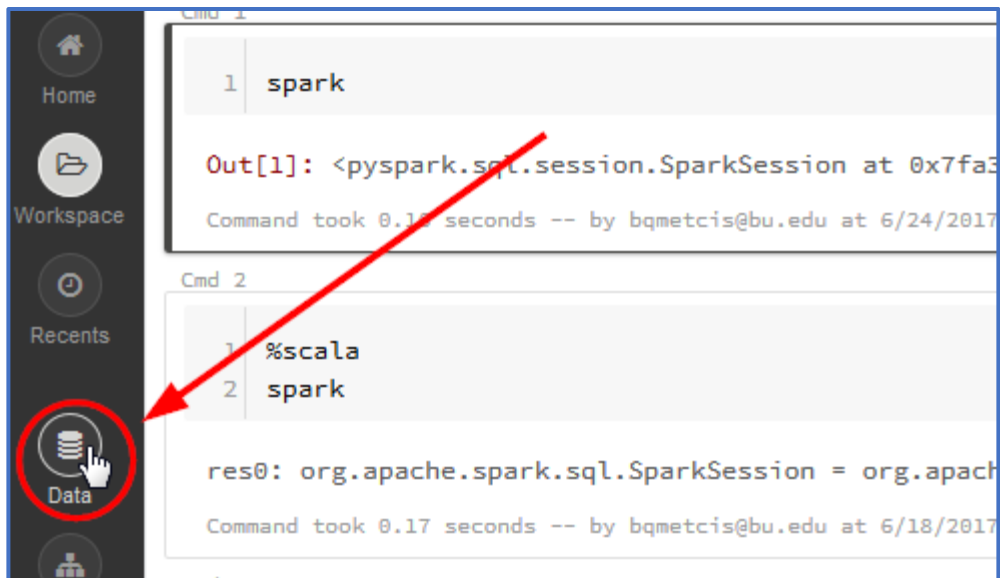
311

CRM

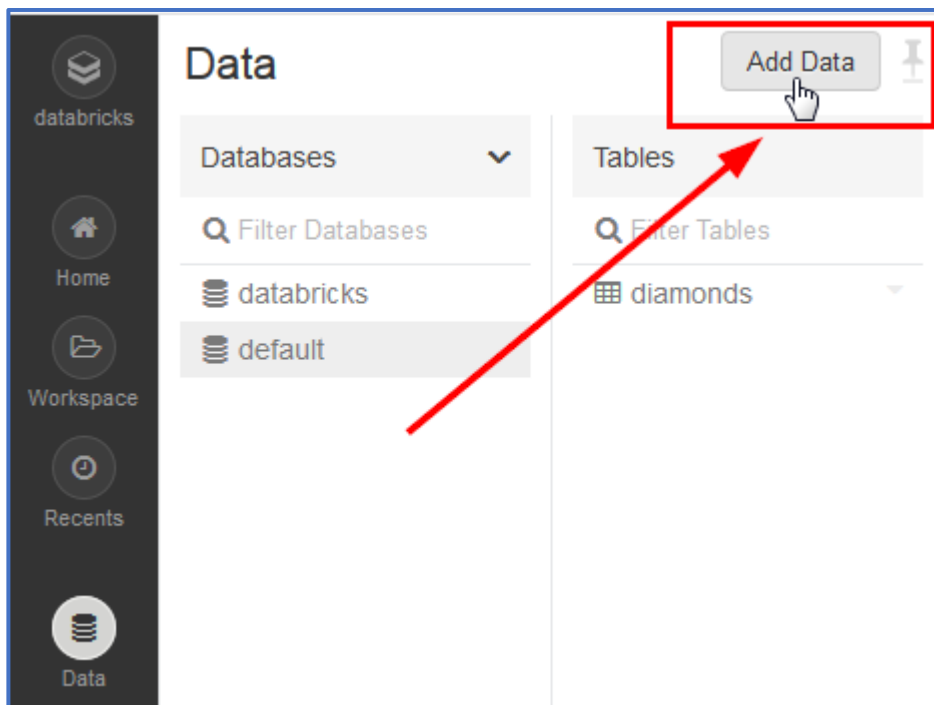
Case management

city services

13. Download the 311 service requests dataset to your local computer. Note that the file size is approximately 725 MB.
14. Now let's upload data to Databricks
15. Click the Data tool in the upper left



