Course Duration (3 Months)

Started with Databricks

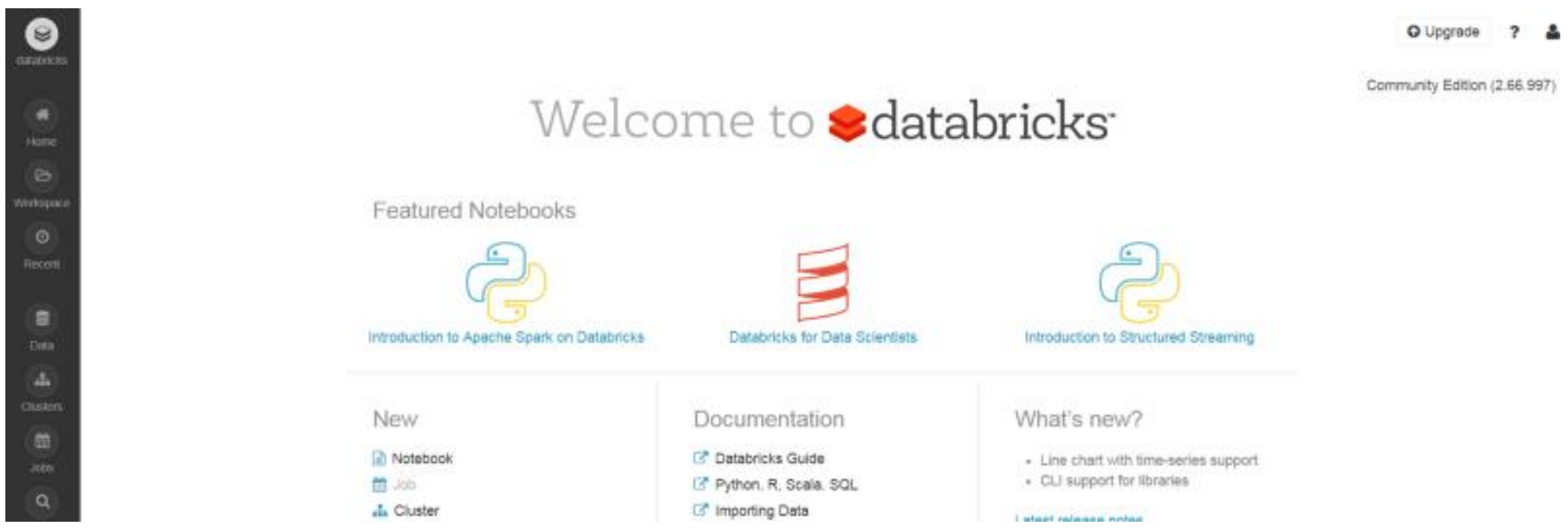
Databricks is a platform that runs on top of Apache Spark. It conveniently has a Notebook systems setup. One can easily provision clusters in the cloud, and it also incorporates an integrated workspace for exploration and visualization

You can also schedule any existing notebook or locally developed Spark code to go from prototype to production without re-engineering.

1. Setup a Databricks account

To get started with the tutorial, navigate to this link and select the free Community Edition to open your account. This option has single cluster with up to 6 GB free storage. It allows you to create a basic Notebook. You'll need a valid email address to verify your account.

You will observe this screen once you successfully log in to your account.

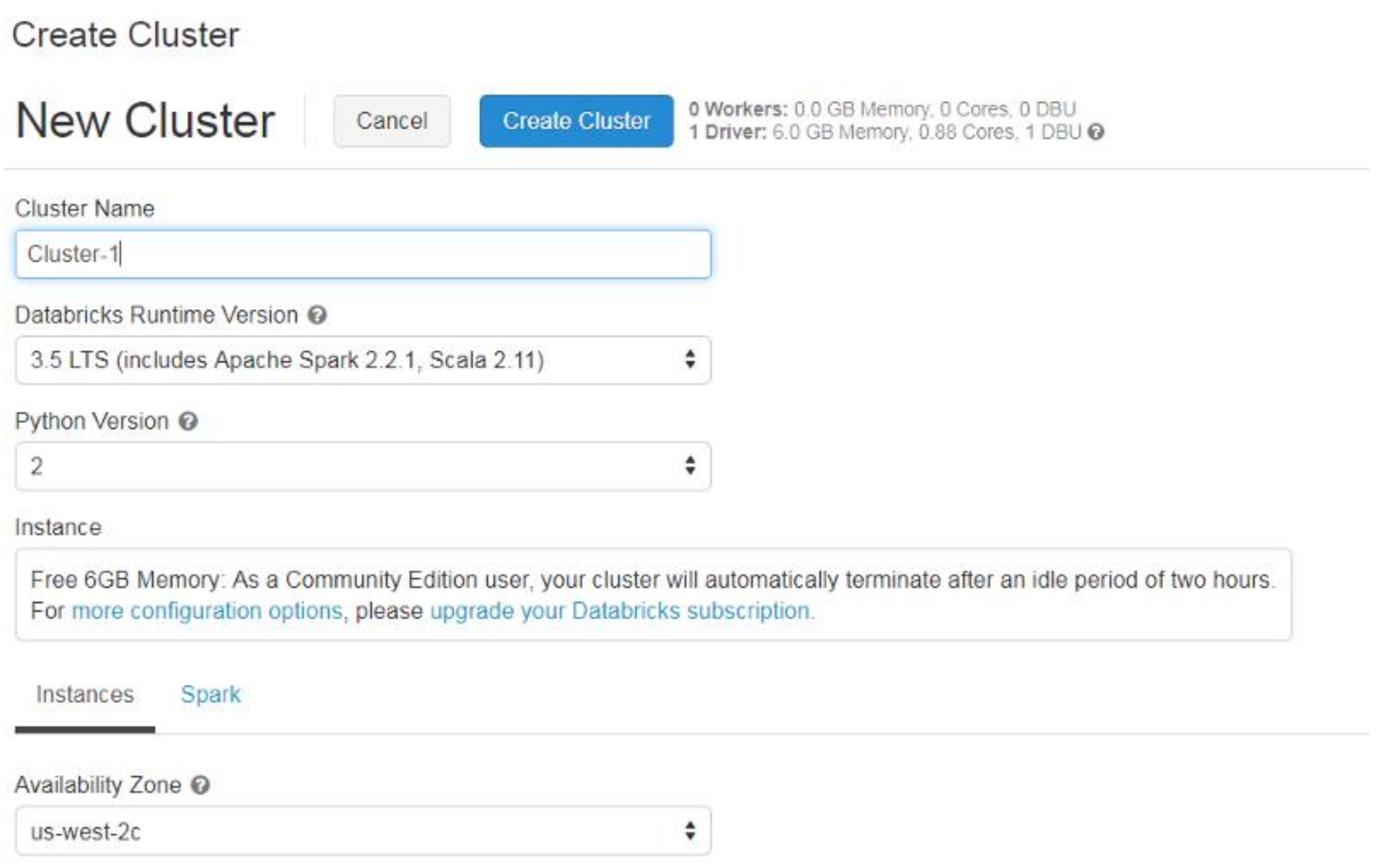


2. Creating a new Cluster

We start with creating a new cluster to run our programs on. Click on "Cluster" on the main page and type in a new name for the cluster.

Next, you need to select the "Databricks Runtime" version. Databricks Runtime is a set of core components that run on clusters managed by Databricks. It includes Apache Spark, but also adds a number of components and updates to improve the usability and performance of the tool.

You can select any Databricks Runtime version—I have selected 3.5 LTS (includes Apache Spark 2.2.1, Scala 2.11). You also have a choice between Python 2 and 3.



