

PYSPARK

Installation Of Pyspark

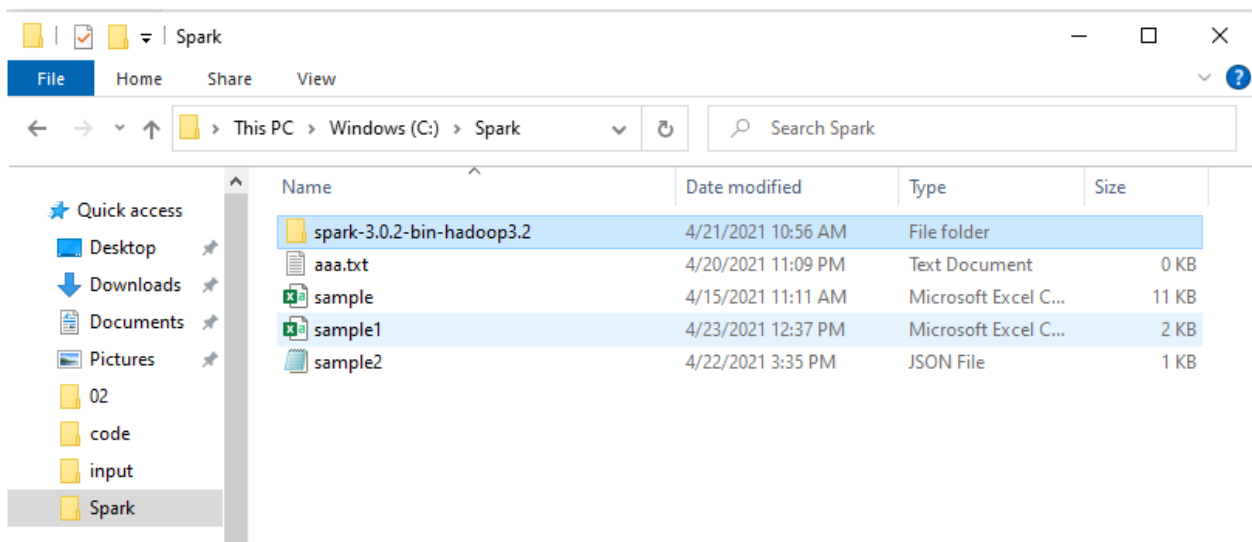
Step 1: In order to install pyspark, we need to have java on our system, if not first need to download java. Check if the java is present by giving a command java -version on command prompt.

```
Command Prompt
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

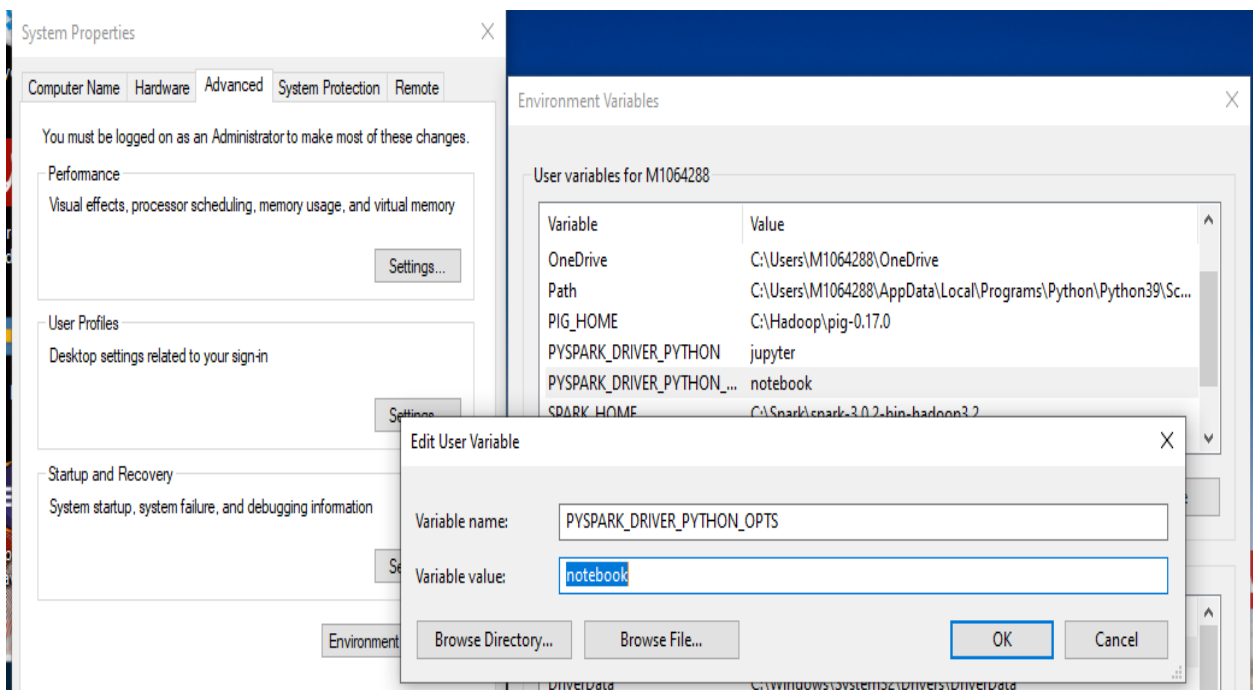
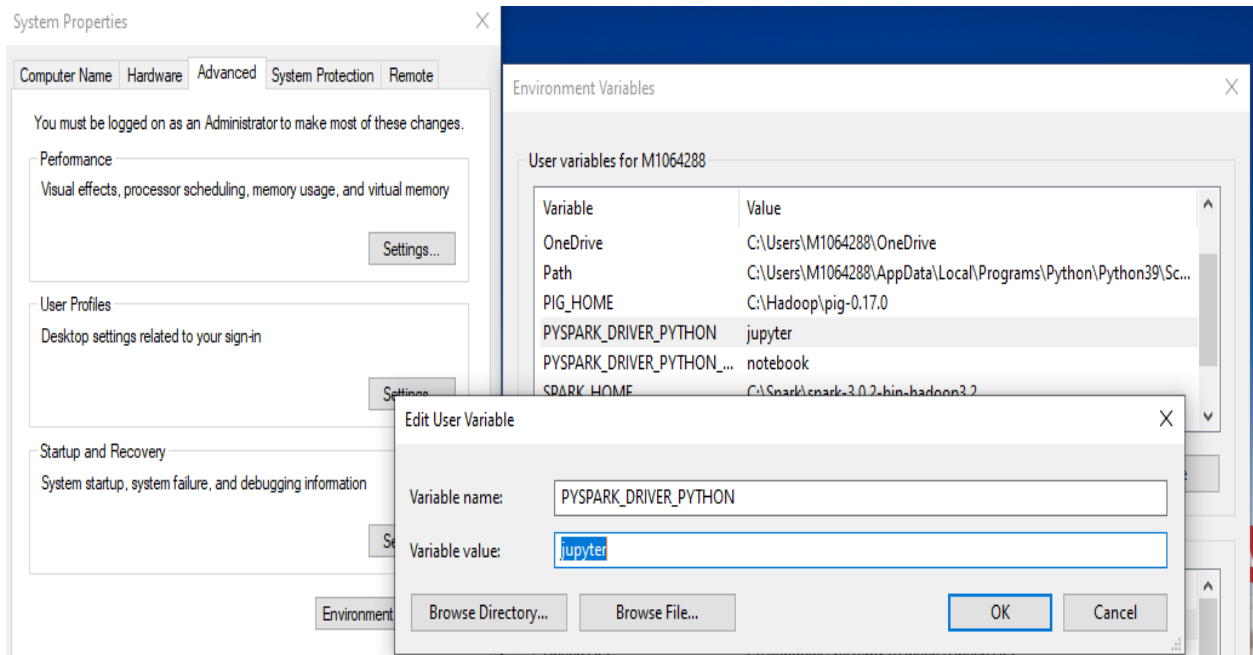
C:\Users\M1064288>java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

Step2: To download pyspark for the windows then click on the given link
<https://mirrors.estointernet.in/apache/spark/spark-3.0.2/spark-3.0.2-bin-hadoop3.2.tgz>

Step 3: After downloading extract the files to a folder.



Step 4: Set environment variables for pyspark in order to open it in jupyter platform. If the Hadoop environment variable is not set earlier then need to create it as well because spark runs on top of Hadoop.



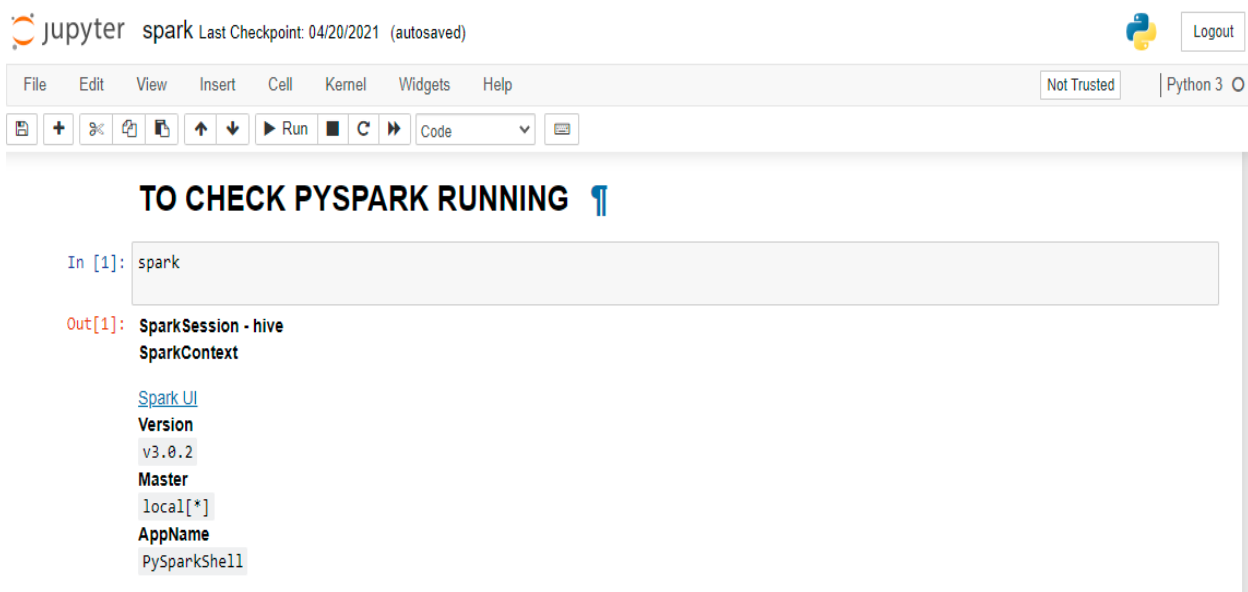
Step 5: Check if pyspark is installed properly and opens in jupyter by giving command as pyspark.

