# **Overview of the Assignment:**

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 semi-structured data to the Spark environment
* Create tables and DataFrames using SQL, and Python
* Run queries against the tables and DataFrames using SQL, Python

**PREREQUISITES**

Before attempting this lab, it is best to read the lecture material covering the objectives listed above. While this lab shows you how to create and use these constructs, the lab does not fully 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 SUBMISSION**

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

# **Part 1: Introduction to Apache Spark**

In Part 1 One, we cover some introductory Apache Spark information.

## Apache Spark Introduction

Apache Spark is a framework capable of running code in parallel across many computers. Developers at UC Berkeley’s [AMPLab](https://amplab.cs.berkeley.edu/), 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.

* 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](https://docs.databricks.com/getting-started/spark/index.html), 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.
   1. **Apache Spark Quickstart tutorial**. This tutorial reviews Spark interfaces (RDD, DataFrame, Dataset) and demonstrates some commands. These commands can copied to or re-typed in your notebook.
2. 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).
3. iAnother 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.
   1. **DataFrames** (the notebook illustrated in this example was created with SQL as the specified language)
      1. Load sample data Page 4 of 17: Note the use of the Magic Function. Note also the options for reading in a CSV file
      2. View the DataFrame
      3. Run SQL queries
   2. **Datasets** (the notebook illustrated in this example was created with Scala as the specified language)
      1. Create sample data
      2. Load sample data
      3. View the Dataset

# **Part 2: Getting started with Apache Spark on Databricks**

In part 2 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

1. Open URL: <https://community.cloud.databricks.com/login.html>
2. Click Sign-up, enter your info, make sure to select Get Started with Community Edition at the bottom.

**Graphical user interface, text, application

Description automatically generated**

Once your account is set up, you can log in Log in to the Databricks Community Edition home page. You can find it by navigating to <https://community.cloud.databricks.com/>

Data bricks home page

Graphical user interface, text, application, website

Description automatically generated

1. Complete the QuickStart tutorial (Guide: QuickStart tutorial from home page) which will run through how to use the Databicks interface

Graphical user interface, text, application, website

Description automatically generated

Within the tutorial you will learn how to

* Create a cluster (it will take a little while for the cluster to start up)
* Attach the notebook
* And run through a few commands

Tips:

* Graphical user interface, application

  Description automatically generatedif you ever get lost, on the left-hand menu, select Workspace, and you will see the QuickStart notebook
* Graphical user interface, text, application, chat or text message

  Description automatically generatedIt is more helpful to execute the commands one cell at a time. Review the results of each command.
* 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.

# **Part 3: Working with Spark**

In part 3, 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.

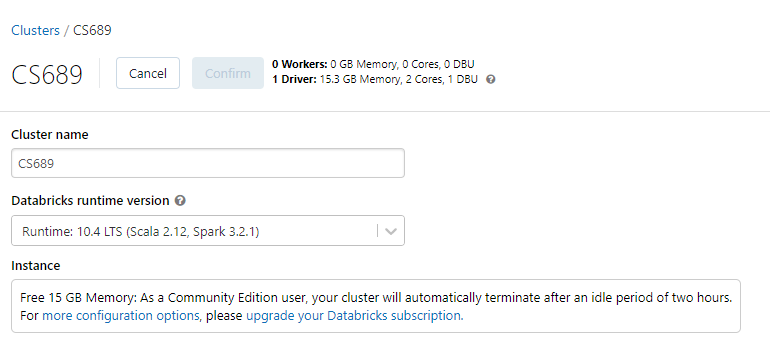
1. Log into your Community Databricks account. The Welcome to Databricks screen should be displayed, along with a toolbar along the left side of the screen.

2. Click the Compute tool in the left toolbar.  Make sure no other clusters are running (i.e. from the tutorial), you can open them up and stop them.

