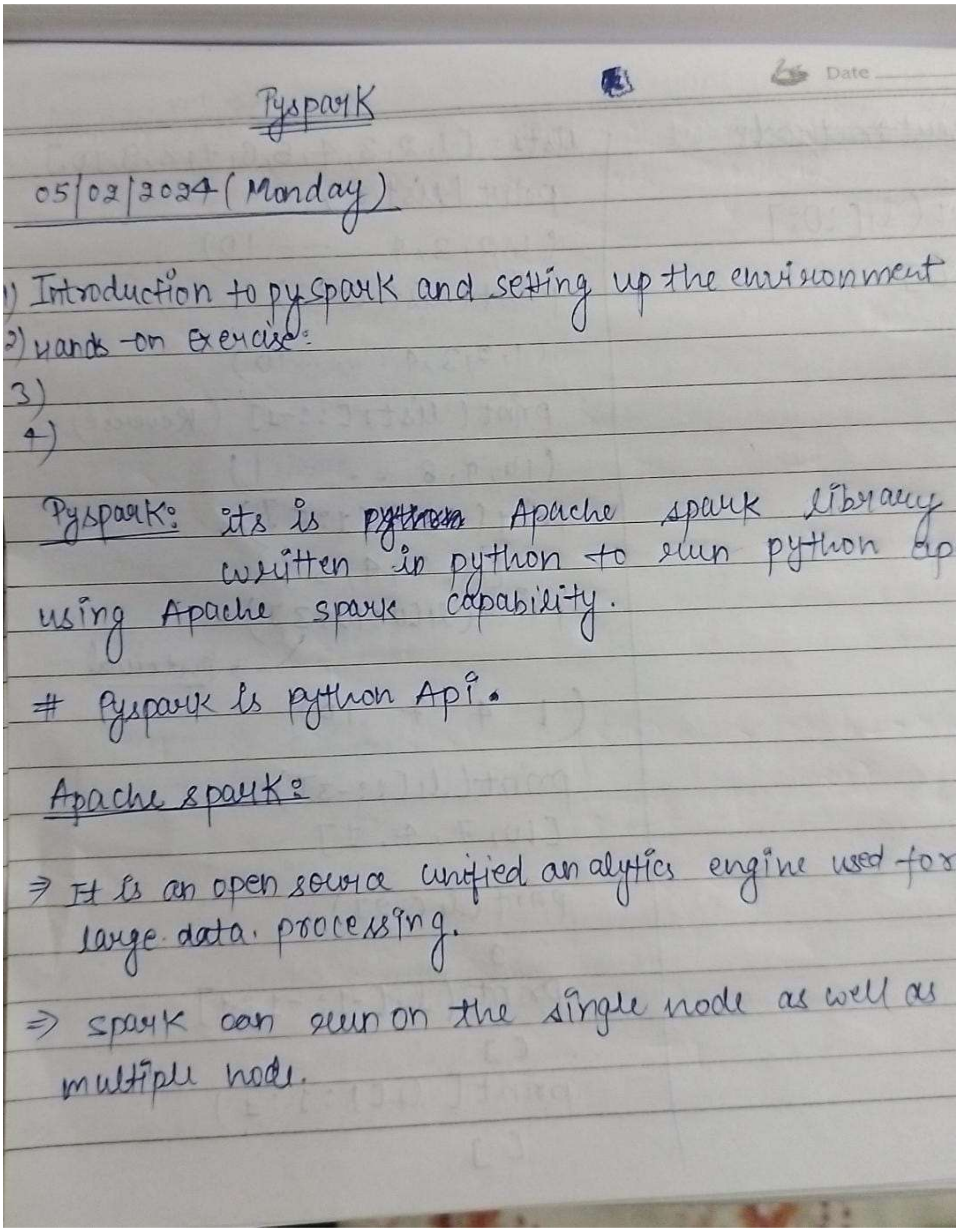
Name: Abhishek Kanoujia

DATA ENGINEERING BATCH 1

DAY 13 ASSIGNMENT

Class hand written notes:-



Date

Pyspark

05/02/2024 (Monday)

1) Introduction to pyspark and setting up the environment
2) hands-on Exercise:
3)
4)