```
C:\Windows\System32\cmd.exe - pyspark

C:\Spark\spark-3.0.2-bin-hadoop3.2>pyspark
[W 11:51:31.497 NotebookApp] Terminals not available (error was No module named 'winpty.cython')
[I 11:51:31.638 NotebookApp] Serving notebooks from local directory: C:\Spark\spark-3.0.2-bin-hadoop3.2
[I 11:51:31.638 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 11:51:31.638 NotebookApp] http://localhost:8888/?token=42669b9269bb9af005f5f5bdc1d818146566cffe3ed8a8dc7
[I 11:51:31.638 NotebookApp] or http://127.0.0.1:8888/?token=42669b9269bb9af005f5f5bdc1d818146566cffe3ed8a8dc7
[I 11:51:31.638 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:51:31.678 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/M1064288/AppData/Roaming/jupyter/runtime/nbserver-19376-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=42669b9269bb9af005f5f5bdc1d818146566cffe3ed8a8dc7
    or http://127.0.0.1:8888/?token=42669b9269bb9af005f5f5bdc1d818146566cffe3ed8a8dc7
```

Step 6: To check if the spark is working properly in jupyter than use a command spark.If it provides the details



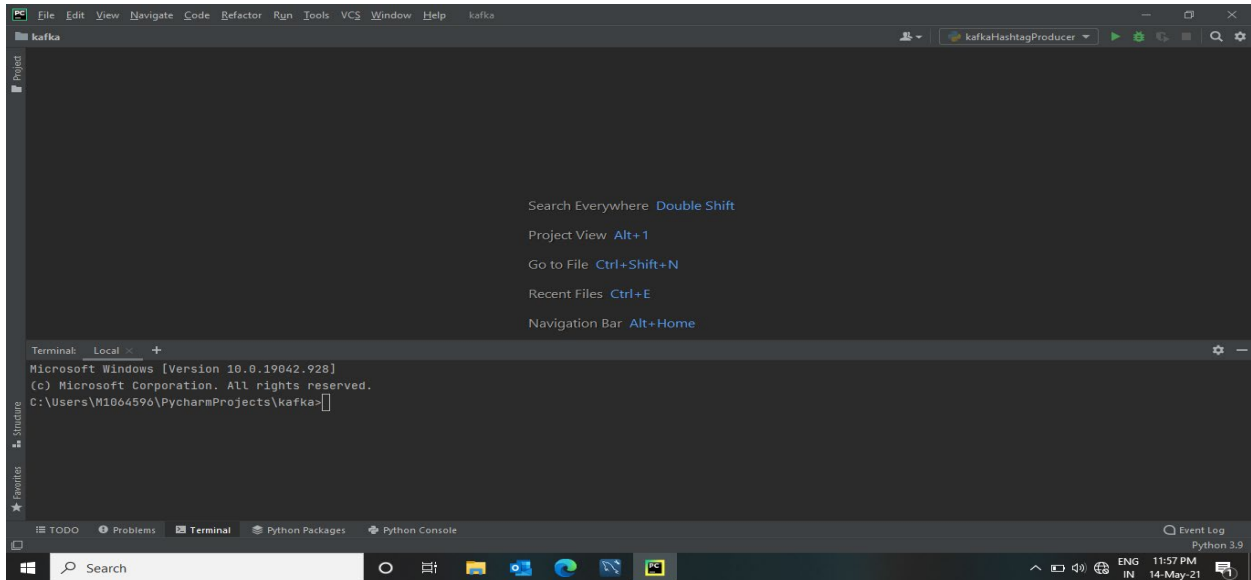
The screenshot shows the Jupyter Notebook interface with the title 'jupyter spark'. The last checkpoint is '04/20/2021 (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The main content area displays the command 'spark' entered in the input cell. The output cell shows the following details:

```
Out[1]: SparkSession - hive
SparkContext

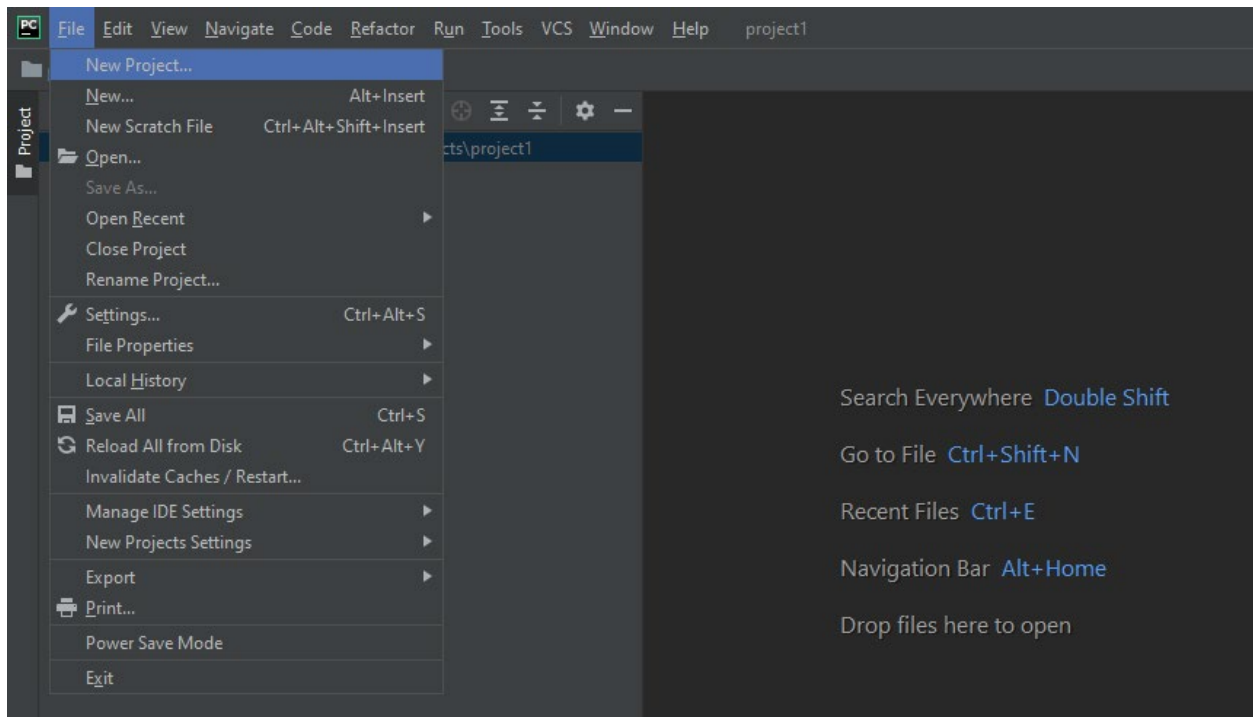
Spark UI
Version
v3.0.2
Master
local[*]
AppName
PySparkShell
```

Ability To Set Up Pyspark Project In Pycharm

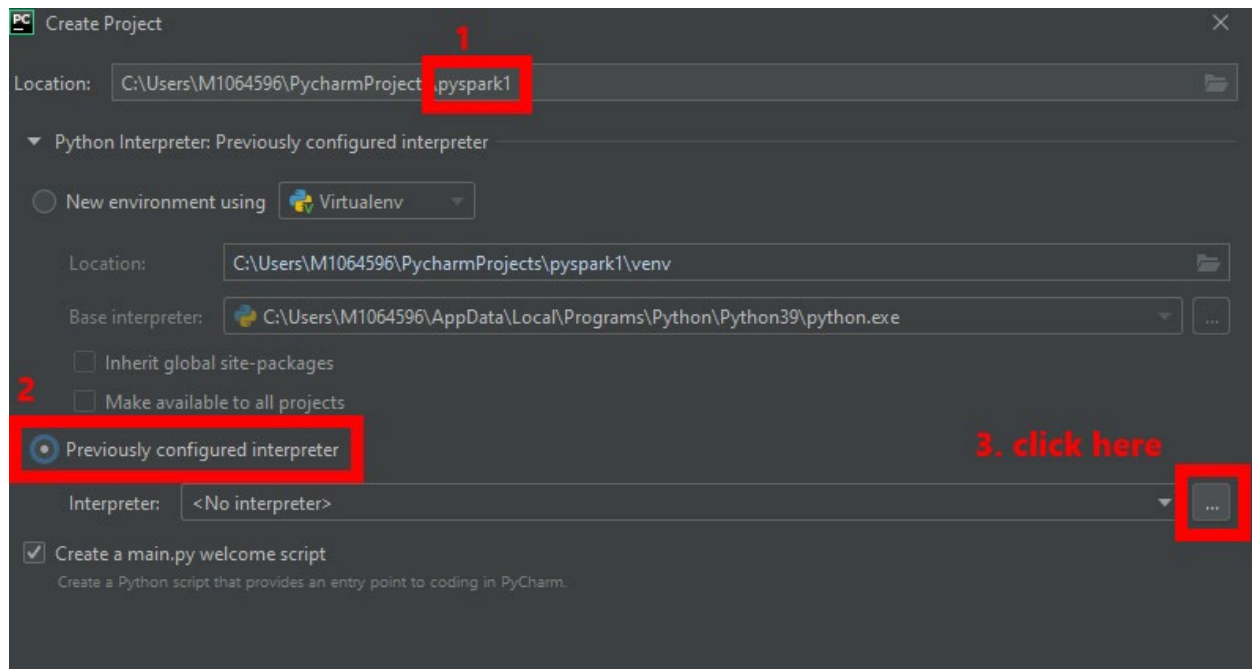
Step 1: Open PyCharm.



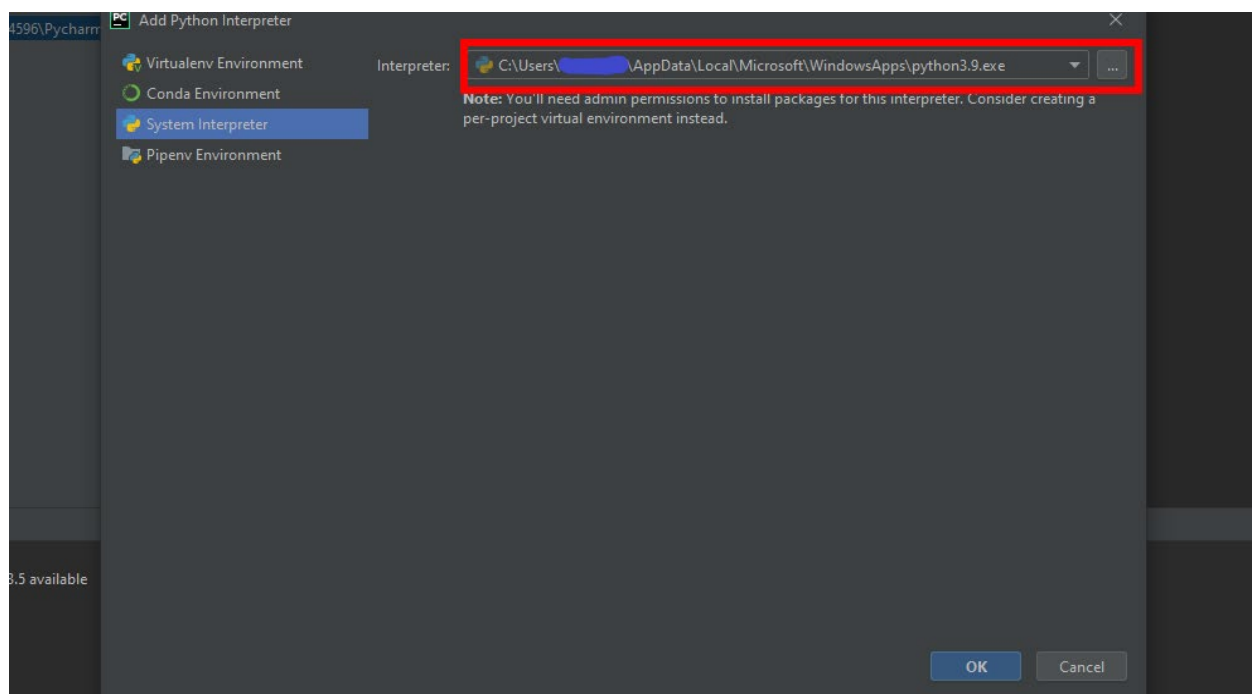
Step 2: Click on File > New Project.



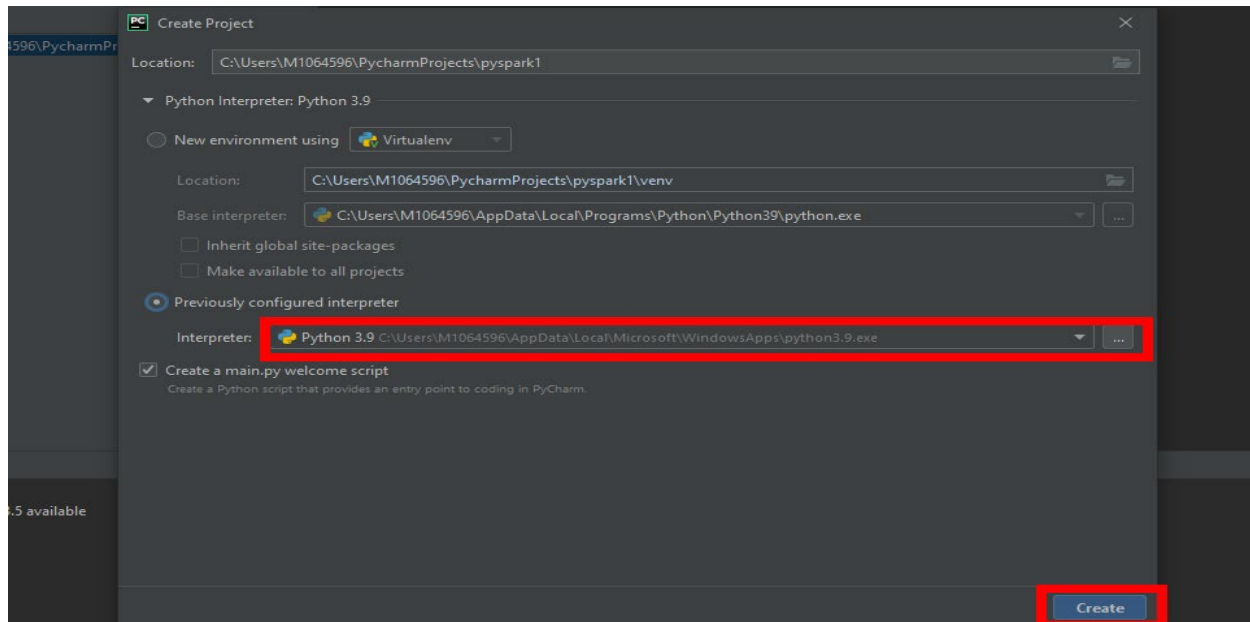
Step 3: Give a name to project, Select 'previously configured interpreter' and click on the drop icon.



Step 4: New drop box appears, select System interpreter, select the python.exe file from the location where python installed in your local system. Then click ok.

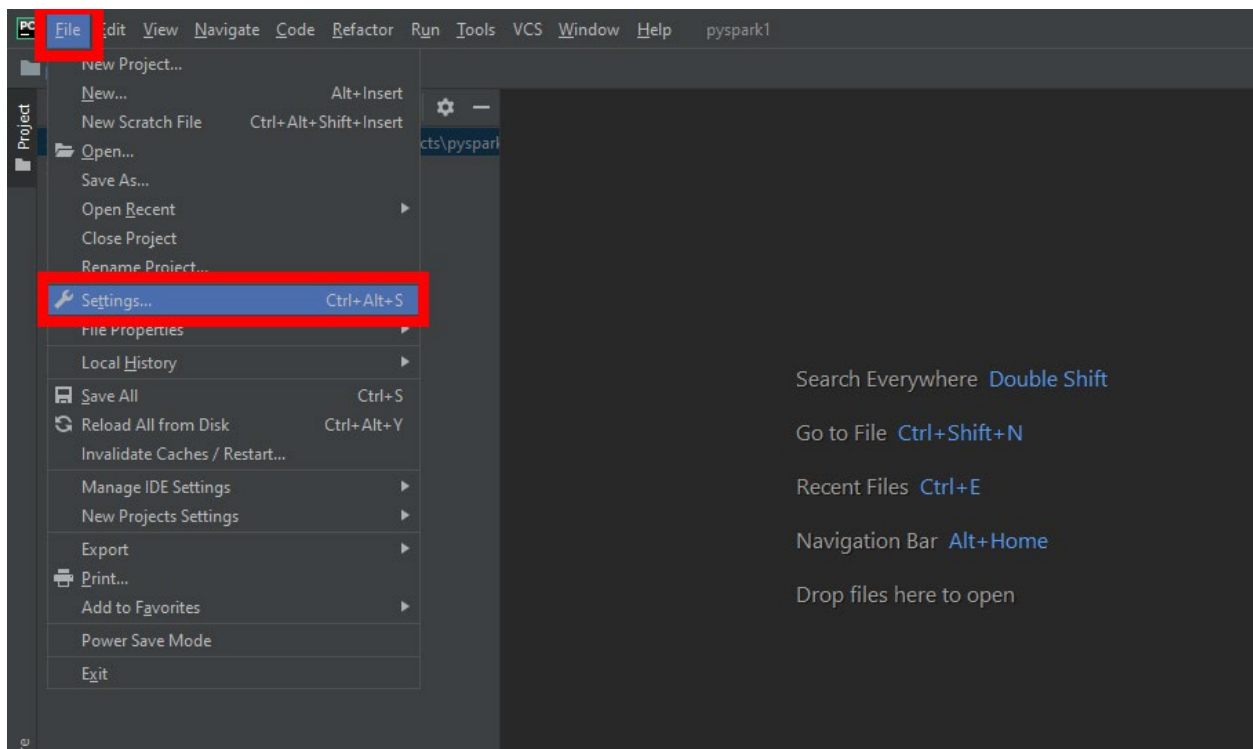


Step 5: Observe the System interpreter added, and click on create.