Create Cluster

New Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

Cluster Name: Cluster-1

Databricks Runtime Version: 3.5 LTS (includes Apache Spark 2.2.1, Scala 2.11)

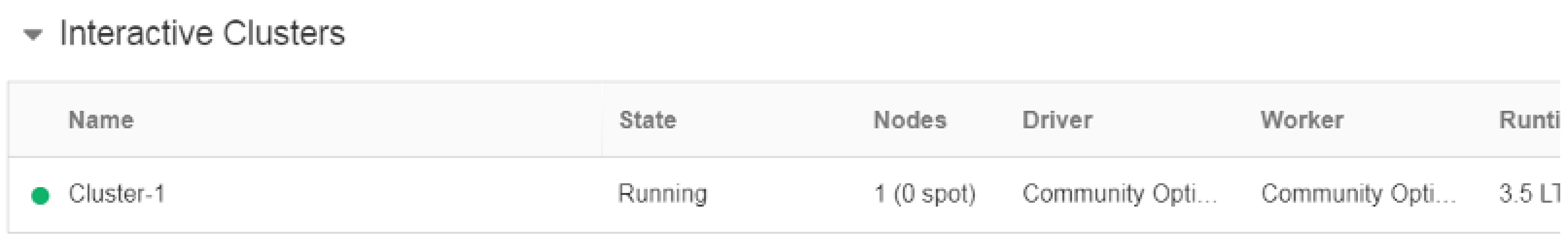
Python Version: 2

Instance

Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Availability Zone: us-west-2c

It’ll take a few minutes to create the cluster. After some time, you should be able to see an active cluster on the dashboard.



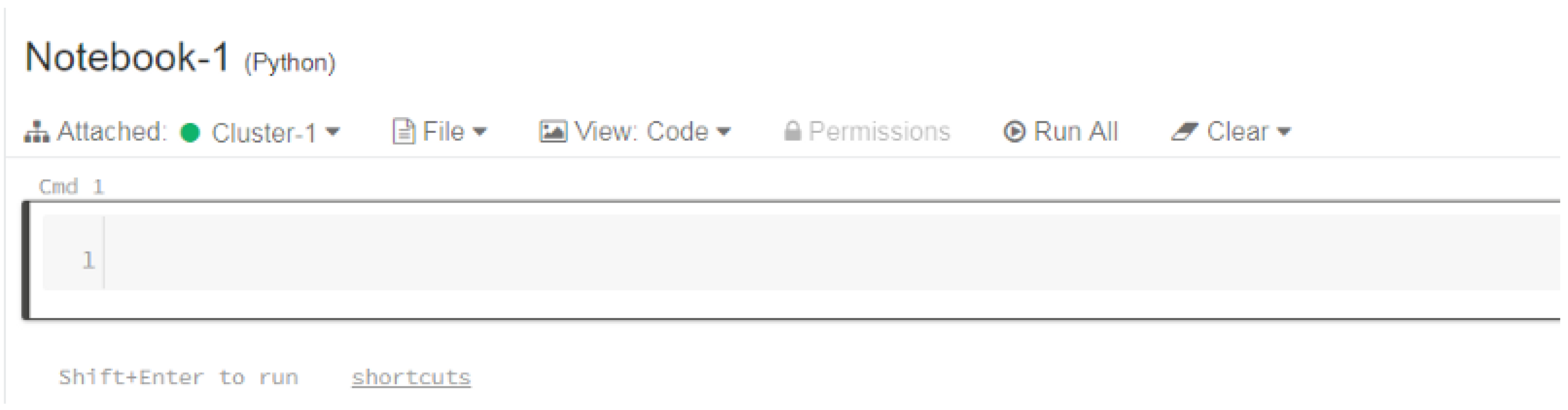
Interactive Clusters

Name	State	Nodes	Driver	Worker	Runti
● Cluster-1	Running	1 (0 spot)	Community Opti...	Community Opti...	3.5 LT

3. Creating a new Notebook

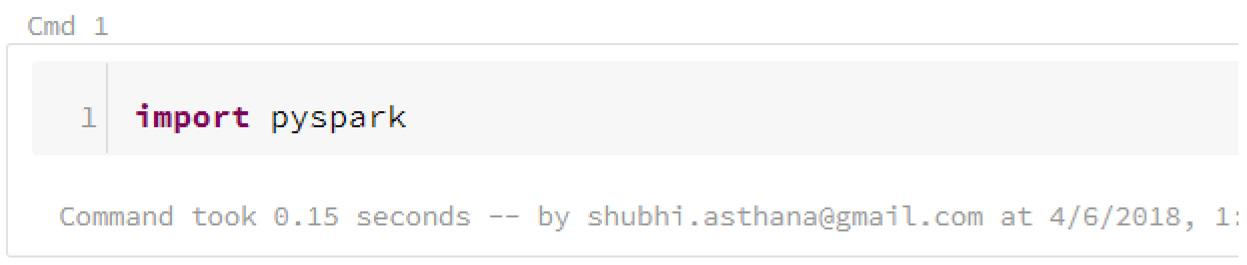
Let’s go ahead and create a new Notebook on which you can run your program.

From the main page, hit “New Notebook” and type in a name for the Notebook. Select the language of your choice—I chose Python here. You can see that Databricks supports multiple languages including Scala, R and SQL.



Notebook-1 (Python)

Once the details are entered, you will observe that the layout of the notebook is very similar to the Jupyter notebook. To test the notebook, let’s import pyspark.



Cmd 1

```
import pyspark
```

```
Command took 0.15 seconds -- by shubhi.asthana@gmail.com at 4/6/2018, 1:
```

The command ran in 0.15 seconds and also gives the cluster name on which it is running. If there are any errors in the code, it would show below the cmd box.

You can hit the keyboard icon on the top right corner of the page to see operating system-specific shortcuts.

The most important shortcuts here are:

- Shift+Enter to run a cell
- Ctrl+Enter keeps running the same cell without moving to the next cell

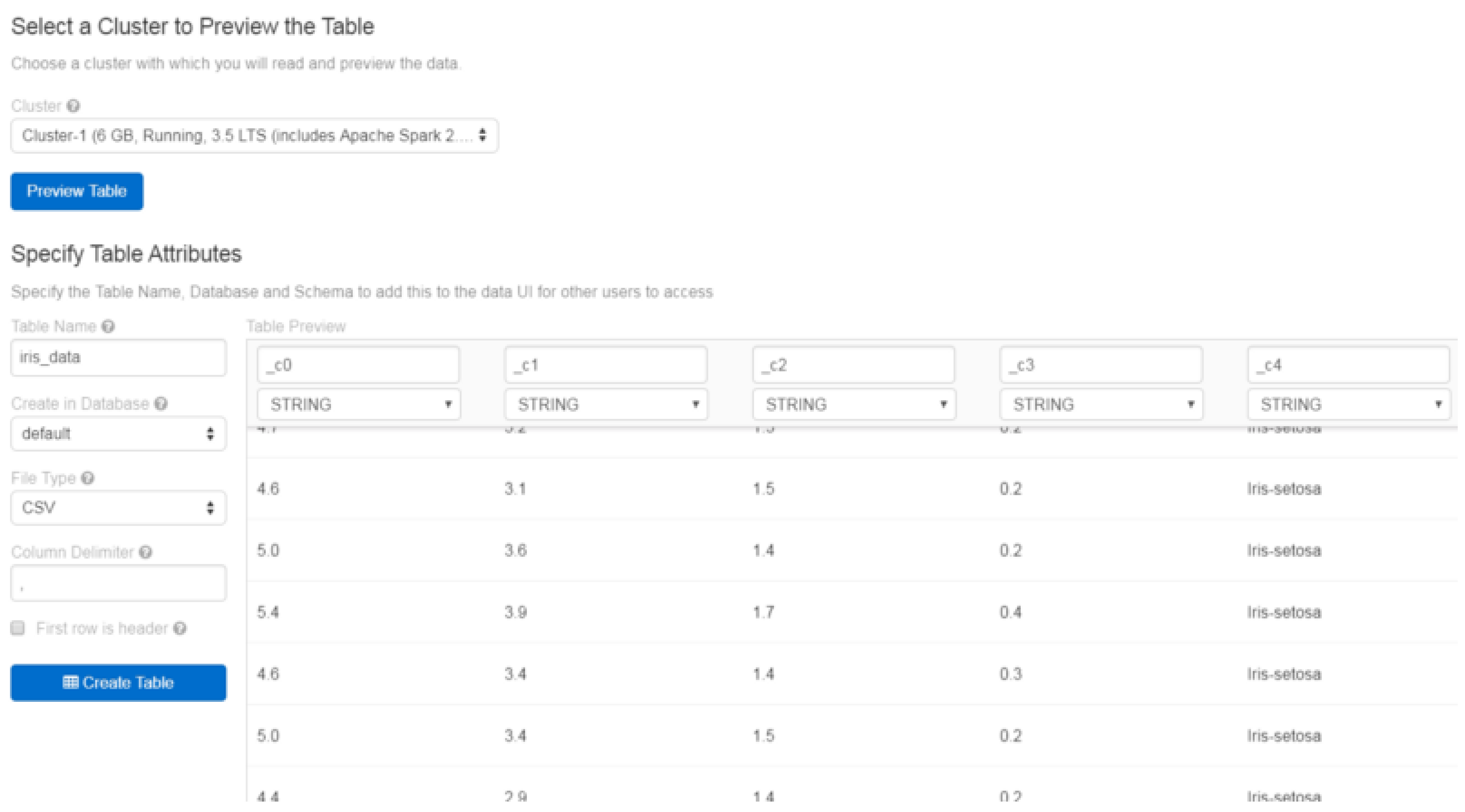
Note these shortcuts are for Windows. You can check the OS-specific shortcuts for your OS on the keyboard icon.