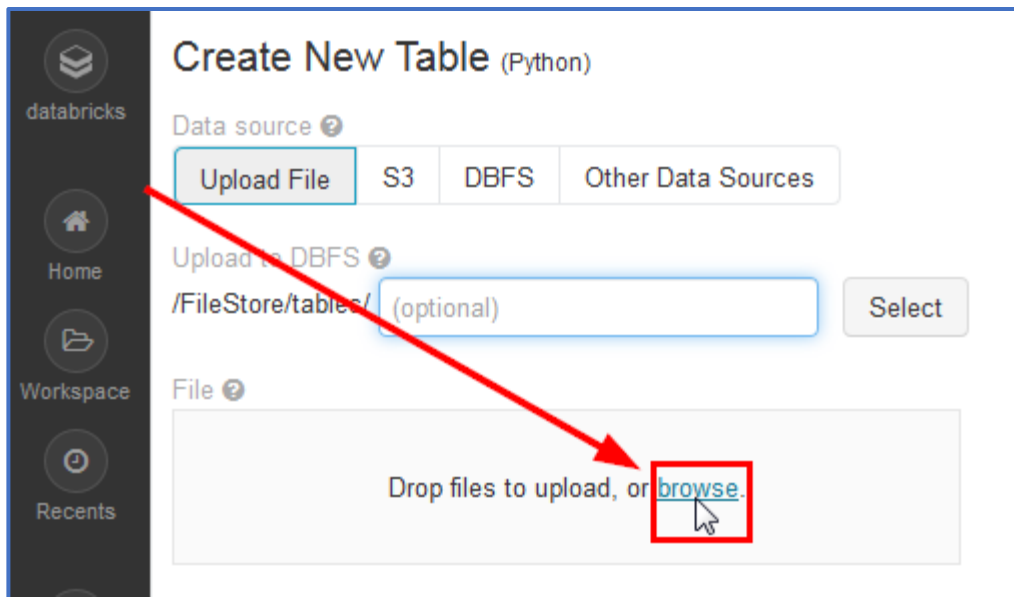
16. Click Add Data



17. Leave the Data source set to "Upload File"

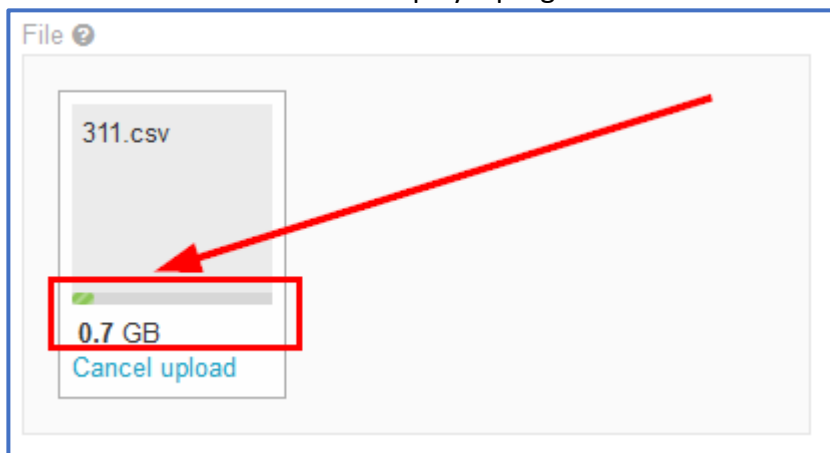
18. Leave the "Upload to DBFS" path as the default, "/FileStore/tables/"