3. Click Create Cluster

Graphical user interface, text, application, email

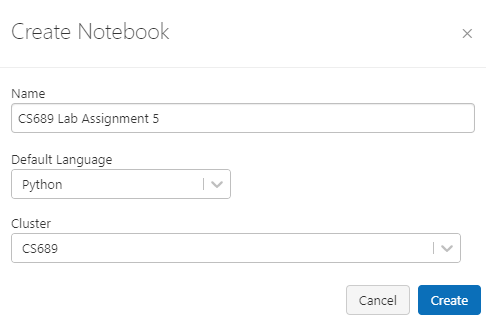
Description automatically generated

1. Set the parameters and click Create Cluster
   1. Cluster Name: choose a name, i.e. CS689
   2. Databricks Runtime Version: can leave as default
   3. 
   4. Note that if the create cluster button is not available – it means that you have a running cluster which should be shut down first.
2. Click the Create tool in the toolbar on the left and select Notebook

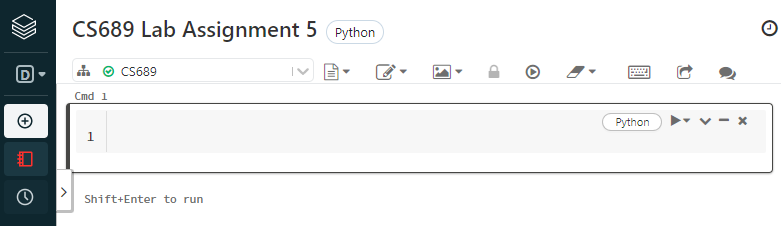
Graphical user interface, application

Description automatically generated

1. Name the notebook. The Python and Cluster fields can be set as shown, click Create.

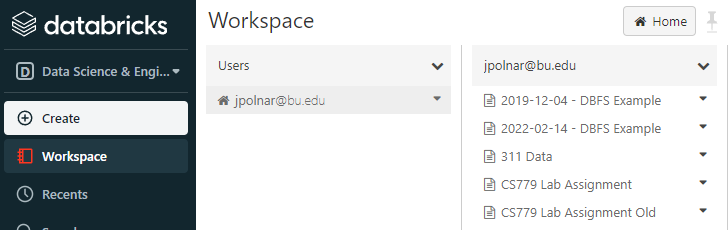


1. Your newly created notebook will be ready to use. It will have a single blank cell at the top of the page.

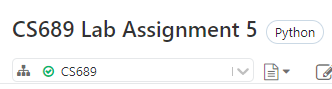


**How to get to your notebook if you come back to the assignment at a later time**:

If you need to come back to your notebook, look for it under Resents  or Workspace  at which point you will need to select your username under Users, and you will see the notebook.



You may also need to re-attach the cluster (again if you are coming back to this assignment later)



1. To add new cells on the notebook, hover near the bottom middle of the cell, until the ‘+’ icon appears.

Graphical user interface, text, application

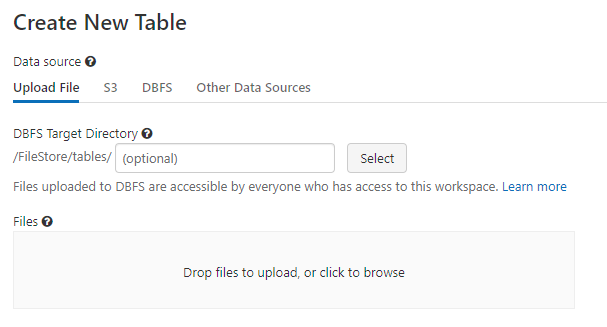
Description automatically generated

1. Download the JSON data set from the City of Boston to your local drive which is included with this assignment. Alternatively, you can download the JSON file at this location: https://data.boston.gov/dataset/wicked-free-wifi-locations
2. Upload the file into Databricks – click on the Data tool in the left menu and select Create Table