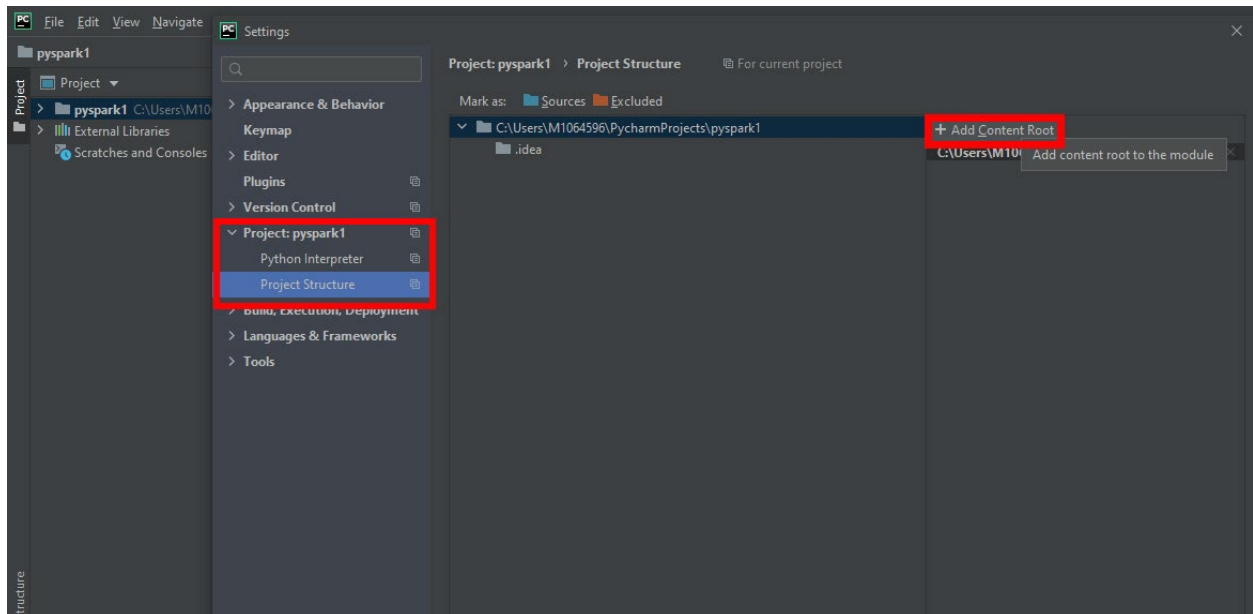


Now project is created.

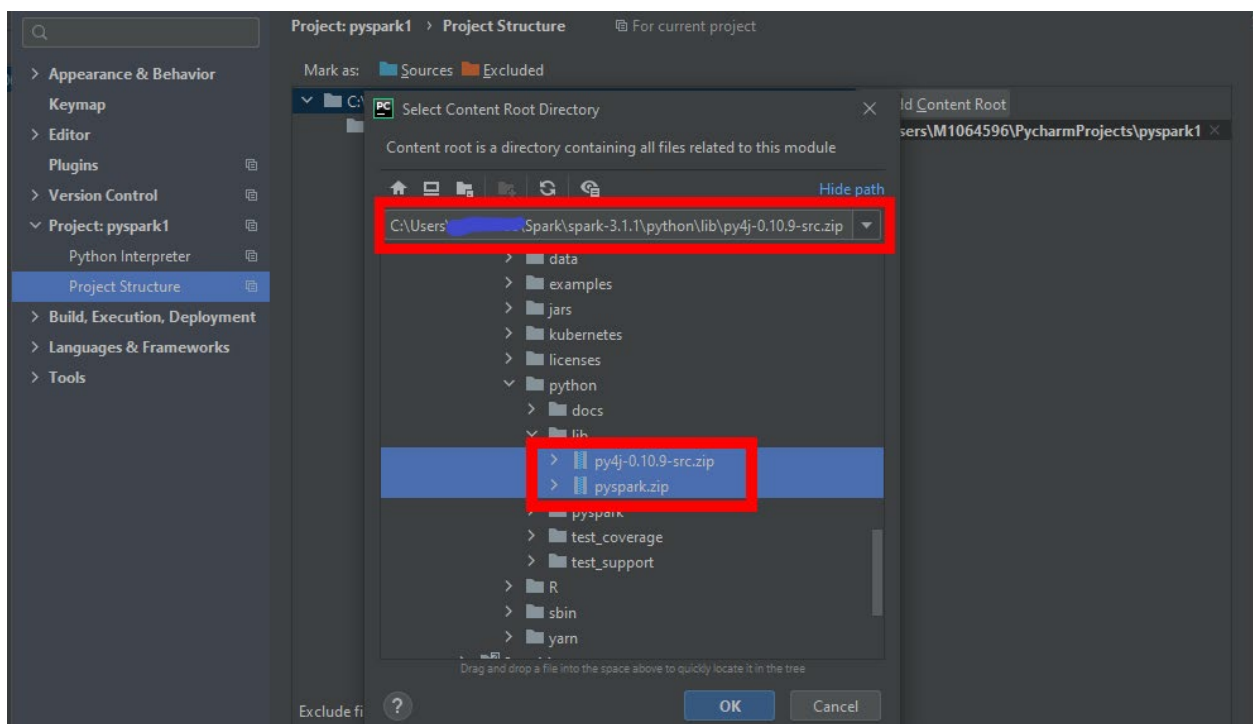
Step 6: Click File > Settings



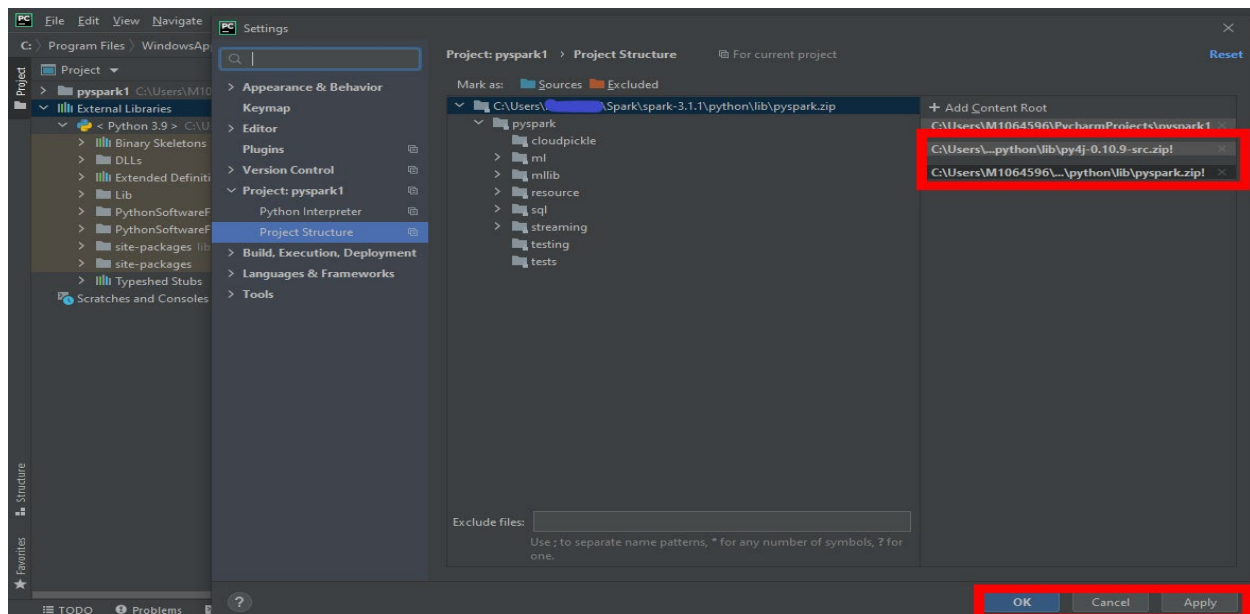
Step 7: Click on project, select Project structure, then click on Add Content Root.



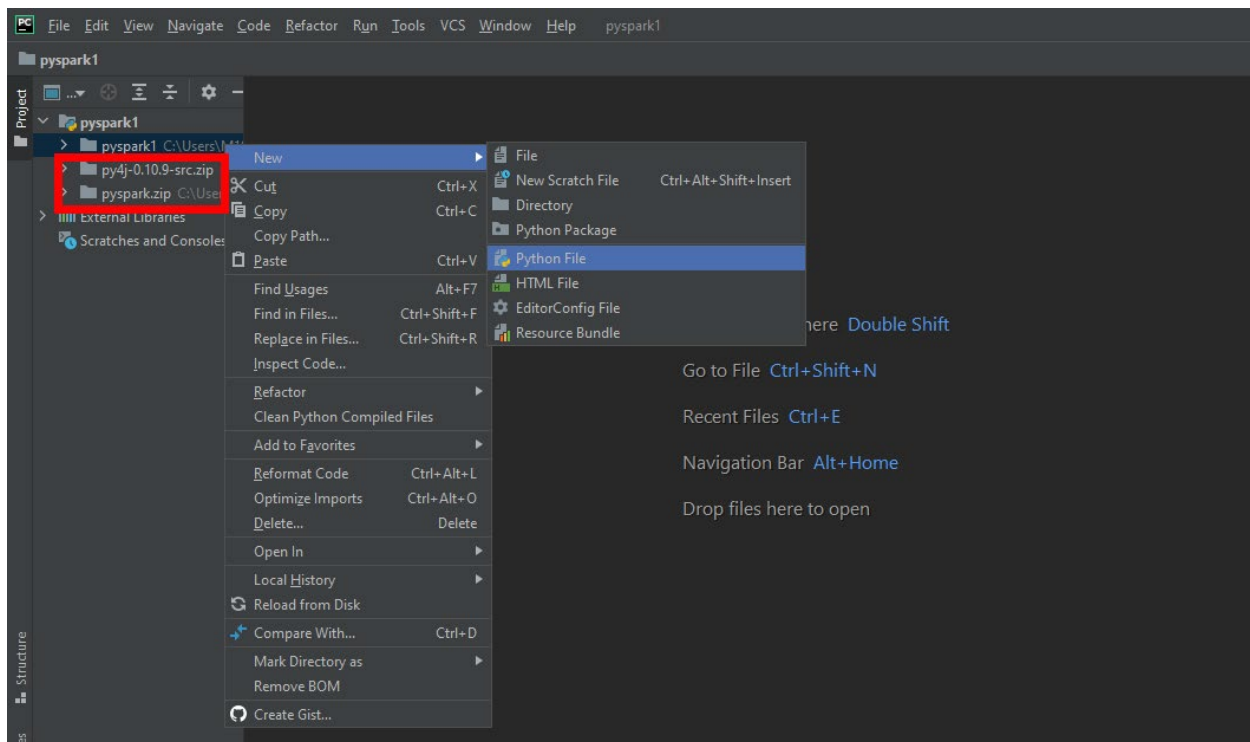
Step 8: Select the Library files py4j-0.10.9-src.zip & pyspark.zip available in the Spark > python > lib directory click on ok.



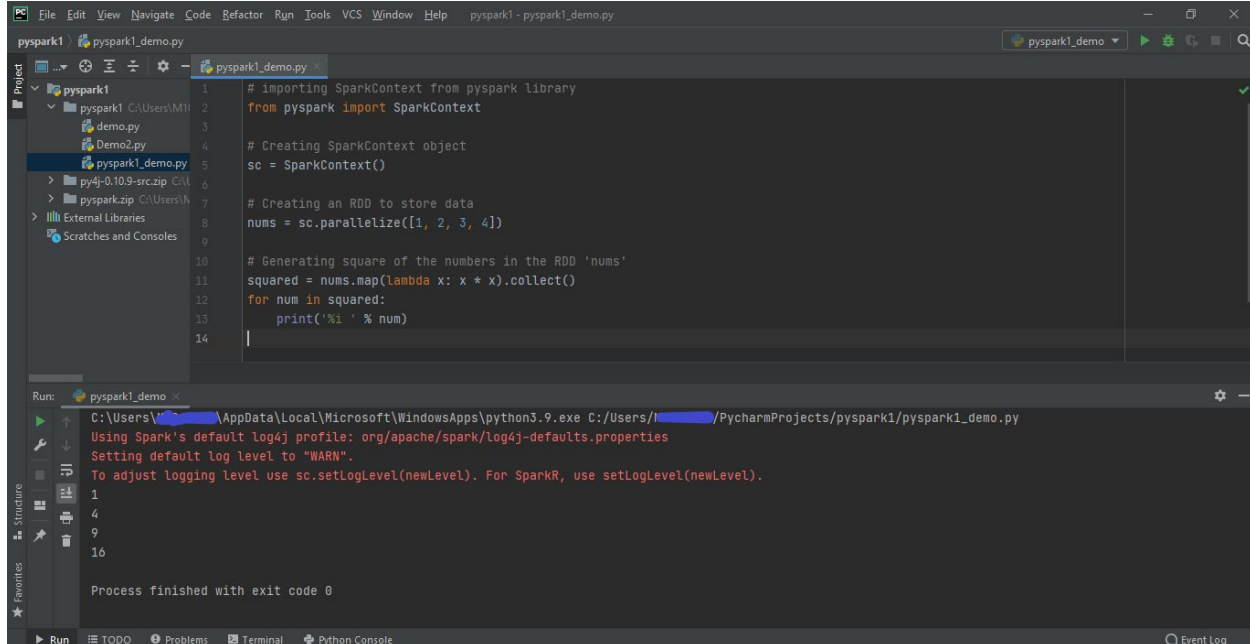
Step 9: Observe the library files root directory added, then click on Apply and then ok.



Step 10: Observe library files added to the project. Then right click on pyspark1 (Project name)> New > Python file.