4. Uploading data to Databricks

Head over to the “Tables” section on the left bar, and hit “Create Table.” You can upload a file, or connect to a Spark data source or some other database.

Let’s upload the commonly used iris dataset file here (if you don’t have the dataset, use this link)

Once you upload the data, create the table with a UI so you can visualize the table, and preview it on your cluster. As you can see, you can observe the attributes of the table. Spark will try to detect the datatype of each of the columns, and lets you edit it too.



Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data

Cluster: Cluster-1 (6 GB, Running, 3.5 LTS (includes Apache Spark 2....

Preview Table

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name: iris_data

Create in Database: default

File Type: CSV

Column Delimiter: ,

First row is header

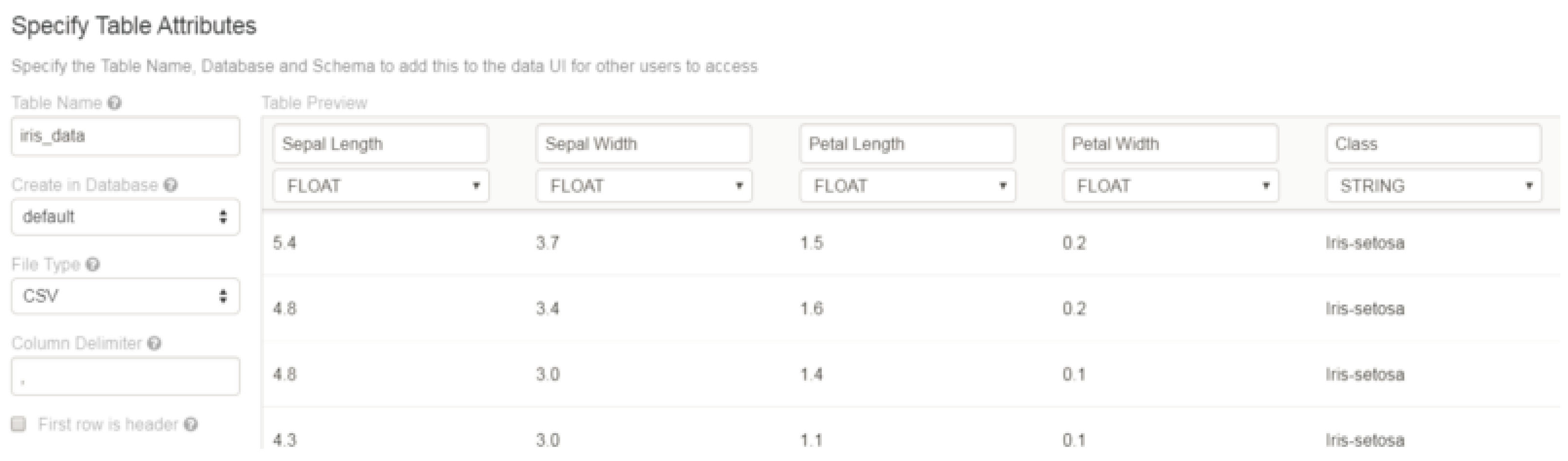
Create Table

Table Preview

_c0	_c1	_c2	_c3	_c4
STRING	STRING	STRING	STRING	STRING
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa

Now I need to put headers for the columns, so I can identify each column by their header instead of `_c0`, `_c1` and so on.

I put their headers as Sepal Length, Sepal Width, Petal Length, Petal Width and Class. Here, Spark detected the datatype of the first four columns incorrectly as a String, so I changed it to the desired datatype—Float.


Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Name	iris_data
Create in Database	default
File Type	CSV
Column Delimiter	,
First row is header	☐

Table Preview

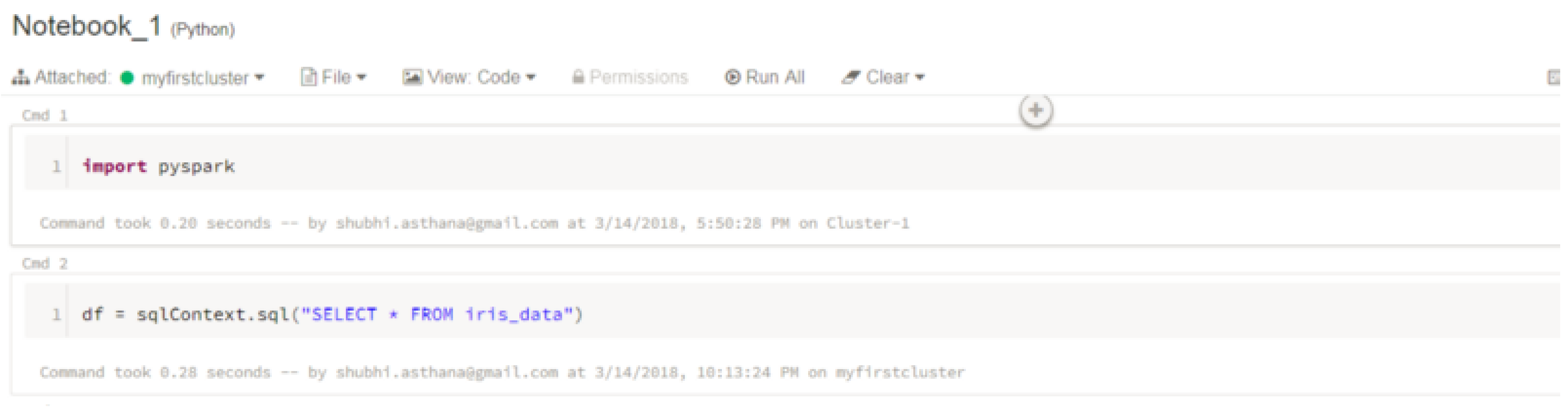
Sepal Length	Sepal Width	Petal Length	Petal Width	Class
FLOAT	FLOAT	FLOAT	FLOAT	STRING
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa

5. How to access data from Notebook

Spark is a framework that can be used to analyze big data using SQL, machine learning, graph processing or real-time streaming analysis. We will be working with SparkSQL and Dataframes in this tutorial.

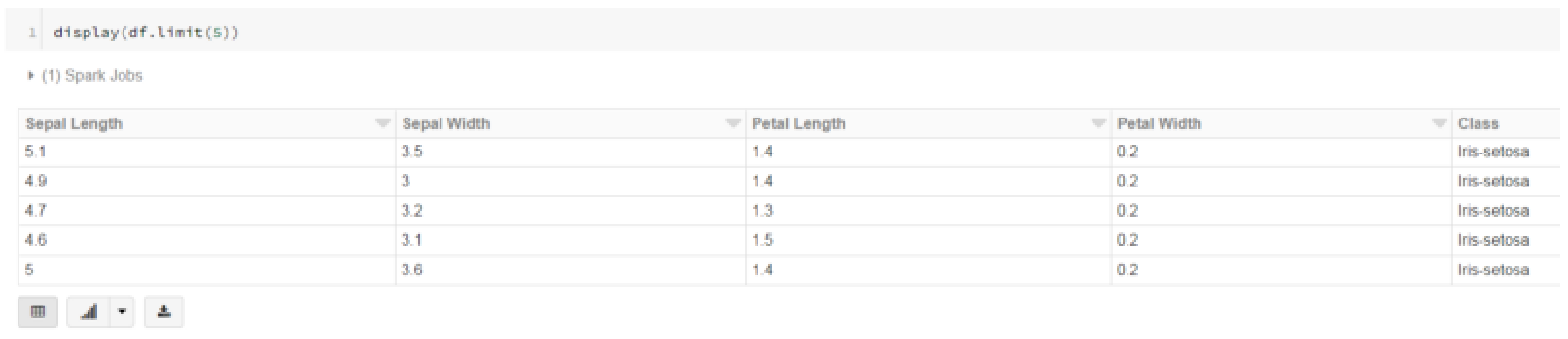
Let's get started with working with the data on the Notebook. The data that we have uploaded is now put in tabular format.We require a SQL query to read the data and put it in a dataframe.