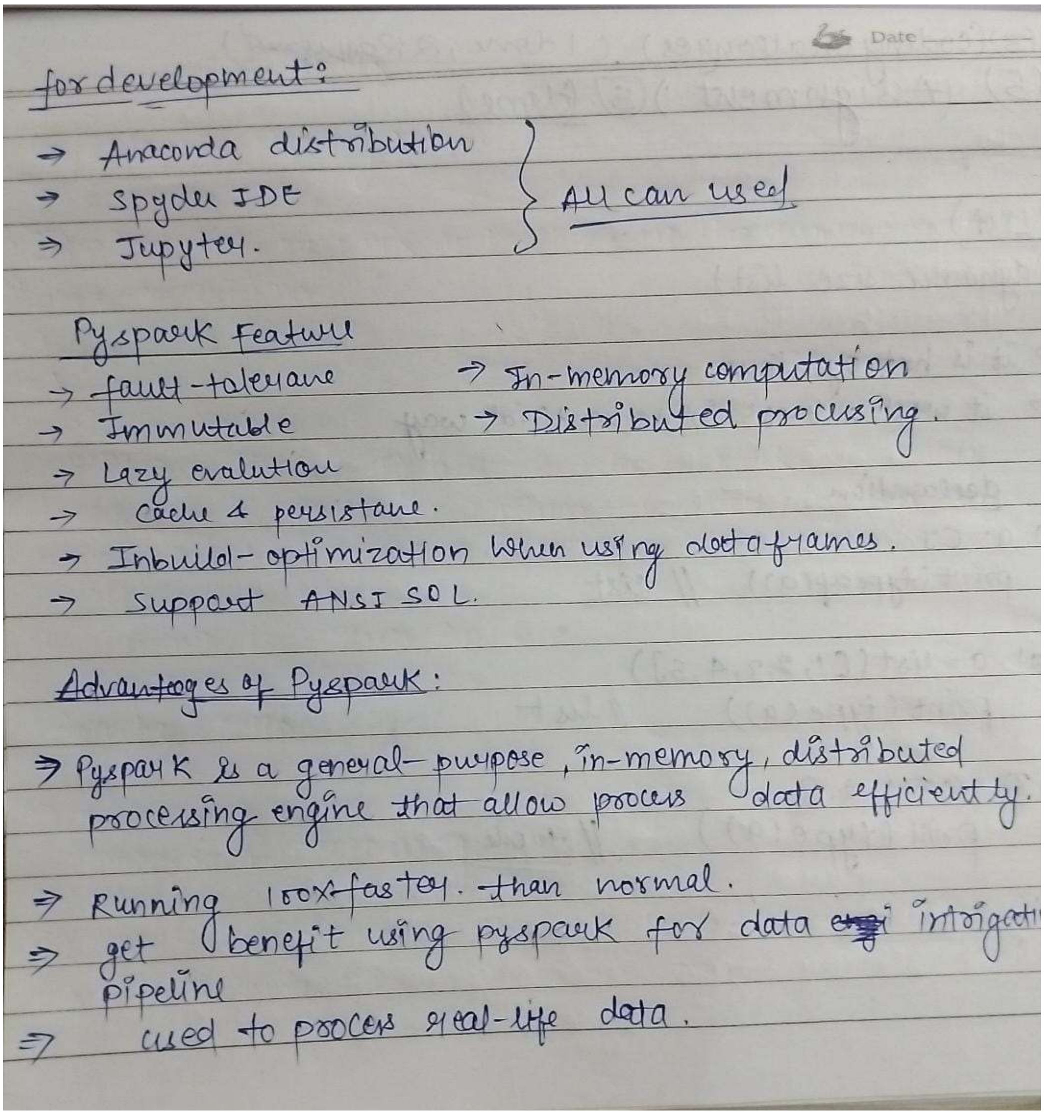
Pyspark: its is ~~python~~ Apache spark library written in python to run python ap using Apache spark capability.

Pyspark is python Api.

Apache spark:

⇒ It is an open sourca unified analytics engine used for large data processing.

⇒ spark can run on the single node as well as multiple node.



for development:

- → Anaconda distribution
- → Spyder IDE
- → Jupyter.

} All can used

Pyspark Feature

- → fault-tolerance
- → Immutable
- → Lazy evalution
- → Cache & persistance.
- → Inbuild-optimization when using dataframes.
- → Support ANSI SQL.
- → In-memory computation
- → Distributed processing.