Step 11: Create a python file with name pyspark1_demo, write pyspark code. Below example shows how to create an RDD with integer values and calculate square of them.



The screenshot shows the PyCharm IDE with a project named 'pyspark1'. A file named 'pyspark1_demo.py' is open in the editor. The code in the file is as follows:

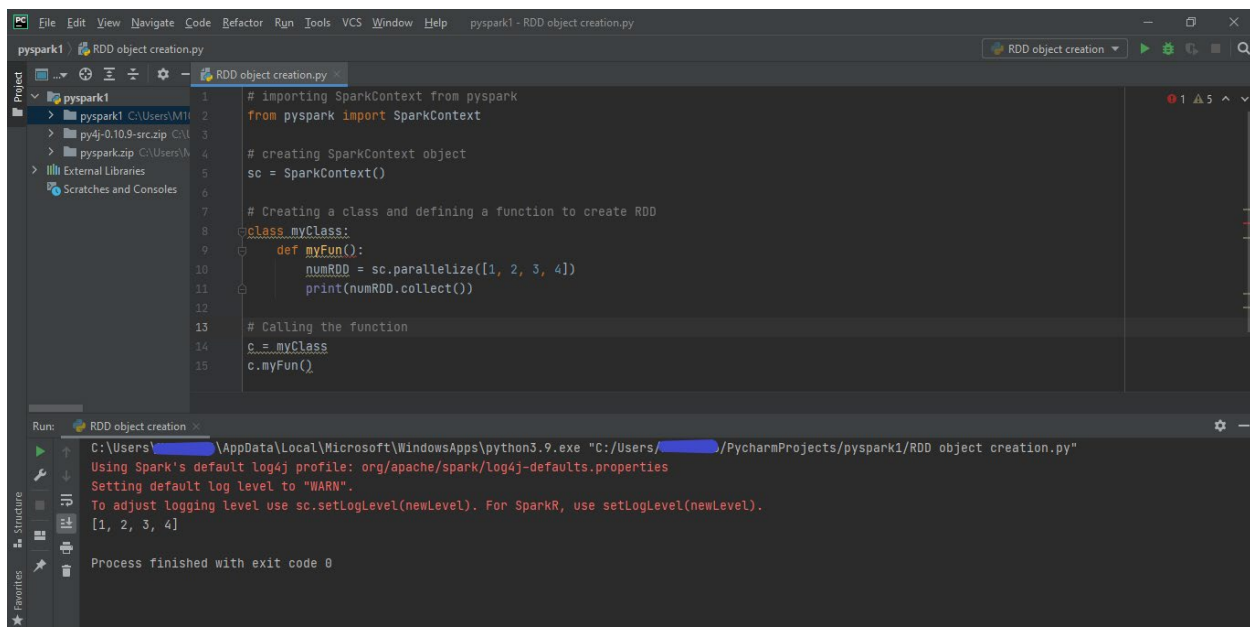
```
1 # Importing SparkContext from pyspark library
2 from pyspark import SparkContext
3
4 # Creating SparkContext object
5 sc = SparkContext()
6
7 # Creating an RDD to store data
8 nums = sc.parallelize([1, 2, 3, 4])
9
10 # Generating square of the numbers in the RDD 'nums'
11 squared = nums.map(lambda x: x * x).collect()
12 for num in squared:
13     print('%i ' % num)
14
```

The Run window at the bottom shows the execution output:

```
C:\Users\W...AppData\Local\Microsoft\WindowsApps\python3.9.exe C:/Users/W.../PycharmProjects/pyspark1/pyspark1_demo.py
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
1
4
9
16
Process finished with exit code 0
```

Creating object/class with oops inline

1. Creating RDD in PySpark using class.



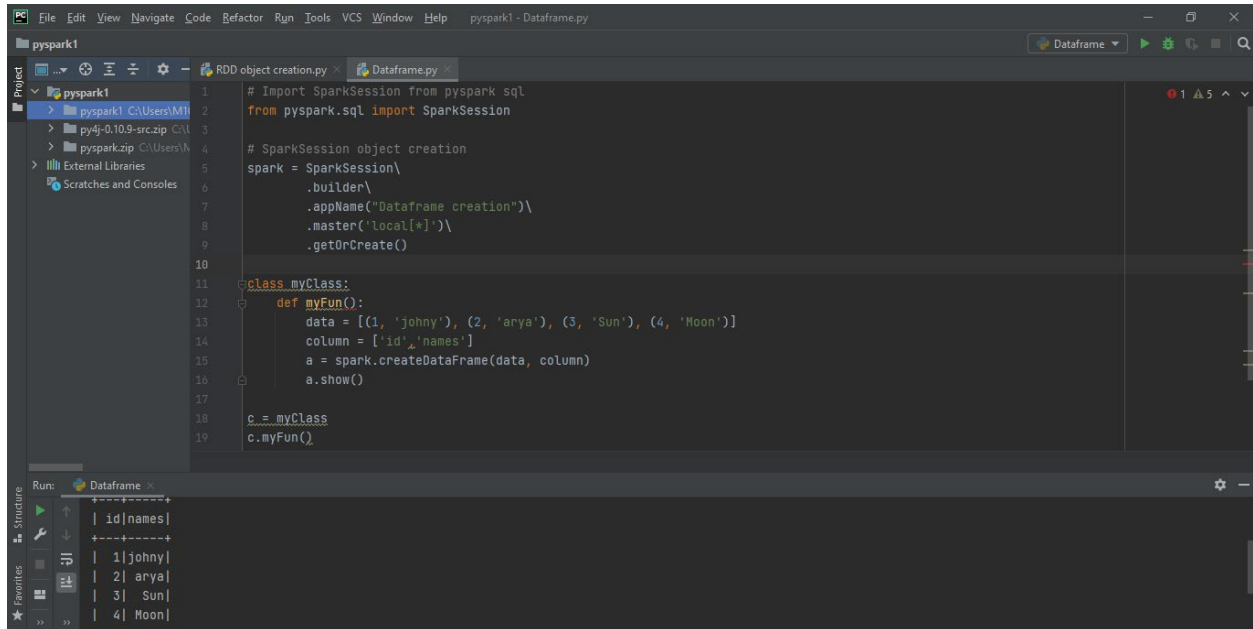
The screenshot shows the PyCharm IDE with a project named 'pyspark1'. A file named 'RDD object creation.py' is open in the editor. The code in the file is as follows:

```
1 # Importing SparkContext from pyspark
2 from pyspark import SparkContext
3
4 # creating SparkContext object
5 sc = SparkContext()
6
7 # Creating a class and defining a function to create RDD
8 class myClass:
9     def myFun():
10         numRDD = sc.parallelize([1, 2, 3, 4])
11         print(numRDD.collect())
12
13 # Calling the function
14 c = myClass
15 c.myFun()
```

The Run window at the bottom shows the execution output:

```
C:\Users\W...AppData\Local\Microsoft\WindowsApps\python3.9.exe "C:/Users/W.../PycharmProjects/pyspark1/RDD object creation.py"
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[1, 2, 3, 4]
Process finished with exit code 0
```

2. Creating Dataframe inside a class

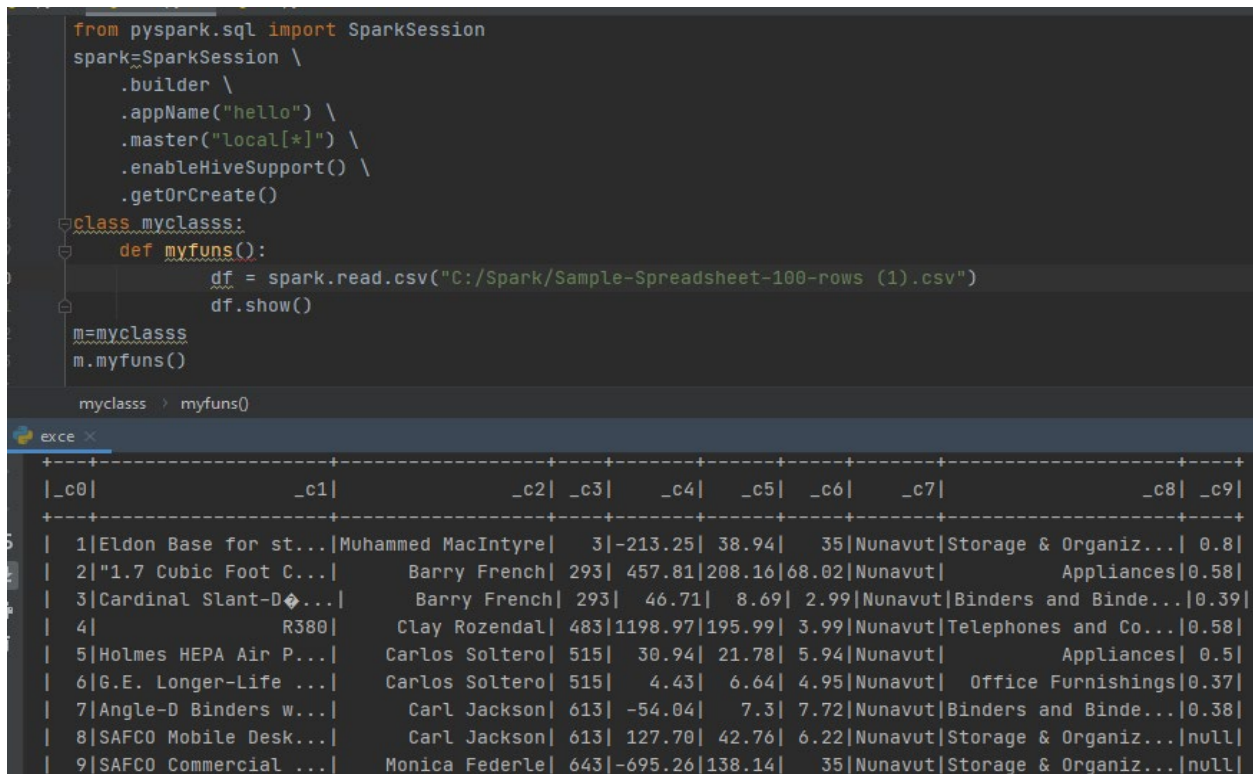