Type `df = sqlContext.sql("SELECT * FROM iris_data")` to read iris data into a dataframe.



```
Cmd 1
import pyspark
Command took 0.20 seconds -- by shubhi.asthana@gmail.com at 3/14/2018, 5:50:28 PM on Cluster-1
Cmd 2
df = sqlContext.sql("SELECT * FROM iris_data")
Command took 0.28 seconds -- by shubhi.asthana@gmail.com at 3/14/2018, 10:13:24 PM on myfirstcluster
```

To view the first five rows in the dataframe, I can simply run the command:
`display(df.limit(5))`

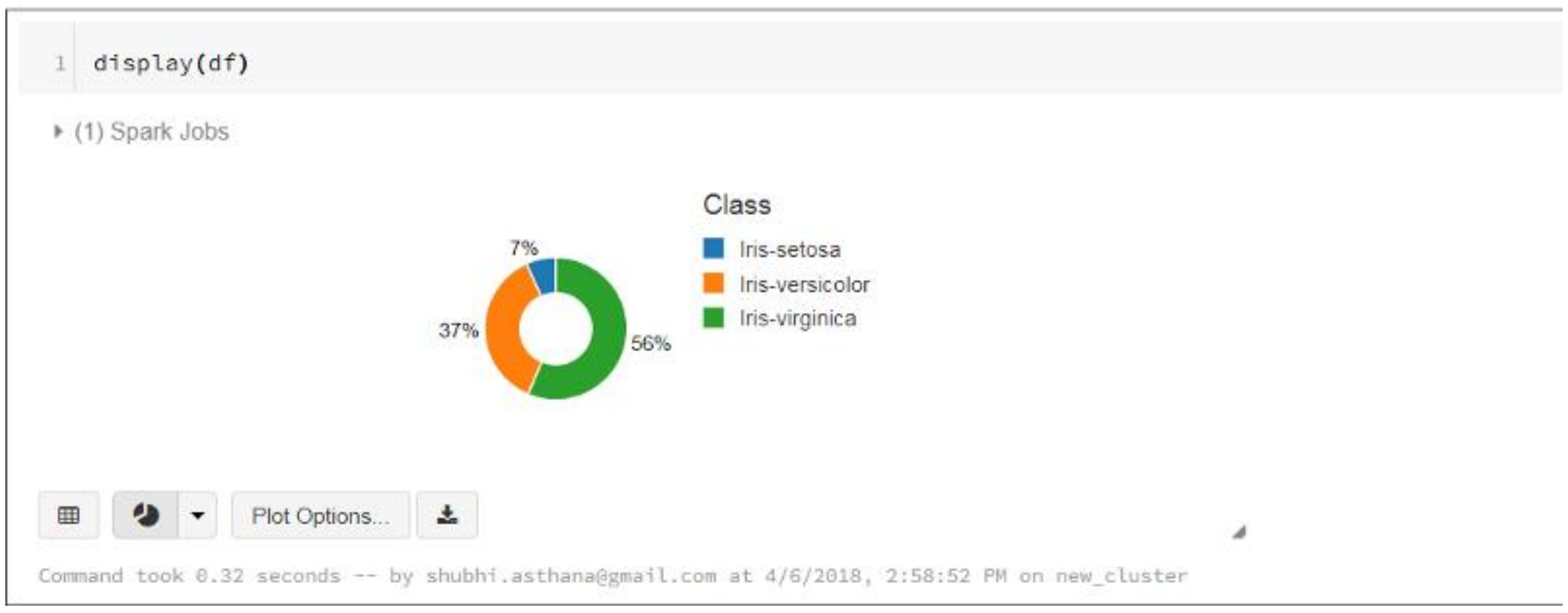


```
display(df.limit(5))
```

(1) Spark Jobs

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa

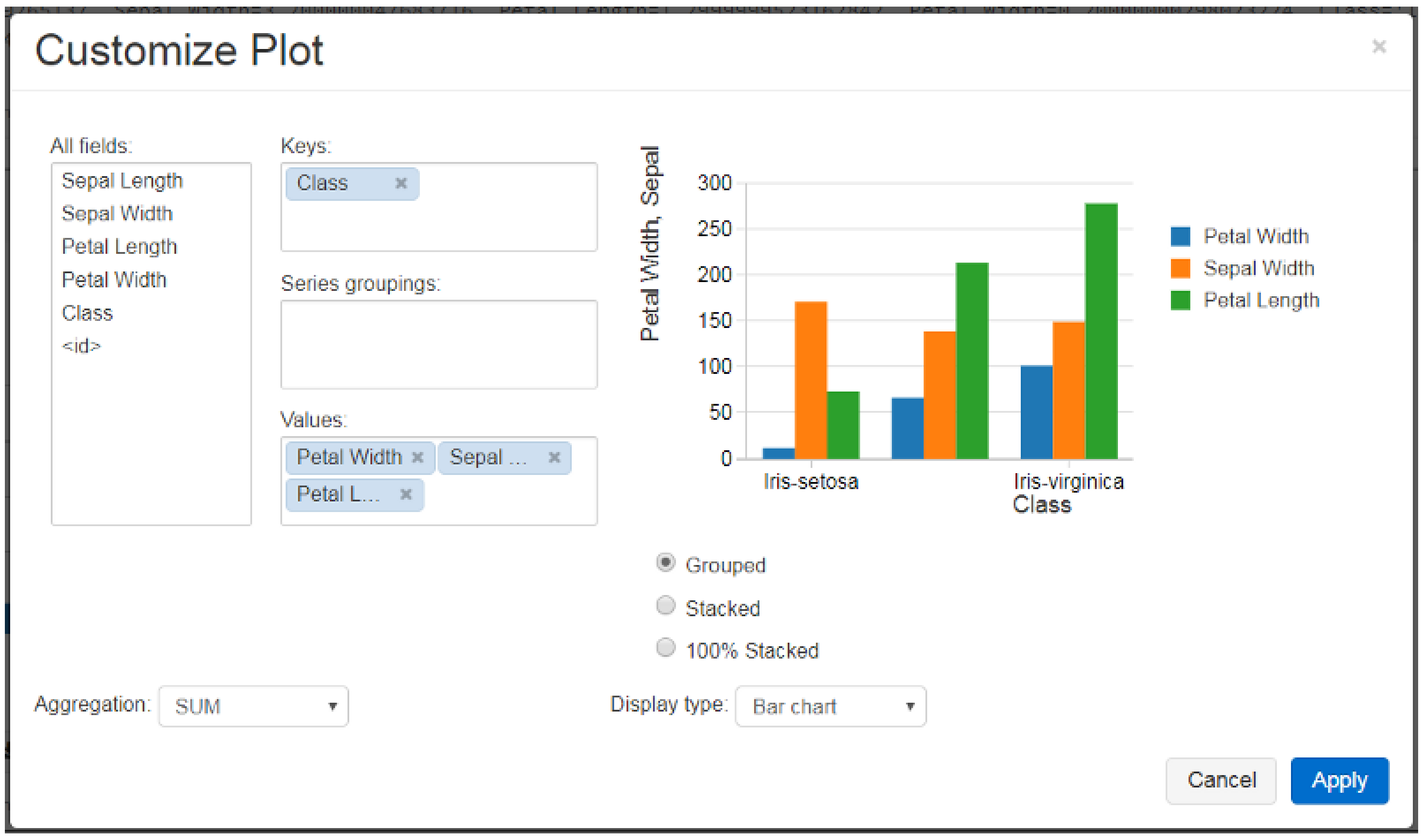
Notice a Bar chart icon at the bottom. Once you click, you can view the data that you have imported into Databricks. To view the bar chart of complete data, run`display(df)` instead of `display(df.limit(5))`.



```
display(df)
```

```
Command took 0.32 seconds -- by shubhi.asthana@gmail.com at 4/6/2018, 2:58:52 PM on new_cluster
```

The dropdown button allows you to visualize the data in different charts like bar, pie, scatter, and so on. It also gives you plot options to customize the plot and visualize specific columns only.