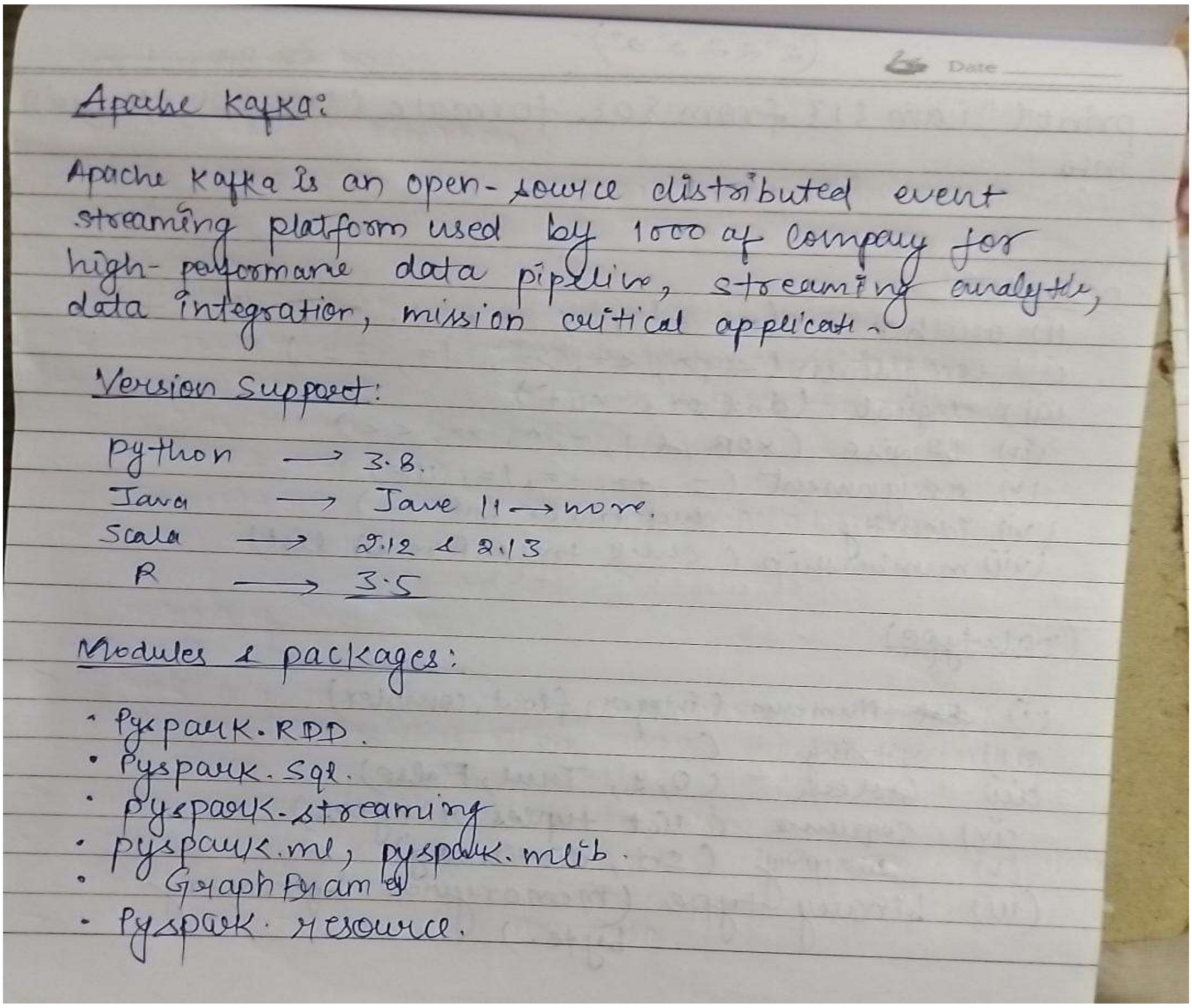
Advantages of Pyspark:

⇒ Pyspark is a general-purpose, in-memory, distributed processing engine that allow process data efficiently.

⇒ Running 100x faster. than normal.

⇒ get benefit using pyspark for data intrigati
pipeline

⇒ used to process real-life data.



Apache kafka:

Apache kafka is an open-source distributed event streaming platform used by 1000 of company for high-performane data pipeline, streaming analytic, data integration, mission critical applicat-.

