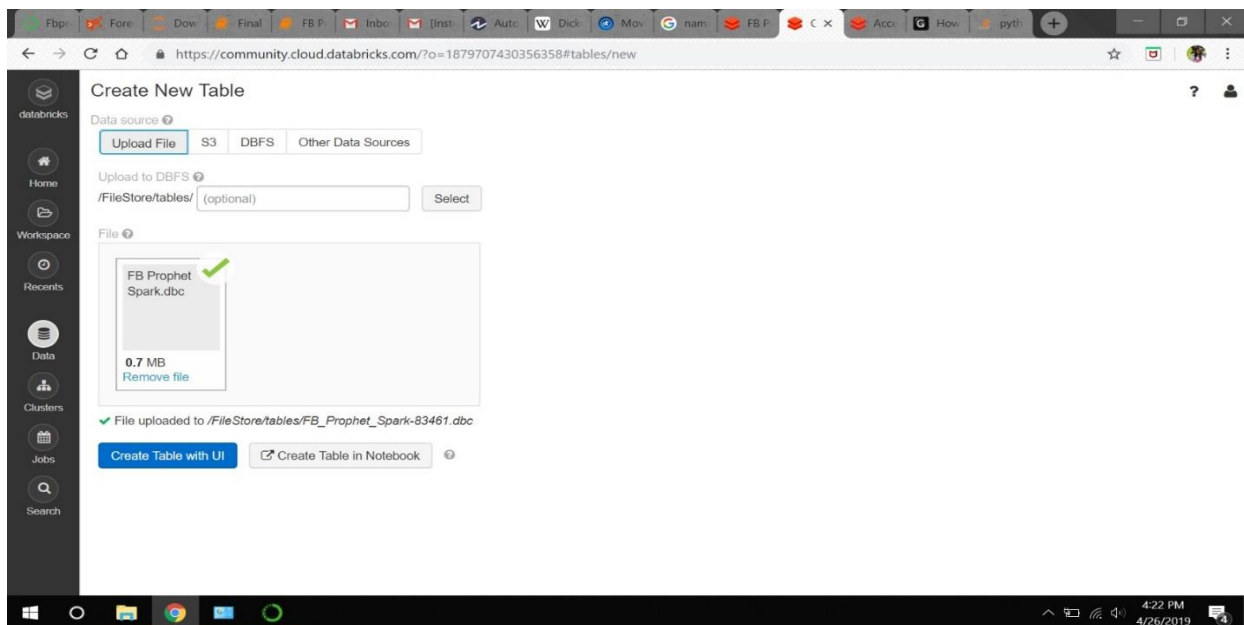
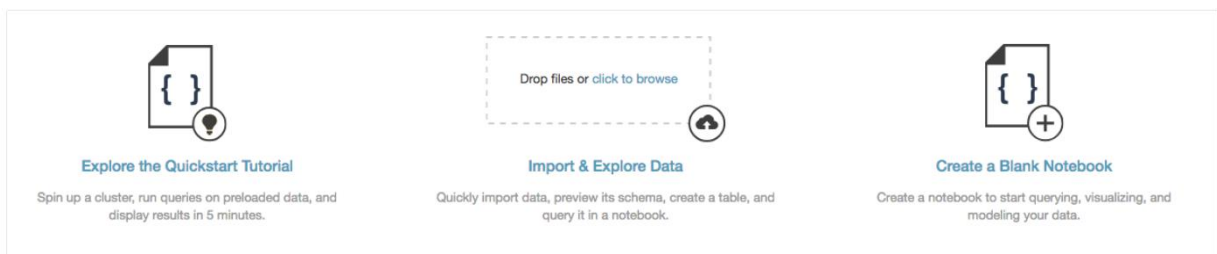


How to run the code in Databricks-

Databricks- Databricks is a managed platform for running Apache Spark and provides a host of features to help its users be more productive with Spark. It's a point and click platform for those that prefer a user interface. However, this UI is accompanied by a sophisticated API for those that want to automate aspects of their data workloads with automated jobs.