Customize Plot

All fields: Sepal Length, Sepal Width, Petal Length, Petal Width, Class, <id>

Keys: Class

Series groupings:

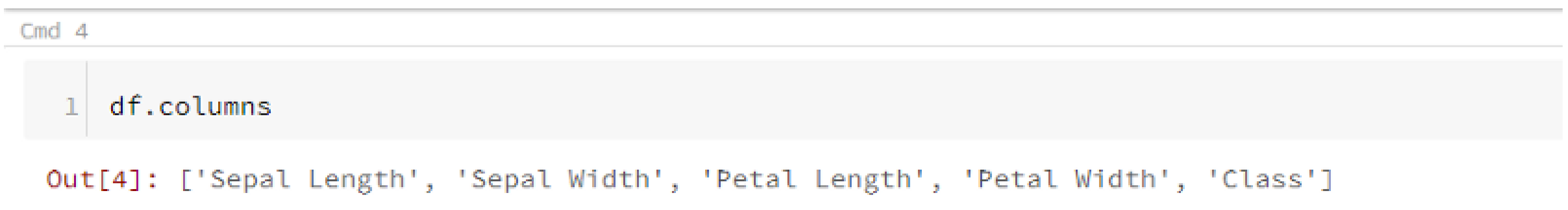
Values: Petal Width, Sepal ..., Petal L...

[✓ Grouped] Stacked 100% Stacked

Aggregation: SUM

Display type: Bar chart

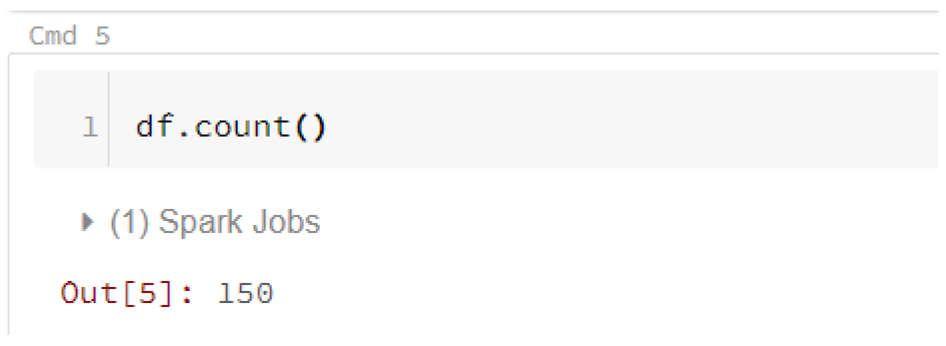
You can also display matplotlib and ggplot figures in Databricks. For a demonstration, see [Matplotlib and ggplot in Python Notebooks](#).
To view all the columns of the data, simply type `df.columns`



```
Cmd 4

df.columns

Out[4]: ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width', 'Class']
```

```
Cmd 4

df.columns

Out[4]: ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width', 'Class']
```

To count how many rows total there are in the Dataframe (and see how long it takes to a full scan from remote disk/S3), run `df.count()`.


Cmd 5

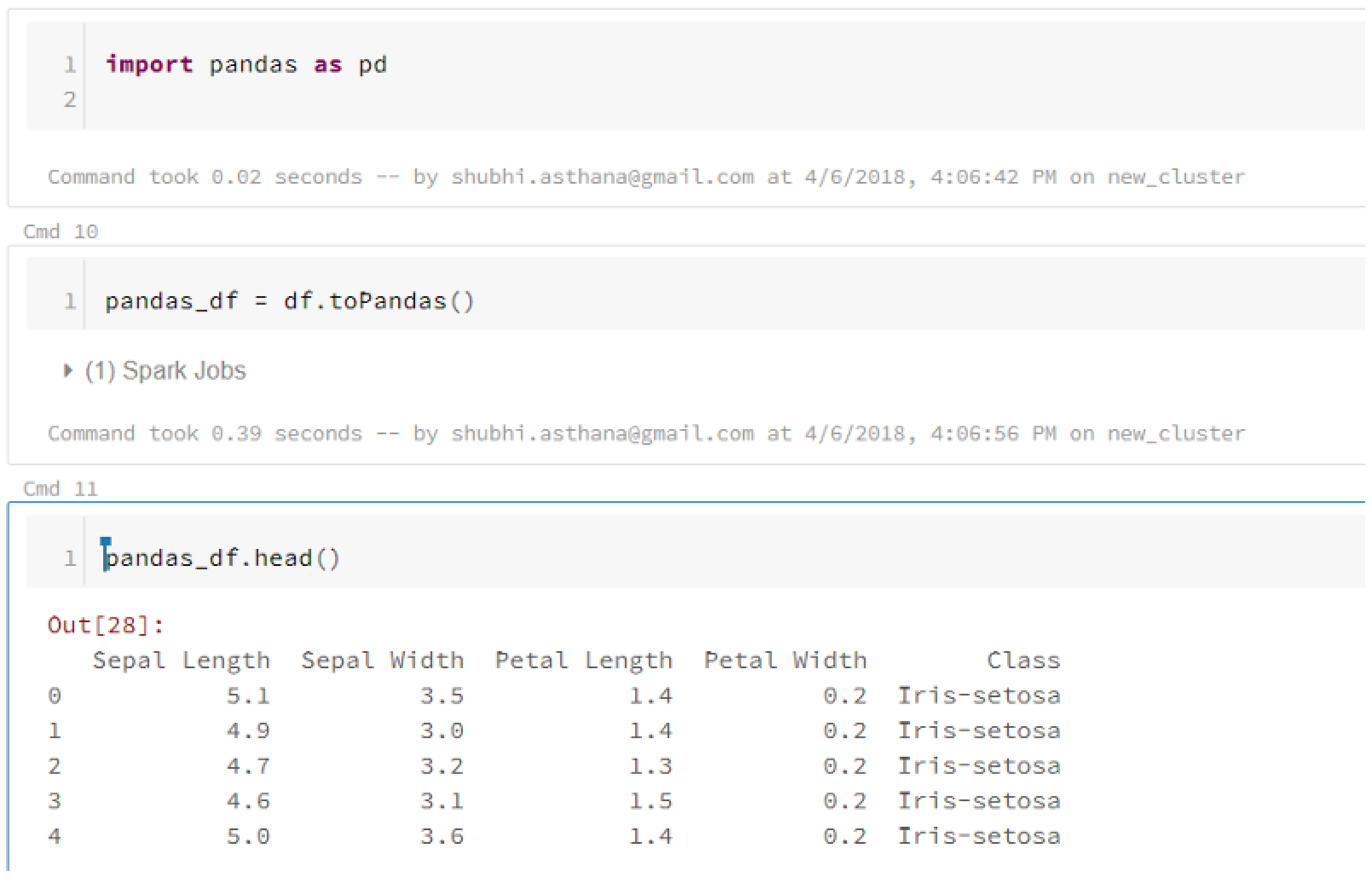
```
df.count()
```

▸ (1) Spark Jobs

```
Out[5]: 150
```

6. Converting a Spark dataframe to a Pandas dataframe.

Now if you are comfortable using pandas dataframes, and want to convert your Spark dataframe to pandas, you can do this by putting the command
\
Now you can use pandas operations on the `pandas_df` dataframe.



```
import pandas as pd

```

Command took 0.02 seconds -- by shubhi.asthana@gmail.com at 4/6/2018, 4:06:42 PM on new_cluster

Cmd 10

```
pandas_df = df.toPandas()
```

▸ (1) Spark Jobs

Command took 0.39 seconds -- by shubhi.asthana@gmail.com at 4/6/2018, 4:06:56 PM on new_cluster