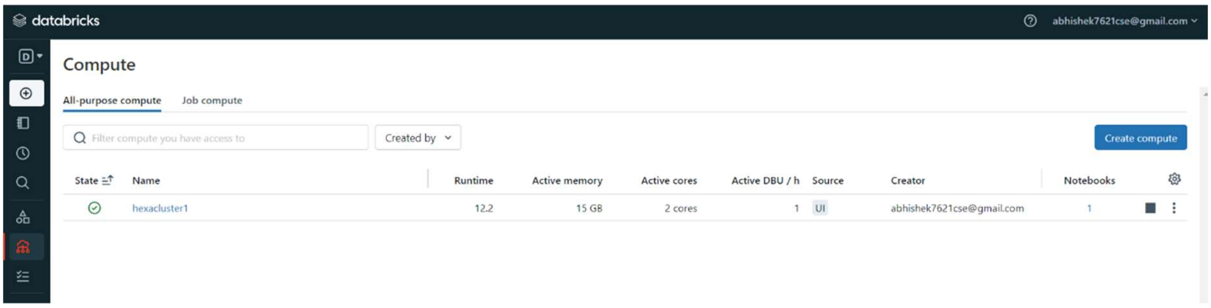
Version Support:

Python →	3.8.
Java →	Jave 11 → more.
Scala →	2.12 & 2.13
R →	3.5

Modules & packages:

- Pyspark.RDD.
- Pyspark.Sql.
- pyspark.streaming
- pyspark.ml, pyspark.mlib.
- GraphFrame
- Pyspark.resource.