19. Click browse to upload a file from the local drive

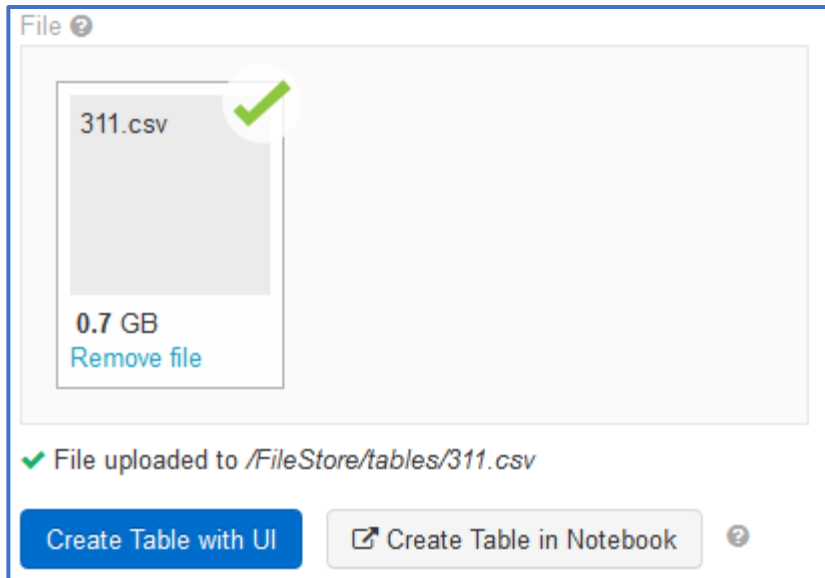


20. Select the file from your local file explorer and click Open

21. The Databricks console will display a progress bar



22. If the file is uploaded successfully, the Databricks console will display a green check



23. Make note of the file path (/FileStore/tables)
24. Go back to the notebook which you created earlier in step 6. You will find your notebook in Workspace under your name.
25. To check the tables folder for uploaded and other files, here are a couple of commonly used commands.
  - a. The “%fs” magic command activates Linux file system commands. When you run the command below – you should be able to see the files that you added

```
Cmd 8
1 %fs
2 ls /FileStore/tables/
```

- b. Databricks uses its own file system called DBFS. The dbutils command has different options for accessing DBFS. Databricks lists additional commands in their online user guide: <https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html#dbfs>.

```
Cmd 9
1 dbutils.fs.ls("/FileStore/tables/")
```

- c. No need to do this as part of the steps, but If you would like to remove a file, here is the Databricks file system (DBFS) command

```
Cmd 10
1 dbutils.fs.rm("/FileStore/tables/lagan_311_service_requests.csv")
```

26. Create a data structure from the uploaded file. This is an example of a Scala command to create a DataFrame from a CSV file.

```
%scala -- Magic command for Scala syntax
val events311_default = spark -- Val indicates a static value in Scala
    .read -- Reads data into the Val
    .csv("/FileStore/tables/lagan_311_service_requests.csv") -- Identifies data as
CSV, includes path
```

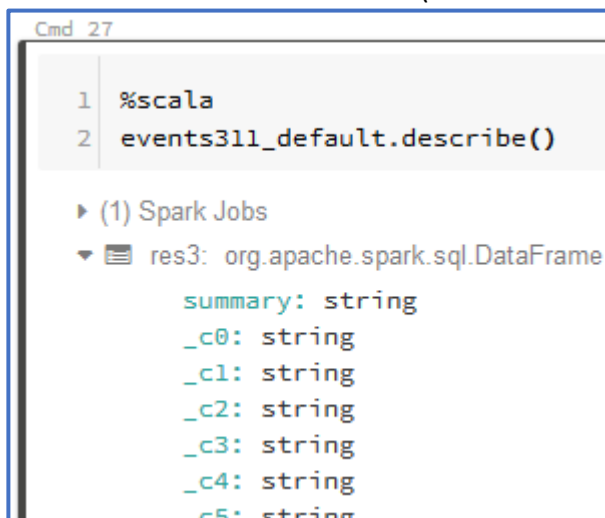
In this screenshot below – the file name we are loading has a slightly different name but your result should look the same.



```
Cmd 25
1 %scala
2 val events311_default = spark
3   .read
4   .csv("/FileStore/tables/311.csv")

▶ (1) Spark Jobs
▶ events311_default: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string, _c2: string, _c3: string, _c4: string, _c5: string]
events311_default: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string, _c2: string, _c3: string, _c4: string, _c5: string]
Command took 3.86 seconds -- by bqmetcis@bu.edu at 10/23/2018, 8:16:10
```

27. Check the data structure of the DataFrame using the following command. Note in the output that columns have generic names and all data formats are string.
28. Provide a screenshot of the loaded data frame – the screenshot needs to show your user name and the date loaded (last line of the output page)



```
Cmd 27
1 %scala
2 events311_default.describe()

▶ (1) Spark Jobs
▼ res3: org.apache.spark.sql.DataFrame
  summary: string
  _c0: string
  _c1: string
  _c2: string
  _c3: string
  _c4: string
  _c5: string
```

29. How many c fields are listed? \_\_\_\_\_

30. Try querying the DataFrame using SQL. It will not work because SQL requires a table to query. The Scala command from #26 above creates a DataFrame, but not a SQL table. SQL can query tables but not DataFrames. When you try the command below you will get an error.

```
Cmd 33
1 %sql
2 Select * from events311_default
3 Limit 10

Error in SQL statement: AnalysisException: Table or view not found:
```