Cmd 11

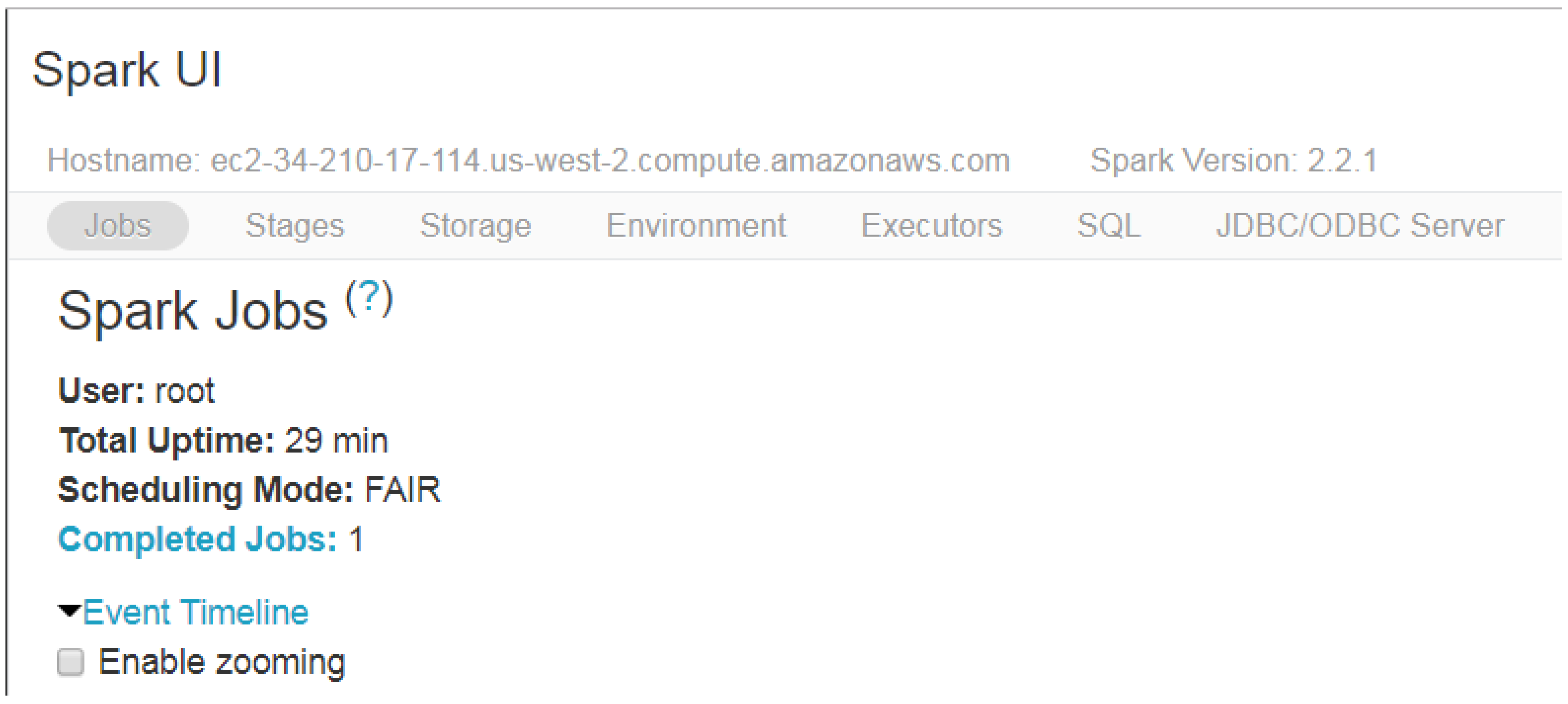
```
pandas_df.head()
```

```
Out[28]:
   Sepal Length  Sepal Width  Petal Length  Petal Width        Class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

7. Viewing the Spark UI

The Spark UI contains a wealth of information needed for debugging Spark jobs. There are a bunch of great visualizations, so let’s view them in a gist.

To go to Spark UI, you need to go to the top of the page where there are some menu options like “File,” “View,” “Code,” “Permissions,” and others. You will find the name of the cluster at the top next to “Attached” and a dropdown button next to it. Hit the dropdown button and select “View Spark UI.” A new tab will open up with the lots of information on your Notebook.


Spark UI

Hostname: ec2-34-210-17-114.us-west-2.compute.amazonaws.com Spark Version: 2.2.1

Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server

Spark Jobs (?)

User: root
Total Uptime: 29 min
Scheduling Mode: FAIR
Completed Jobs: 1

Event Timeline
Enable zooming

The UI view gives plenty of information on each job executed on the cluster, stages, environment, and SQL queries executed. This UI can be helpful for users to debug their applications. Also, this UI gives a good visualization on Spark streaming statistics. To learn in more detail about each aspect of the Spark UI, refer to this link.

Once you are done with the Notebook, you can go ahead and publish it or export the file in different file formats, such that somebody else can use it using a unique link. I have attached my Notebook in HTML format.

Wrapping up

This is a short overview on how you can get started with Databricks quickly and run your programs. The advantage of using Databricks is that it offers an end-to-end service for building analytics, data warehousing, and machine learning applications. The entire Spark cluster can be managed, monitored, and secured using a self-service model of Databricks.

Here are some interesting links for Data Scientists and for Data Engineers. Also, here is a tutorial which I found very useful and is great for beginners.

1. Just Enough Scala for Apache Spark™

Overview

This 1-day course provides an overview of the core features of Scala that you need to know to use Apache Spark effectively. You'll learn the most important Scala syntax, idioms, and APIs for Spark development.

Each topic includes lecture content along with hands-on use of Scala through an elegant web-based notebook environment. Students may keep the notebooks and continue to use them with the free Databricks Community Edition offering; all examples are guaranteed to run in that environment. Alternatively, each notebook can be exported as source code and run within any Spark environment.

Learning Objectives

- Understand the basics of Scala programming, without delving into the more advanced areas of Scala that aren't necessary for Spark.
- Compare and contrast Scala with languages like Python and Java.
- Learn how to write classes, functions, and full programs in Scala.
- Understand the basics of Scala's compile time typing.
- Learn how Scala's compact, powerful syntax can help you write better Spark jobs with less code.

Topics

- Brief comparison of Scala and Java
- Brief overview of the Scala language
- How to compile a Scala program
- The Scala shell (interpreter)
- Brief overview of tooling
    - developing Scala in an IDE
    - SBT, the Scala Build Tool
- Basic Scala syntax

- Scala variables, including mutable vs. immutable values
- Basic Scala types (primitives, tuples)
- Control flow (loops, conditionals)
- Functions, and lambdas
- Scoping
- Object-oriented programming in Scala
    - classes, traits and inheritance
    - methods
- Scala collections and the common operations on them
- Type inference
- Imports
- Overview of functional vs. imperative programming
- Case classes
- Pattern matching
- For-comprehensions

1. Apache Spark™ Overview

This 1-day course is for data engineers, analysts, architects, data scientist, software engineers, IT operations, and technical managers interested in a brief hands-on overview of Apache Spark.

The course provides an introduction to the Spark architecture, some of the core APIs for using Spark, SQL and other high-level data access tools, as well as Spark's streaming capabilities and machine learning APIs. The class is a mixture of lecture and hands-on labs. Each topic includes lecture content along with hands-on labs in the Databricks notebook environment. Students may keep the notebooks and continue to use them with the free Databricks Community Edition offering after the class ends; all examples are guaranteed to run in that environment.

Objectives

After taking this class, students will be able to:

- Use a subset of the core Spark APIs to operate on data.
- Articulate and implement simple use cases for Spark
- Build data pipelines and query large data sets using Spark SQL and Data Frames
- Create Structured Streaming jobs
- Understand how a Machine Learning pipeline works
- Understand the basics of Spark's internals

Topics

- Spark Overview
- Introduction to Spark SQL and Data Frames, including:
    - Reading & Writing Data
    - The Data Frames/Datasets API
    - Spark SQL
    - Caching and caching storage levels
- Overview of Spark internals
    - Cluster Architecture
    - How Spark schedules and executes jobs and tasks
    - Shuffling, shuffle files, and performance
    - The Catalyst query optimizer
- Spark Structured Streaming
    - Sources and sinks
    - Structured Streaming APIs
    - Windowing & Aggregation
    - Check pointing & Watermarking
    - Reliability and Fault Tolerance
- Overview of Spark's MLlib Pipeline API for Machine Learning
    - Transformer/Estimator/Pipeline API
    - Perform feature pre-processing
    - Evaluate and apply ML models

Course Syllabus

MODULE	LECTURE	HANDS-ON
Apache Spark Overview	Overview of Databricks Spark Capabilities Spark Ecosystem Basic Spark Components	Databricks Lab Environment Working with Notebooks Spark Clusters and Files
Apache Spark SQL and DataFrames	Use of Spark SQL Use of DataFrames / DataSets Reading & Writing Data DataFrame, DataSet and SQL APIs Catalyst Query Optimization ETL	Creating DataFrames Querying with DataFrames Querying with SQL ETL with DataFrames Visualization
Structured Streaming	Streaming Sources and Sinks Structured Streaming APIs Windowing & Aggregation Checkpointing Watermarking Reliability and Fault Tolerance	Reading from TCP Continuous Visualization
Machine Learning	Spark MLlib Pipeline API Built-in Featuring and Algorithms	Featurization Building a Machine Learning Pipeline

2. Apache Spark™ Programming

Overview

This 3-day course is equally applicable to data engineers, data scientist, analysts, architects, software engineers, and technical managers interested in a thorough, hands-on overview of Apache Spark.

The course covers the fundamentals of Apache Spark including Spark's architecture and internals, the core APIs for using Spark, SQL and other high-level data access tools, as well as Spark's streaming capabilities and machine learning APIs. The class is a mixture of lecture and hands-on labs.

Each topic includes lecture content along with hands-on labs in the Databricks notebook environment. Students may keep the notebooks and continue to use them with the free Databricks Community Edition offering after the class ends; all examples are guaranteed to run in that environment.

Learning Objectives

After taking this class, students will be able to:

- Use the core Spark APIs to operate on data
- Articulate and implement typical use cases for Spark
- Build data pipelines and query large data sets using Spark SQL and Data Frames
- Analyse Spark jobs using the administration UIs inside Databricks
- Create Structured Streaming jobs
- Work with relational data using the Graph Frames APIs
- Understand how a Machine Learning pipeline works
- Understand the basics of Spark's internals

Topics

- Spark Overview
- In-depth discussion of Spark SQL and Data Frames, including:
  - The Data Frames/Datasets API
  - Spark SQL
  - Data Aggregation
  - Column Operations
  - The Functions API: date/time, string manipulation, aggregation

    - Joins & Broadcasting
    - User Defined Functions
    - Caching and caching storage levels
    - Use of the Spark UI to analyse behaviour and performance
- In-depth discussion of Spark internals
    - Cluster Architecture
    - The Catalyst query optimizer
    - The Tungsten in-memory data format
    - How Spark schedules and executes jobs and tasks
    - Shuffling, shuffle files, and performance
    - How various data sources are partitioned
    - How Spark handles data reads and writes
- Spark Structured Streaming
    - Sources and sinks