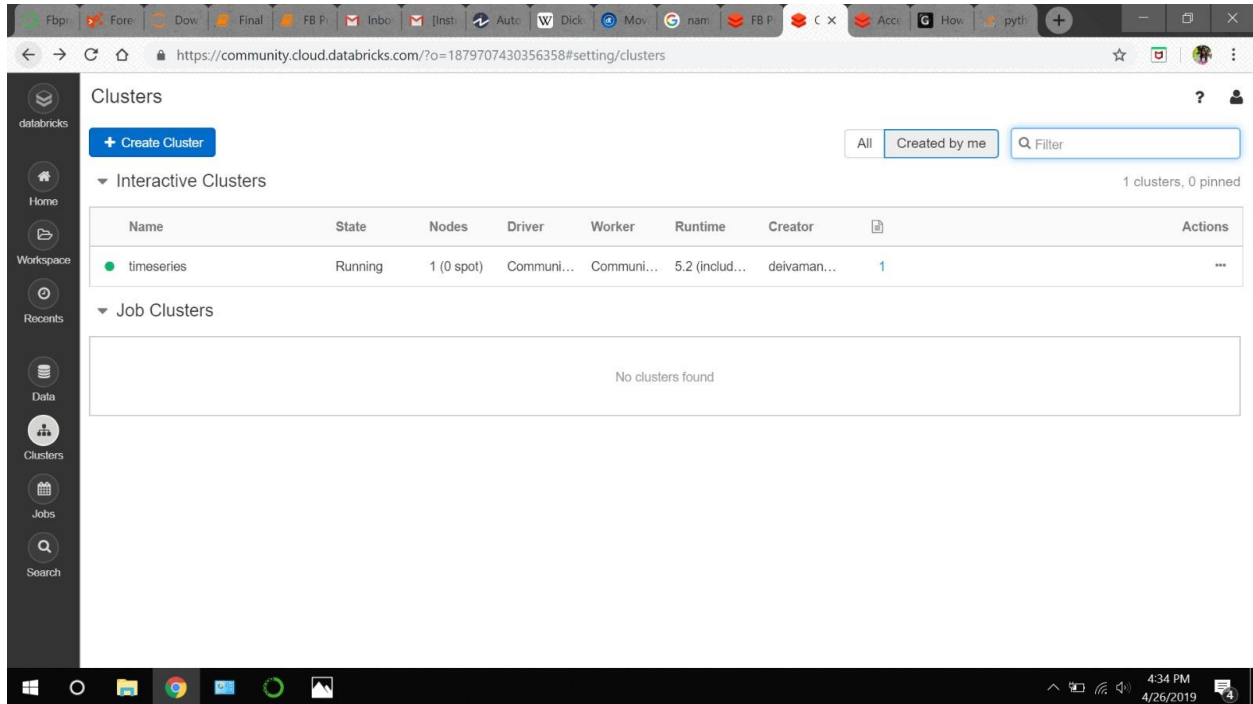


1. Import data- Drop files into or click to browse in Import & Explore data box on landing page.



2. Creating Cluster:

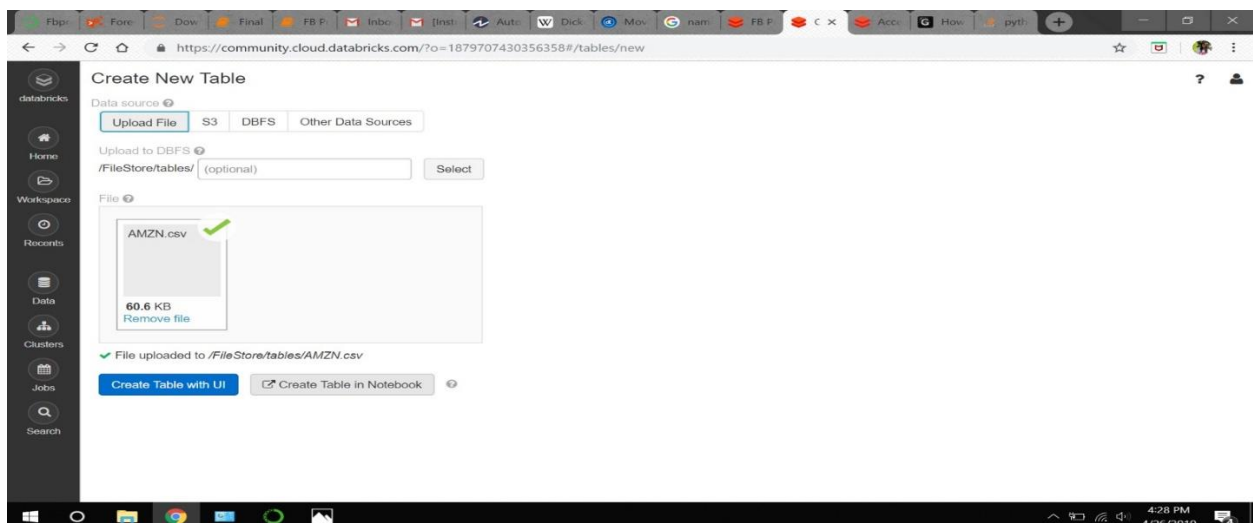
In Databricks you can create two different types of clusters: standard and high concurrency. Standard clusters are the default and can be used with Python, R, Scala, and SQL. High-concurrency clusters are tuned to provide the efficient resource utilization, isolation, security, and the best performance for sharing by multiple concurrently active users. High concurrency clusters support only SQL, Python, and R languages.