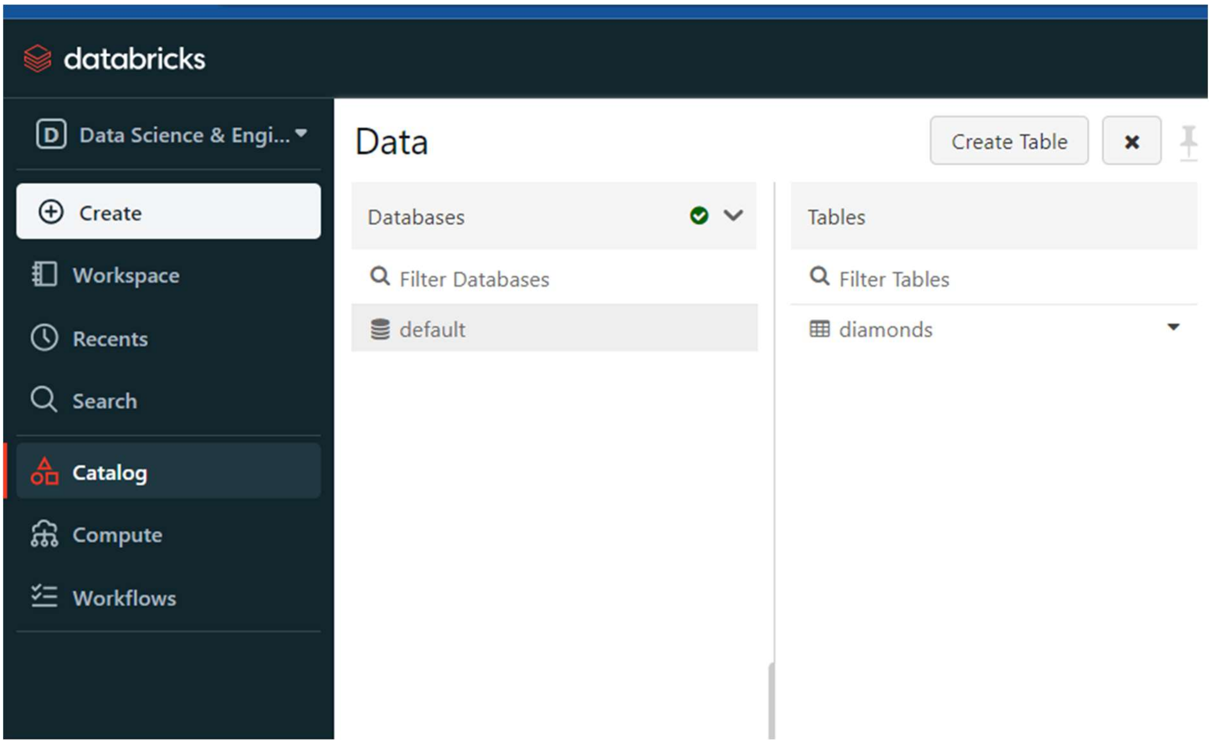
Create cluster in databricks:-



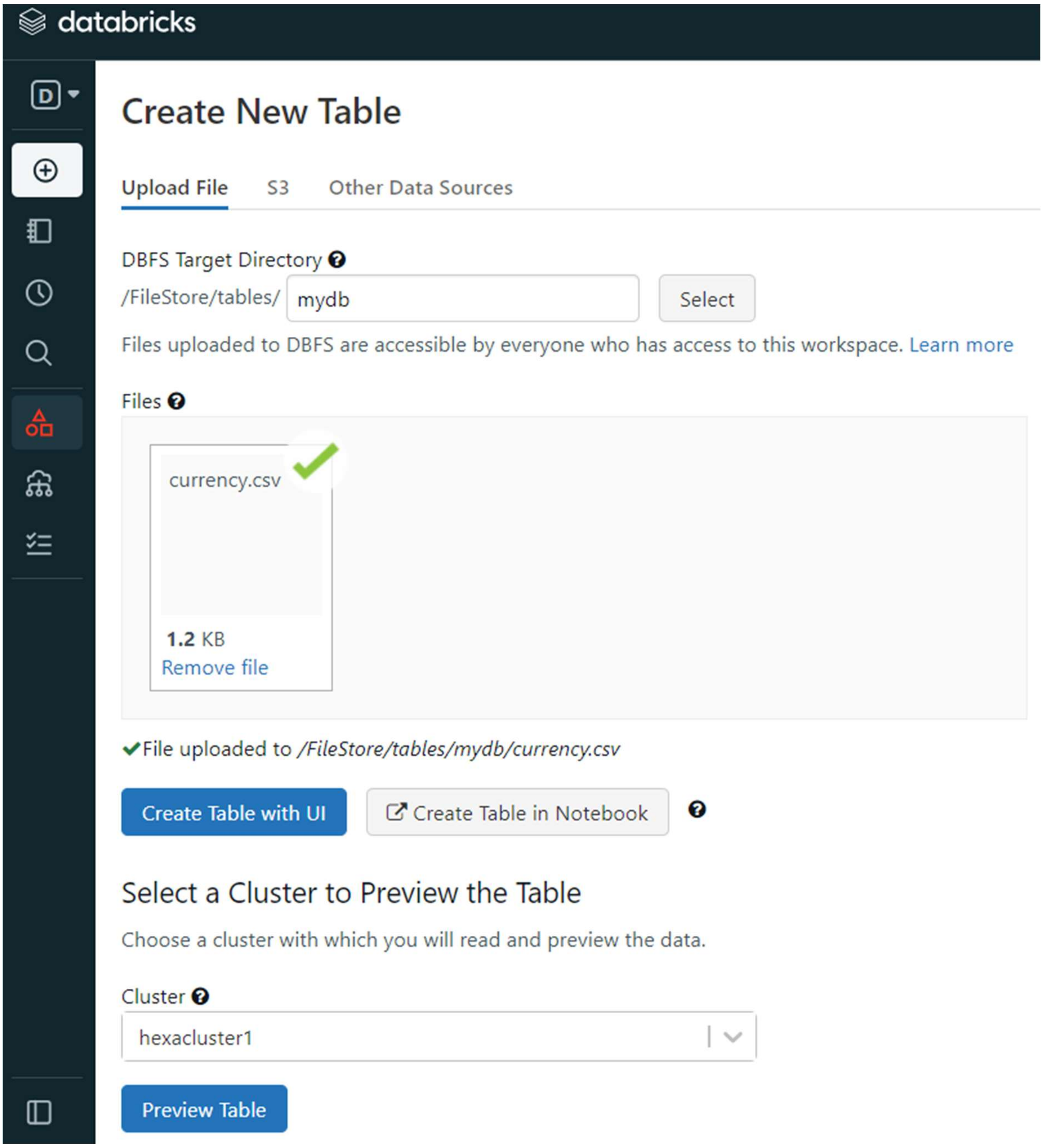
State	Name	Runtime	Active memory	Active cores	Active DBU / h	Source	Creator	Notebooks
✓	hexacluster1	12.2	15 GB	2 cores	1	UI	abhishek7621cse@gmail.com	1

Create table using csv file in databricks:-

Step:1


databricks
Data Science & Engi...
Create
Workspace
Recents
Search
Catalog
Compute
Workflows
Data
Create Table
Databases
Filter Databases
default
Tables
Filter Tables
diamonds