    - Structured Streaming APIs
    - Windowing & Aggregation
    - Check pointing & Watermarking
    - Reliability and Fault Tolerance
    - Kafka Integration
- Overview of Spark's MLlib Pipeline API for Machine Learning
    - Transformer/Estimator/Pipeline API
    - Perform feature pre-processing
    - Evaluate and apply ML models
- Graph processing with Graph Frames
    - Transforming Data Frames into a graph
    - Perform graph analysis, including Label Propagation, PageRank, and Shortest Paths

3. Apache Spark™ Tuning and Best Practices

Overview

This 3-day course is primarily for software engineers but is directly applicable to analysts, architects and data scientist interested in a deep dive into the processes of tuning Spark applications, developing best practices and avoiding many of the common pitfalls associated with developing Spark applications.

This course is a lab-intensive workshop in which students implement various best practices while inducing, diagnose and then fixing various performance problems. The course continues with numerous instructor-led coding challenges to refactor existing code with the effect of an increase in overall performance by applying learned best practices. It then concludes with a full day workshop in which students work individually or in teams to complete a typical, full-scale data migration of a poorly maintained dataset.

Each topic includes lecture content along with hands-on labs in the Databricks notebook environment. Students may keep the notebooks and continue to use them with the free Databricks Community Edition offering after the class ends; all examples are guaranteed to run in that environment.

Learning Objectives

After taking this class, students will be able to:

- Shortcut investigations by developing common-sense intuition as to the root cause of various performance issues.
- Diagnose & fix various storage-related performance issues including
    - The tiny files problem
    - Malformed partitions
    - Un partitioned data
    - Over partitioned data
    - Incorrectly typed data
- Identify when and when not to cache data
- Articulate the performance ramifications of different caching strategies
- Diagnose & fix common coding mistakes that lead to de-optimization
- Optimize joins via broadcasting, pruning, and pre-joining
- Apply tips and tricks for
    - Investigating the files system
    - Diagnosing partition skew
    - Developing and distributing utility functions
    - Developing micro-benchmarks & avoiding related pitfalls
    - Rapid ETL development for extremely large datasets
    - Working in shared cluster environments
- Develop different strategies for testing ETL components
- Rapidly develop insights on otherwise costly datasets

Topics

- Coding Exercises

    - This course serves as an excellent follow-up to Databricks' other courses:
        - Apache Spark Programming (DB 105)
        - Apache Spark for Machine Learning and Data Science (DB 301)
        - Students will implement more than 75% of all exercises which in turn induce the various performance problems to be diagnosed and fixed
- Partitioning
    - Explore the effects of different partitioning strategies
    - Diagnose performance problems related to improperly partitioned data
    - Explore different solutions to fixing mal-partitioned data
    - Working with and understanding on-disk partitioning strategies
- Caching
    - Develop tips and tricks for caching data on shared clusters
    - Explore the ramifications of different caching strategies
    - Learn why caching is one of the most common performance problems
    - Develop intuitions as to when and when not to cache data
    - How to use caching as an aid to troubleshooting
- Joins
    - Explore different options for optimizing joins
    - Working with broadcast joins
    - Explore different options for avoiding joins
- Utility Functions
    - Diagnosing performance problems
    - Caching data
    - Benchmarking
    - Common ETL tasks
    - Discuss deployment strategies for utility functions
- Testing strategies
    - Strategies for testing transformations
    - Developing test datasets for unit tests
- De-optimization
    - Exploring common coding practices that induce de-optimization
    - Solutions for avoiding de-optimization
    - Review of the Catalyst Optimizer and its role in optimizing applications

4. Apache Spark™ for Machine Learning and Data Science

Overview

This 3-day course is primarily for data scientists but is directly applicable to analysts, architects, software engineers, and technical managers interested in a thorough, hands-on overview of Apache Spark and its applications to Machine Learning.

The course covers the fundamentals of Apache Spark including Spark's architecture and internals, the core APIs for using Spark, SQL and other high-level data access tools, Spark's streaming capabilities and a heavy focus on Spark's machine learning APIs. The class is a mixture of lecture and hands-on labs.

Each topic includes lecture content along with hands-on labs in the Databricks notebook environment. Students may keep the notebooks and continue to use them after the class ends; all examples are guaranteed to run in the environment the class was taught on (Azure Databricks or CE)

Learning Objectives

After taking this class, students will be able to:

- Understand when and where to use Spark
- Articulate the difference between an RDD, Data Frame, and Dataset
- Explain supervised vs unsupervised machine learning, and typical applications of both
- Build a Machine Learning Pipeline using a combination of Transformers and Estimators
    - Save/Restore Models
    - Apply models to streaming data
- Perform hyper parameter tuning with cross-validation
- Analyse Spark query performance using the Spark UI
- Train models with 3rd party libraries such as XGBoost
- Perform hyper parameter search in parallel using single node algorithms such as scikit-learn
- Gain familiarity with Decision Trees, Random Forests, Gradient Boosted Trees, Linear Regression, Collaborative Filtering, and K-Means
- Explain options for putting models into production

Topics

- Spark Overview
- In-depth discussion of Spark SQL and Data Frames, including:
    - RDD vs Data Frame vs Dataset API
    - Spark SQL
    - Data Aggregation
    - Column Operations
    - The Functions API: date/time, string manipulation, aggregation
    - Caching and caching storage levels
    - Use of the Spark UI to analyse behaviour and performance
- Overview of Spark internals
    - Cluster Architecture
    - How Spark schedules and executes jobs and tasks
    - Shuffling, shuffle files, and performance
    - The Catalyst query optimizer
- Spark Structured Streaming
    - Sources and sinks
    - Structured Streaming APIs
    - Windowing & Aggregation
    - Check pointing & Watermarking
    - Reliability and Fault Tolerance
- In-depth overview of Spark's MLlib Pipeline API for Machine Learning
    - Build machine learning pipelines for both supervised and unsupervised learning
    - Transformer/Estimator/Pipeline API
    - Use transformers to perform pre-processing on a dataset prior to training
    - Train analytical models with Spark ML's DataFrame-based estimators including Decision Trees, Random Forests, Gradient Boosted Trees, Linear Regression, K-Means, and Alternating Least Squares
    - Tune hyperparameters via cross-validation and grid search
    - Evaluate model performance
- MLflow
    - Track and benchmark model performance
- 3rd Party Library Integrations
    - XGBoost
    - How to distribute single-node algorithms (like scikit-learn) with Spark
        - Spark-Sklearn: Perform scikit-learn hyperparameter search in parallel
- Production Discussion

5. Databricks Delta

OVERVIEW

This 1-day course is for data engineers, architects, data scientists and software engineers who want to use Databricks Delta for ETL processing on Data Lakes. The course ends with a capstone project building a complete data pipeline using Databricks Delta.

Each topic includes lecture content along with hands-on labs in the Databricks notebook environment. Students may keep the notebooks and continue to use them with the free Databricks Community Edition offering after the class ends; all examples are guaranteed to run in that environment

OBJECTIVES

After taking this class, students will be able to:

- Use the interactive Databricks notebook environment.
- Use Databricks Delta to create, append and upsert data into a Data Lake.
- Use Databricks Delta to manage and extract actionable insights out of a Data Lake.
- Use Databricks Delta's advanced optimization features to speed up queries.
- Use Databricks Delta to seamlessly ingest streaming and historical data.