The screenshot shows the Databricks Clusters management interface. The left sidebar contains navigation icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main content area is titled 'Clusters' and includes a '+ Create Cluster' button. Below this, there are two sections: 'Interactive Clusters' and 'Job Clusters'. The 'Interactive Clusters' section shows a table with one cluster named 'timeseries' in a 'Running' state, with 1 (0 spot) nodes, using a 'Community' driver and worker, and a runtime of '5.2 (includ...'. The 'Job Clusters' section is empty, displaying 'No clusters found'.

Name	State	Nodes	Driver	Worker	Runtime	Creator	Actions
timeseries	Running	1 (0 spot)	Communi...	Communi...	5.2 (includ...	deivaman...	1

3 Importing CSV file-



The screenshot shows the 'Create New Table' interface in Databricks. The 'Data source' section has 'Upload File' selected. The 'Upload to DBFS' section shows the file path '/FileStore/tables/' and a 'Select' button. A file named 'AMZN.csv' (60.6 KB) is shown as uploaded, with a green checkmark and a 'Remove file' link. Below the file upload, there is a message 'File uploaded to /FileStore/tables/AMZN.csv' and two buttons: 'Create Table with UI' and 'Create Table in Notebook'.

4. Importing Libraries- The below libraries are the required for our model. Steps to getting libraries installed- Select the running cluster which you have created earlier then click on the libraries you want to install for your model and done(required libraries installed)

<input type="checkbox"/>	Name	Type	Status	Source
<input type="checkbox"/>	adfuller	PyPI	-	
<input type="checkbox"/>	fbprophet	PyPI	-	
<input type="checkbox"/>	fix_yahoo_finance	PyPI	-	
<input type="checkbox"/>	keras	PyPI	-	
<input type="checkbox"/>	matplotlib	PyPI	-	
<input type="checkbox"/>	numpy	PyPI	-	
<input type="checkbox"/>	pandas	PyPI	-	
<input type="checkbox"/>	PrettyTable	PyPI	-	
<input type="checkbox"/>	pyspark	PyPI	-	
<input type="checkbox"/>	spark	PyPI	-	
<input type="checkbox"/>	statsmodel	PyPI	-	
<input type="checkbox"/>	statsmodels	PyPI	-	
<input type="checkbox"/>	stattools	PyPI	-	
<input type="checkbox"/>	tensorflow	PyPI	-	

5. View databases and tables- Click on the sidebar. Databricks selects any running cluster to which you have access. The Databases folder displays the list of databases with the default database selected. The Tables folder displays the list of tables in the default database.

The screenshot shows the Databricks web interface. The browser address bar displays the URL: https://community.cloud.databricks.com/?o=1879707430356358#table/default/amzn_csv. The sidebar on the left includes navigation options: Recents, Data, Clusters, Jobs, and Search. The main content area is titled 'Table: amzn_csv' and includes a 'Refresh' button. Below this, the cluster information is shown: 'timeseries (6 GB, Running, 5.2 (includes Apache Spark 2.4.0,...)'. The 'Schema:' section displays a table with three columns: 'col_name', 'data_type', and 'comment'. The 'Sample Data:' section displays a table with eight columns: 'Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', and 'Volume'.

col_name	data_type	comment
Date	timestamp	null
Open	string	null
High	string	null
Low	string	null
Close	string	null
Adj Close	string	null
Volume	string	null