Step:2



Create New Table

DBFS Target Directory

/FileStore/tables/ mydb

Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files

currency.csv

1.2 KB

✓File uploaded to */FileStore/tables/mydb/currency.csv*

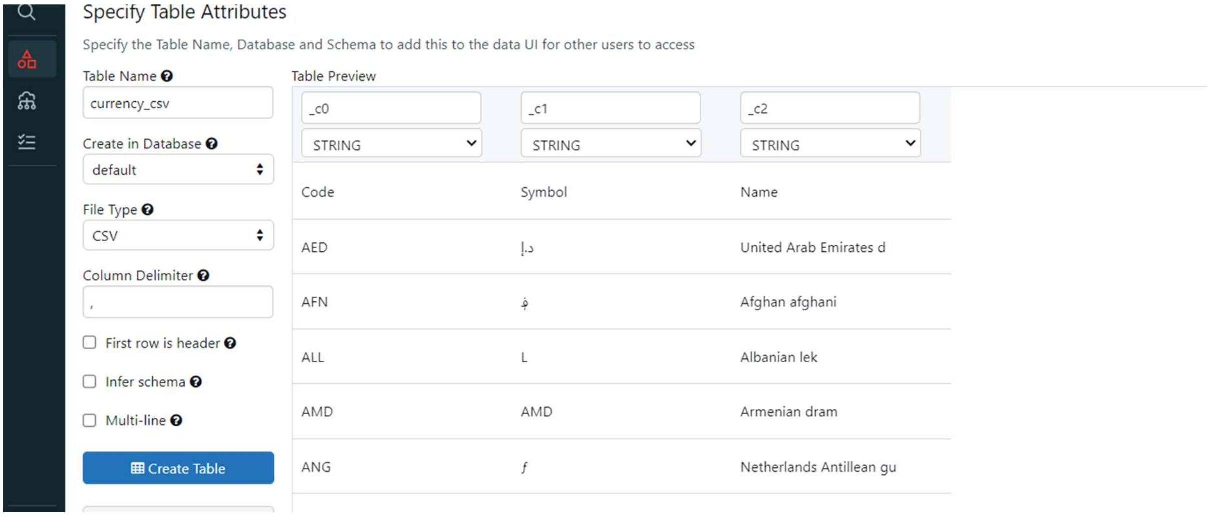
Select a Cluster to Preview the Table

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

Cluster

hexacluster1

Step:3



Specify Table Attributes

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

Table Name: currency_csv

Create in Database: default

File Type: CSV

Column Delimiter: ,

☐ First row is header

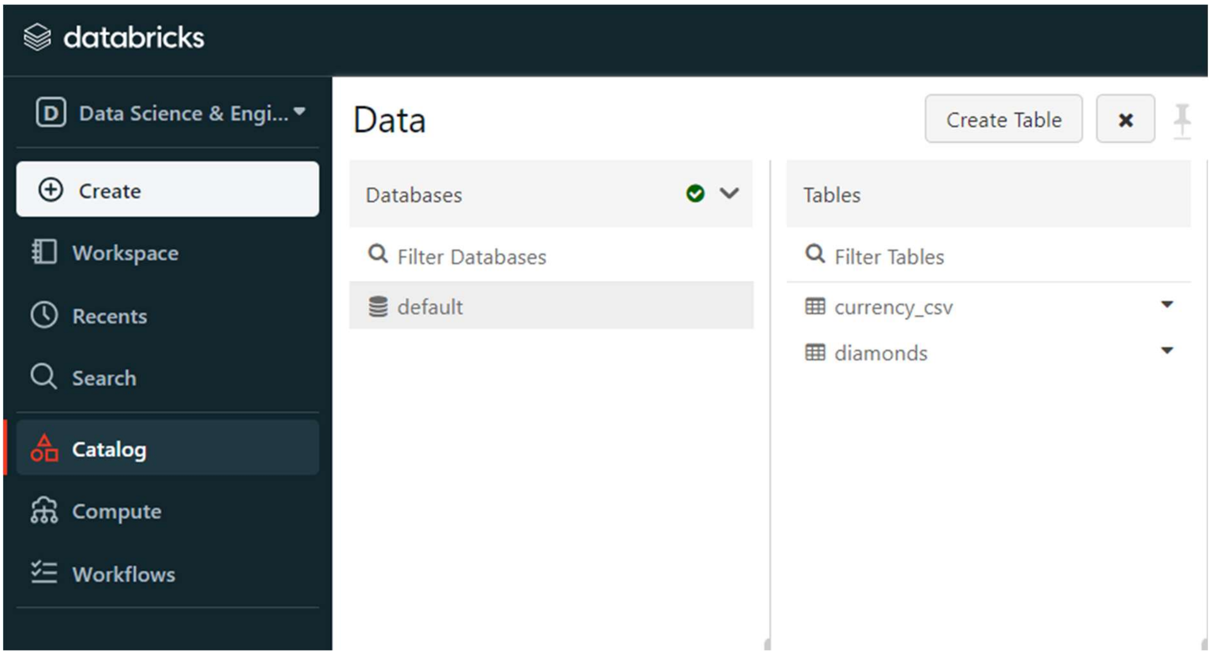
☐ Infer schema

☐ Multi-line

Table Preview

_c0	_c1	_c2
STRING	STRING	STRING
Code	Symbol	Name
AED	د.إ	United Arab Emirates d
AFN	؋	Afghan afghani
ALL	L	Albanian lek
AMD	AMD	Armenian dram
ANG	ƒ	Netherlands Antillean gu

Step:4



Data

Databases

Filter Databases

default

Tables

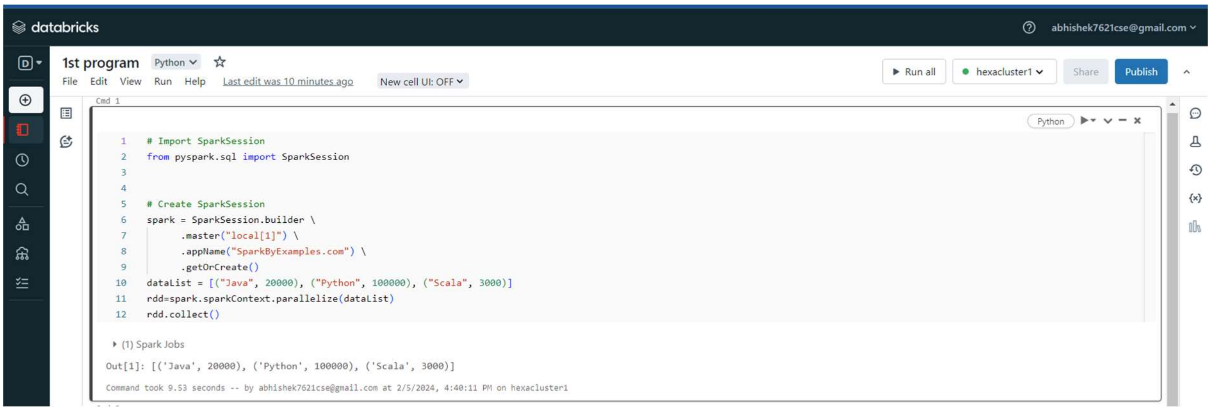
Filter Tables

currency_csv

diamonds

Table created successfully.