- Implement a Databricks Delta data pipeline architecture.

PLATFORMS

Supported platforms include Azure Databricks, Databricks Community Edition, and non-Azure Databricks.

- If you're planning to use the course on Azure Databricks, select the "Azure Databricks" **Platform** option.
- If you're planning to use the course on Databricks Community Edition or on a non-Azure version of Databricks, select the "Other Databricks" **Platform** option.

TOPICS

- Create
  - Work with a traditional data pipeline using online shopping data
  - Identify problems with the traditional data pipeline
  - Use Databricks Delta features to mitigate those problems
- Append
  - Append new records to a Databricks Delta table
- Upsert
  - Use Databricks Delta to UPSERT data into existing Databricks Delta tables
- Streaming
  - Read and write streaming data into a data lake
- Optimization
  - Optimize a Databricks Delta data pipeline backed by online shopping data
  - Learn about best practices to apply to data pipelines
- Architecture
  - Get streaming Wikipedia data into a data lake via Kafka broker
  - Write streaming data into a **raw** table
  - Clean up bronze data and generate normalized **query** tables
  - Create **summary** tables of key business metrics
  - Create plots/dashboards of business metrics

MODULE	LECTURE	HANDS-ON
Create	Identify problems with the traditional data pipeline Use Databricks Delta features to mitigate those problems Investigate Delta transaction logs	Create records and insert into a Databricks Delta data pipeline using online shopping data
Append	Append new records to a Databricks Delta table	Append records to a Databricks Delta data pipeline using online shopping data Work with Internet-of-Things data
Upsert	Use Databricks Delta to UPSERT data into existing Databricks Delta tables	UPSERT data into existing Databricks Delta tables
Streaming	Learn how to use Databricks Delta with Structured Streaming to ingest batch and streaming data into the same locations	Work with Internet-of-Things data Visualize output in LIVE plots
Optimization	Use Databricks Delta advanced optimization features to speed up queries Learn about best practices to apply to data pipelines	Optimize a Databricks Delta data pipeline backed by online shopping data
Architecture	Learn about Databricks Delta Architecture Discuss trade offs with lambda architecture	Learn about Databricks Delta Architecture Discuss trade-offs with lambda architecture Get streaming Wikipedia data into a data lake via Kafka broker Write streaming data into a **raw** table Clean up bronze data and generate normalized **query** tables Create **summary** tables of key business metrics Create plots/dashboards of business metrics

6. Use cases for Apache Spark

Big Data Spark—Real World Project—NYC Texi DataSet

Download Data : https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

You will learn the following typical data preparation concepts:

- Exporting and understanding the data--including the fields, schema, size--by visualizing the data in both table and graph form.
- Identifying outliers and techniques to mitigate their impact
- Finding and removing null values from datasets
- Converting and formatting the data.

You will learn about the following capabilities of Spark to achieve the above:

1. Dataframes: a distributed collection of data organized into named columns. They are derived from Resilient Distributed Datasets (RDDs), the core Spark data structure, and are conceptually equivalent to a table in a relational database.

2. Spark functions including:
   - Filter
   - Display
   - Show
   - Cast
   - Count
   - with Column.

The data we will be using is the publicly available NYC Taxi Trip Record.

1. Week 1 and Week 2 Task

you will be performing the following major tasks:

1. Exploring and visualizing the data
   - Using print Schema (), show (), count () and display () functions
   - Using Spark's built-in charting capabilities
2. Converting data types
   - Using cast () function
3. Detecting and handling outliers
4. Null detection and handling
5. Formatting Data
6. Using standard date functions

Introducing Data Frames

Data Frames are a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database, with richer optimizations under the hood and the benefit of being distributed across a cluster. They are derived from data structures known as Resilient Distributed Datasets (RDDs). RDD's are:

- **Resilient**: They are fault tolerant, so if part of your operation fails, Spark quickly recovers the lost computation.
- **Distributed**: RDDs are distributed across networked machines known as a cluster.
- **Datasets**: A Dataset is a strongly typed collection of domain-specific objects that can be transformed in parallel using functional or relational operations.

An RDD in Spark is simply an immutable distributed collection of objects. All work is expressed as either creating new RDDs, transforming existing RDDs, or calling operations on RDDs to compute a result.

DATASET: New York City Taxi Trips

This dataset contains detailed trip-level data on New York City Taxi trips. It was collected from both drivers inputs as well as the GPS coordinates of individual cabs.

The data is in a CSV format and has the following fields:

- tripID: a unique identifier for each trip
- VendorID: a code indicating the provider associated with the trip record
- tpep_pickup_datetime: date and time when the meter was engaged
- tpep_dropoff_datetime: date and time when the meter was disengaged
- passenger_count: the number of passengers in the vehicle (driver entered value)
- trip_distance: The elapsed trip distance in miles reported by the taximeter
- RatecodeID: The final rate code in effect at the end of the trip -1= Standard rate -2=JFK -3=Newark -4=Nassau or Westchester -5=Negotiated fare -6=Group ride
- store_and_fwd_flag: This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
- PULocationID: TLC Taxi Zone in which the taximeter was engaged
- DOLocationID: TLC Taxi Zone in which the taximeter was disengaged
- payment_type: A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
- fare_amount: The time-and-distance fare calculated by the meter.
- extra: Miscellaneous extras and surcharges
- mta_tax: $0.50 MTA tax that is automatically triggered based on the metered rate in use
- tip_amount: Tip amount –This field is automatically populated for credit card tips. Cash tips are not included
- tolls_amount:Total amount of all tolls paid in trip
- improvement_surcharge: $0.30 improvement surcharge assessed trips at the flag drop.
- total_amount: The total amount charged to passengers. Does not include cash tips.

2. Week 3 and Week 4 Task

You will be performing the following tasks

1. Convert Data Frames to tables using the saveAsTable ()
2. Basic Aggregation
   - Using the `abs` and `round` functions to perform calculations and transformations
3. Explore and implement various SQL *joins* to combine tables and Data Frames.
   - **Shuffle join** is the default join method in Spark SQL. Like every shuffle operation, it consists of moving data between executors.
   - **Broadcast join** uses broadcast variables to send Data Frames to join with other Data Frames as a broadcast variable (e.g. only once).
   - Query and perform joins on data directly from SQL Data Warehouse.
4. Implement various performance and optimization techniques
   - Caching interim partial results to be reused in subsequent stages.
   - Manipulating the size and number of the partitions that help parallelize distributed data processing executors.
5. Explore the features and Spark Databricks UI

3. Week 5 and Week 6 Task

3. you will be performing the following tasks:

1. Pre-process the data.
2. Use the built-in Correlation function to find correlation between columns of the dataset.
3. Perform the following transformations:
   - Indexing
   - Encoding
   - Assembling into vectors
   - Normalizing the vectors

4. Week 7 and Week 8 Task

This tutorial covers the following tasks:

- Attach libraries for Event Hubs
- Send data to Event Hubs
- Read data from Event Hubs
- Perform real-time analytics using Spark Structured Streaming.
- Perform aggregations and filtering operations on Streaming dataset.
- Perform Windowed operations based on event time.
- Join operation between static and streaming data.
- Performing predictions on streaming dataset via model trained on static data.

5. Week 9 and Week 10 Task

This tutorial covers the following tasks:

- Databricks Delta Introduction - Parquet format & Transaction Logs
- Operations on Delta Tables - Create, Read, Append data
- Table Metadata - Table History, Table Details
- ACID Properties Support - Updates and Upserts
- High performance Spark queries - Compaction, ZOrdering,