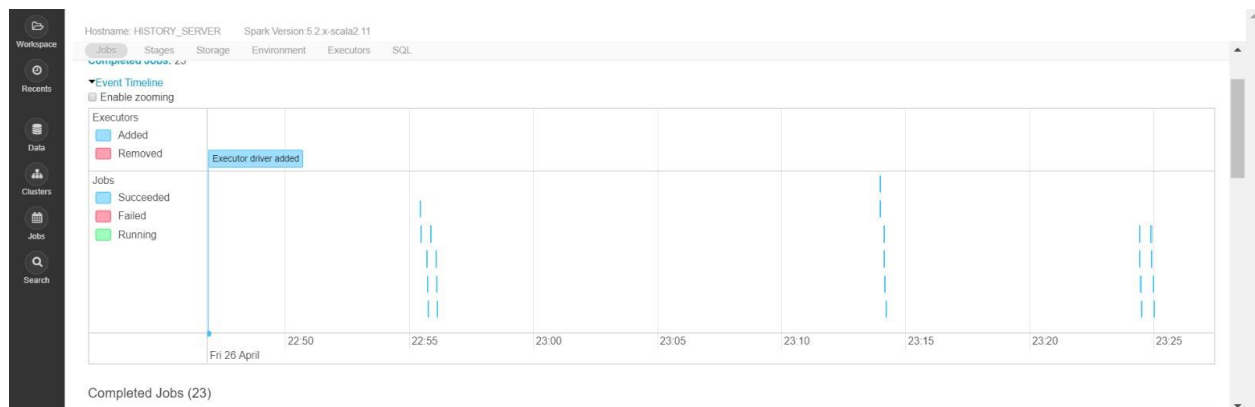
Date	Open	High	Low	Close	Adj Close	Volume
2016-04-25T00:00:00.000+0000	616.6099849999999	626.97998	616.25	626.200012	626.200012	2682900
2016-04-26T00:00:00.000+0000	626.169983	626.75	614.880005	616.880005	616.880005	2521400
2016-04-27T00:00:00.000+0000	611.799988	615.950012	601.280029	606.570007	606.570007	4068800
2016-04-28T00:00:00.000+0000	615.539978	626.799988	599.200012	602.0	602.0	7872600
2016-04-29T00:00:00.000+0000	666.0	669.97998	654.0	659.590027	659.590027	10310700
2016-05-02T00:00:00.000+0000	663.919983	685.5	662.030029	683.849976	683.849976	6578500

6. Notebook in Databricks

The image shows a Jupyter Notebook interface. On the left is a sidebar with icons for Workspace, Recents, Data, Clusters, Jobs, and Search. The main area is divided into two sections. The top section, labeled 'Cmd 40', contains the text: 'We are evaluating combinations of p, d and q values for ARIMA model to find the best combination of p, d, q values for our model'. The bottom section, labeled 'Cmd 41', contains a Python code cell with the following code:

```
1 import warnings
2 from pandas import Series
3 from statsmodels.tsa.arima_model import ARIMA
4 from sklearn.metrics import mean_squared_error
5 from math import sqrt
6
7 order=(2,1,1)
8 # evaluate an ARIMA model for a given order (p,d,q) and return RMSE
9 def evaluate_arima_model(X, arima_order):
10     # prepare training dataset
11     X = X.astype('float32')
12     train_size = int(len(X) * 0.50)
13     train, test = X[0:train_size], X[train_size:]
14     history = [x for x in train]
15     # make predictions
16     predictions = list()
17     for t in range(len(test)):
18         model = ARIMA(history, order=arima_order)
19         # model_fit = model.fit(dispatch=0)
20         model_fit = model.fit(trend='nc', dispatch=0)
21         what = model_fit.forecast()[0]
```

7. Spark UI



8. Computational Time

SQL

Completed Queries

ID	Description		Submitted	Duration	Jobs
30	collectedResult at OutputAggregator.scala:132	+details	2019/04/26 23:28:20	3 ms	
29	toDF at CustomDisplayTypes.scala:77	+details	2019/04/26 23:28:20	0 ms	
28	collectedResult at OutputAggregator.scala:132	+details	2019/04/26 23:28:20	17 ms	
27	toDF at CustomDisplayTypes.scala:77	+details	2019/04/26 23:28:20	0 ms	
25	sql at SQLDriverLocal.scala:87	+details	2019/04/26 23:28:20	27 ms	
26	sql at SQLDriverLocal.scala:87	+details	2019/04/26 23:28:20	9 ms	
24	toPandas at <command-2828119668611524>-1	+details	2019/04/26 23:24:59	0.1 s	22
23	collectedResult at OutputAggregator.scala:132	+details	2019/04/26 23:24:51	0.2 s	20
22	load at NativeMethodAccessorImpl.java:0	+details	2019/04/26 23:24:51	0.2 s	19
21	toPandas at <command-2828119668611524>-1	+details	2019/04/26 23:24:28	0.1 s	18
20	collectedResult at OutputAggregator.scala:132	+details	2019/04/26 23:24:23	0.2 s	16
19	load at NativeMethodAccessorImpl.java:0	+details	2019/04/26 23:24:23	0.2 s	15
18	toPandas at <command-2828119668611524>-1	+details	2019/04/26 23:14:11	0.2 s	14