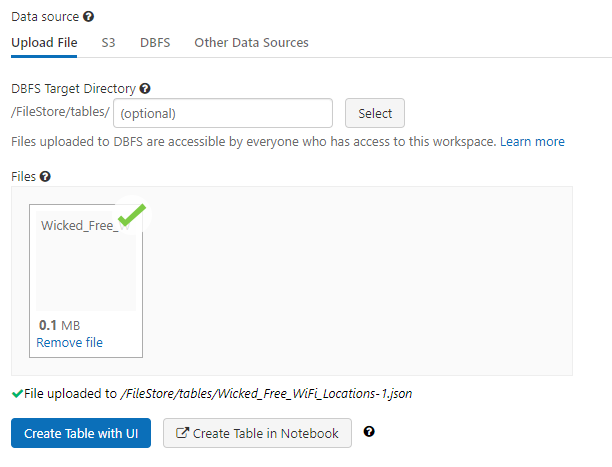
Graphical user interface, application

Description automatically generated

1. In the Create New Table Leave the Data source set to “Upload File”.
   1. Make sure to rename your file on your local computer to something like this before uploading (note the year): Wicked\_Free\_WiFi\_Locations.json
   2. Under Files area drag the file or click to browse and attach it (in the image below, the upload is in process). The upload may take some time.



1. Once the file is uploaded successfully, the Databricks console will display a green check. There is a way to create the table via the UI, however we are going to explore how to first create a data frame and then create the table from the data frame. Make note of the file path in the example below it is (/FileStore/tables/Wicked\_Free\_WiFi\_Locations.json)



1. Go back to the notebook which you created earlier in step 6. You will find your notebook in Workspace under your name. Note that you can also explore create table in Notebook which will give you the example commands we will work with next.
2. **QUESTION FOR YOU TO ANSWER** We will create the table via dbutils command to make it a data frame.

To check the tables folder for uploaded and other files, here are a couple of commonly used commands. 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.

%fs

ls /FileStore/tables

**Provide a screenshot of the loaded file–** **the screenshot needs to show your username and the date**

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>

For example, if you made a mistake and want to remove a file, try this command, replace the name of the file below:

%python

dbutils.fs.rm("/FileStore/tables/filetoremove.csv")

1. **QUESTION FOR YOU TO ANSWER** Now let’s create a dataframe from the uploaded file.

Execute the following command in a new cell. Note that we are calling the new data frame Wifi\_df and we are loading json data from it.

Wifi\_df = spark.read.format("json").load("/FileStore/tables/Wicked\_Free\_WiFi\_Locations.json")

Wifi\_df.display()

**Provide a screenshot of the command above–** **the screenshot needs to show your username and the date**

1. **QUESTION FOR YOU TO ANSWER**: Check the data structure of the DataFrame using the following command.

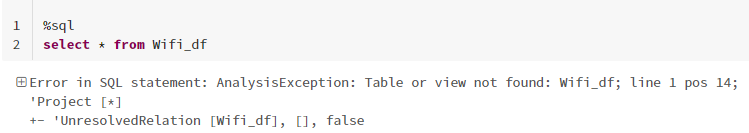
Wifi\_df.printSchema()

**Provide a screenshot of the command above–** **the screenshot needs to show your username and the date**

1. **QUESTION FOR YOU TO ANSWER**: Briefly describe the structure of the data frame:
2. Try querying the DataFrame using SQL. It will not work because SQL requires a table to query. The Scala command from 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.

%sql

select \* from Wifi\_df



1. Let’s query the table via Python, specifically selecting the specific columns within properties.

Wifi\_df.select('properties.\* ').show()

You do not need to provide a screen-shot just yet

1. **QUESTION FOR YOU TO ANSWER**: Update the command above to explicitly show columns from the geometry and properties objects of your choice. Before deciding on which columns to select, review questions 24 and 26. **Extra Credit Option (1 point):** Note how one of the columns in the original data frame is an array of coordinates. Look to use the explode function to extract those coordinates into the command above.

**Provide a screenshot of the command above–** **the screenshot needs to show your username and the date**

Wifi\_df.select('geometry.coordinates','properties.ObjectId', 'properties.device\_address','properties.device\_connectedto','properties.device\_lat', 'properties.device\_long','properties.device\_serial', 'properties.etl\_updatedtimestamp', 'properties.neighborhood\_id', 'properties.neighborhood\_name').show()

Table

Description automatically generated with low confidence

Extra credit code:

from pyspark.sql.functions import explode

Wifi\_df.select(explode("geometry.coordinates").alias("coordinates"))

Graphical user interface, text, application

Description automatically generated

1. **QUESTION FOR YOU TO ANSWER**: Now let’s create a table for use with SQL, note to replace the select statement with appropriate columns.