31. The DataFrame can be queried using Scala. Note the hints Databricks provides by pressing Tab when composing a command.

```
Cmd 34
1 %scala
2 events311_default.s|

sample
schema
select
selectExpr
```

The command below uses Scala to query the events311\_default DataFrame created in step #26.

```
Cmd 34
1 %scala
2 events311_default.select("_c0", "_c1", "_c2", "_c3").show(10)|
```

► (1) Spark Jobs

_c0	_c1	_c2	_c3
CASE_ENQUIRY_ID	open_dt	target_dt	closed_dt
101002709389	2018-10-21 00:01:03	2018-10-23 08:30:00	null
101002709388	2018-10-20 23:56:33	2018-10-23 08:30:00	null
101002709387	2018-10-20 23:46:50	null	null
101002709386	2018-10-20 23:41:50	2018-10-23 08:30:00	null

32. Update the command above to show rows c0, c1,c2,c3,c4 and c5 showing first 15 rows, paste your screenshot clearly showing your user name and date and time executed.

33. Now let's create a table using Scala.

```
Cmd 35
1 %scala
2 events311_default.write.saveAsTable("events311_default_tbl")

▶ (1) Spark Jobs

Command took 25.89 seconds -- by bqmetcis@bu.edu at 10/23/2018, 9:20:18 PM on CS779 HW
```

34. This table created from the DataFrame can be queried using SQL like in the command below. The Limit parameter returns only the first 5 rows from the table.

```
Cmd 33
1 %sql
2 Select * from events311_default_tbl
3 Limit 5

▶ (1) Spark Jobs
```

_c0	_c1	_c2	_c3	_c4	_c5	_c6
101001976343	2016-12-26 13:06:00	null	2016-12-30 16:30:46	ONTIME	Closed	Case Closed. Closed date : 2016-12-30 16:30:46.627 Bulk Item Automation
101001976341	2016-12-26	null	2016-12-26	ONTIME	Closed	Case Closed.


35. Write a query which shows top 5 records for Trees which should be filtered in \_c9 field. Take a screen-shot of your query result – make sure to show your user name and time-stamp in the screen-shot.

36. The events311\_default\_tbl table can also be queried using Scala, using the command below.

```
Cmd 36

1 %scala
2 val evt311_def = spark.sql("""
3   Select * from events311_default_tbl
4   Limit 5
5   """)
6
7 display(evt311_def)
```

► (1) Spark Jobs

►  evt311\_def: org.apache.spark.sql.DataFrame = [\_c0: string, \_c1: string ... 27 more fields]

_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7
101001976343	2016-12-26 13:06:00	null	2016-12-30 16:30:46	ONTIME	Closed	Case Closed. Closed date : 2016-12-30 16:30:46.627 Bulk Item Automation	Schedule a Bulk Item Pickup

37. SQL can create a table from the CSV file on the cluster. The option inferSchema attempts to identify the data type and header identifies the header row. Recall the path of the file when you run the command below as your file name may differ.

```
Cmd 46

1 %sql
2
3 CREATE TABLE IF NOT EXISTS events311_sql_tbl
4 USING csv
5 OPTIONS (
6   path '/FileStore/tables/311.csv',
7   inferSchema 'true',
8   header 'true'
9 )
```

► (2) Spark Jobs

38. Verify the table format using the following command. Notice the column titles and data types are identified.



Cmd 47

```
1 %sql
2 Describe events311_sql_tbl
```

col_name	data_type	comment
CASE_ENQUIRY_ID	bigint	null
open_dt	timestamp	null
target_dt	timestamp	null
closed_dt	timestamp	null
OnTime_Status	string	null
CASE_STATUS	string	null
CLOSURE_REASON	string	null
CASE_TITLE	string	null
SUBJECT	string	null

- a. Spark has built-in functions for querying including *year*, *month*, and *day* for isolating parts of a timestamp field.

39. Write a query which shows top 5 records where the Reason is Signs & Signals. Take a screen-shot of your query result – make sure to show your user name and time-stamp in the screen-shot.
40. Write an aggregate query of your choice based on the above data set and show the result both in table form and utilizing the chart tool. The aggregate you select needs to be thought out as the results need to be meaningful, meaning if you just write an aggregate to return results that don't have any meaning this won't be helpful.
41. Very briefly explain what you have discovered based on your data set from the query above.