First program :-



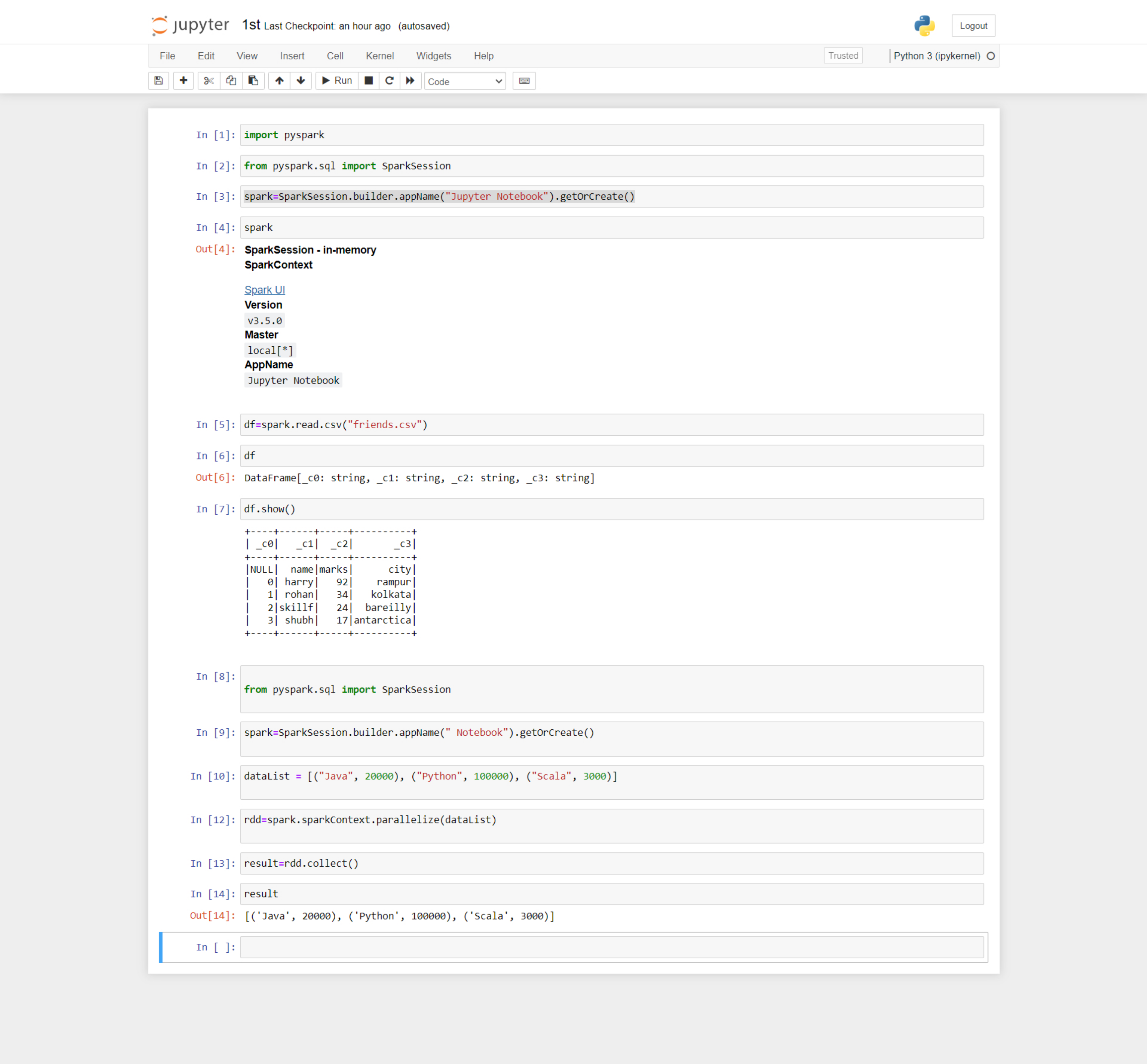
```
# Import SparkSession
from pyspark.sql import SparkSession


# Create SparkSession
spark = SparkSession.builder \
      .master("local[1]") \
      .appName("SparkByExamples.com") \
      .getOrCreate()
dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]
rdd=spark.sparkContext.parallelize(dataList)
rdd.collect()
```

(1) Spark Jobs

```
Out[1]: [('Java', 20000), ('Python', 100000), ('Scala', 3000)]

Command took 9.53 seconds -- by abhishek7621cse@gmail.com at 2/5/2024, 4:40:11 PM on hexacluster1
```



```
In [1]: import pyspark

In [2]: from pyspark.sql import SparkSession

In [3]: spark=SparkSession.builder.appName("Jupyter Notebook").getOrCreate()

In [4]: spark
```

Out[4]: **SparkSession - in-memory**

SparkContext

Spark UI

Version

`v3.5.0`

Master

`local[*]`

AppName

`Jupyter Notebook`

```
In [5]: df=spark.read.csv("friends.csv")

In [6]: df

Out[6]: DataFrame[_c0: string, _c1: string, _c2: string, _c3: string]

In [7]: df.show()

+----+------+-----+----------+
| _c0|   _c1|  _c2|       _c3|
+----+------+-----+----------+
|NULL|  name|marks|      city|
|   0| harry|   92|    rampur|
|   1| rohan|   34|   kolkata|
|   2|skillf|   24|  bareilly|
|   3| shubh|   17|antarctica|
+----+------+-----+----------+

In [8]:
        from pyspark.sql import SparkSession

In [9]: spark=SparkSession.builder.appName(" Notebook").getOrCreate()

In [10]: dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]

In [12]: rdd=spark.sparkContext.parallelize(dataList)

In [13]: result=rdd.collect()

In [14]: result

Out[14]: [('Java', 20000), ('Python', 100000), ('Scala', 3000)]

In [ ]:
```