Wifi\_df.select('geometry.coordinates','properties.ObjectId', 'properties.device\_address','properties.device\_connectedto','properties.device\_lat', 'properties.device\_long','properties.device\_serial', 'properties.etl\_updatedtimestamp', 'properties.neighborhood\_id', 'properties.neighborhood\_name').write.mode("overwrite").saveAsTable("Wifi\_tbl")

**Provide a screenshot of the command above which you have modified–** **the screenshot needs to show your username and the date**

Graphical user interface, text, application

Description automatically generated

1. **QUESTION FOR YOU TO ANSWER**: This table created from the DataFrame can be queried using SQL like in the command below. Note: adjust the query to limit the result to only the first 10 rows.

%sql

select \* from Wifi\_tbl limit 10

**Provide a screenshot of the command above which you have modified–** **the screenshot needs to show your username and the date**

**Graphical user interface, text, application, table, email

Description automatically generated**

**Now you are in a familiar place!**

1. **QUESTION FOR YOU TO ANSWER**: Write an aggregate query of your choice based on the above data set using SQL and show the result both in table form and utilizing the chart tool.
   1. 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. Impress us with a complex query for a higher score!
   2. Try what we have leaned in analytical functions so far.
   3. Use a built-in function to transform some of the data, for example extract date from timestamp, or whatever else you might think is appropriate.

**Provide a screenshot of the command above which you have modified–** **the screenshot needs to show your username and the date**

**%sql**

**SELECT**

**neighborhood\_name as neighborhood,**

**count(\*) AS num\_devices,**

**avg(device\_lat) OVER (PARTITION BY neighborhood\_id) AS avg\_lat,**

**avg(device\_long) OVER (PARTITION BY neighborhood\_id) AS avg\_long,**

**date(etl\_updatedtimestamp) as date**

**FROM Wifi\_tbl**

**GROUP BY neighborhood\_id,device\_lat,device\_long, neighborhood\_name, date(etl\_updatedtimestamp)**

Table

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

1. **QUESTION FOR YOU TO ANSWER**: Very briefly explain what you have discovered based on your data set from the query above.

The query above provides a summary of the total number of devices in each neighborhood and the average location of the devices in that neighborhood at different points in time. These results will help to identify neighborhoods with higher or lower densities of devices. The query above also extracts the date from the etl\_updatedtimestamp columns. Which will help to analysis the data by date.

1. **QUESTION FOR YOU TO ANSWER**: Write a second aggregate query of your choice based on the above data set using Python and show the result both in table form and utilizing the chart tool.

**Provide a screenshot of the command above which you have modified–** **the screenshot needs to show your username and the date**

1. **QUESTION FOR YOU TO ANSWER**: Very briefly explain what you have discovered based on your data set from the query above.

**Provide a screenshot of the command above which you have modified–** **the screenshot needs to show your username and the date**

Use the **Ask the Teaching Team Discussion Forum** if you have any questions regarding the how to approach this assignment.

Save your assignment as ***lastnameFirstname\_assign5\_0.docx*** and submit it in the *Assignments* section of the course.

For help uploading files please refer to the *Technical Support* page in the syllabus.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Criterion | A | B | C | D | F | Letter Grade |
| Correctness and Completeness of Results (70%) | All steps' results are entirely complete and correct | About ¾ of the steps' results are correct and complete | About half of the steps' results are correct and complete | About ¼ of the steps' results are correct and complete | Virtually none of the step's results are correct and complete |  |
| Constitution of SQL/Python and Explanations (30%) | Excellent use and integration of appropriate SQL/Python constructs and supporting explanations | Good use and integration of appropriate SQL/Python constructs and supporting explanations | Mediocre use and integration of appropriate SQL/Python constructs and supporting explanations | Substandard use and integration of appropriate SQL/Python constructs and supporting explanations | Virtually all SQL/Python constructs and supporting explanations are unsuitable or improperly integrated |  |
|  |  |  |  |  | Assignment Grade: |  |