```
1 # Import SparkSession from pyspark sql
2 from pyspark.sql import SparkSession
3
4 # SparkSession object creation
5 spark = SparkSession\
6     .builder\
7     .appName("Dataframe creation")\
8     .master('local[*]')\
9     .getOrCreate()
10
11 class myClass:
12     def myFun():
13         data = [(1, 'johny'), (2, 'arya'), (3, 'Sun'), (4, 'Moon')]
14         column = ['id', 'names']
15         a = spark.createDataFrame(data, column)
16         a.show()
17
18 c = myClass
19 c.myFun()
```

Run: Dataframe

id	names
1	johny
2	arya
3	Sun
4	Moon

3. Reading CSV file in PySpark



```
from pyspark.sql import SparkSession
spark=SparkSession \
    .builder \
    .appName("hello") \
    .master("local[*]") \
    .enableHiveSupport() \
    .getOrCreate()

class myclasss:
    def myfuns():
        df = spark.read.csv("C:/Spark/Sample-Spreadsheet-100-rows (1).csv")
        df.show()

m=myclasss
m.myfuns()
```

myclasss > myfuns()

_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9
1 Eldon Base for st...	Muhammed MacIntyre	3	-213.25	38.94	35	Nunavut	Storage & Organiz...	0.8	
2 "1.7 Cubic Foot C...	Barry French	293	457.81	208.16	68.02	Nunavut	Appliances	0.58	
3 Cardinal Slant-D...	Barry French	293	46.71	8.69	2.99	Nunavut	Binders and Binde...	0.39	
4	R380	483	1198.97	195.99	3.99	Nunavut	Telephones and Co...	0.58	
5 Holmes HEPA Air P...	Carlos Soltero	515	30.94	21.78	5.94	Nunavut	Appliances	0.5	
6 G.E. Longer-Life ...	Carlos Soltero	515	4.43	6.64	4.95	Nunavut	Office Furnishings	0.37	
7 Angle-D Binders w...	Carl Jackson	613	-54.04	7.3	7.72	Nunavut	Binders and Binde...	0.38	
8 SAFCO Mobile Desk...	Carl Jackson	613	127.70	42.76	6.22	Nunavut	Storage & Organiz...	null	
9 SAFCO Commercial ...	Monica Federle	643	-695.26	138.14	35	Nunavut	Storage & Organiz...	null	

Sparkcontext And Sparksession

SparkContext

SparkContext is the entry gate of Apache Spark functionality. The most important step of any Spark driver application is to generate SparkContext.

It provides a way to interact with various spark's functionality with a lesser number of constructs.

In Scala

```
C:\Windows\System32\cmd.exe - spark-shell
Welcome to
  ____  __
 / ___/ /  \
/_  _/ /_  \
 \___/____/
version 3.0.2

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 11.0.11)
Type in expressions to have them evaluated.
Type :help for more information.

scala> 21/05/12 14:18:23 WARN ProcsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of
ProcessTree metrics is stopped
val sc = spark
sc: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@68009f48

scala> val sc = spark.sparkContext.parallelize(Array(1,2,3,4,5))
sc: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:23

scala> sc.collect.foreach(println)
1
2
3
4
5

scala>
```

In python

```
C:\Windows\System32\cmd.exe - pyspark
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/05/12 15:39:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/05/12 15:40:00 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Welcome to
  ____  __
 / ___/ /  \
/_  _/ /_  \
 \___/____/
version 3.0.2

Using Python version 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020 17:59:51)
SparkSession available as 'spark'.
>>> list = ["apple",1], ["mango",2], ["banana",3]
>>> 21/05/12 15:40:12 WARN ProcsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of Pr
ocessTree metrics is stopped

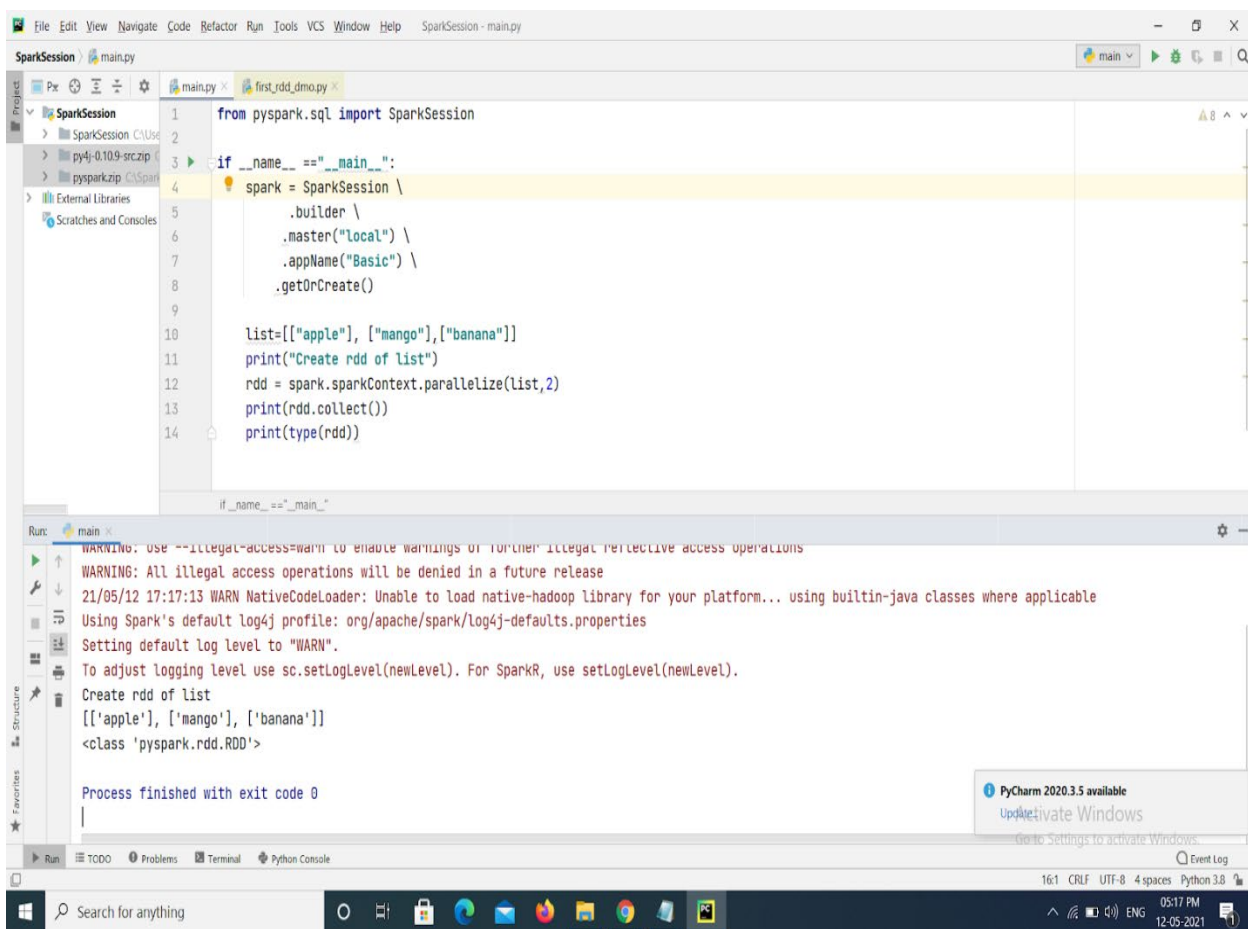
>>> rdd = spark.sparkContext.parallelize(list, 1)
>>> print(rdd.collect())
[['apple', 1], ['mango', 2], ['banana', 3]]
>>>
```

SparkSession

Spark session is a unified entry point of a spark application from Spark 2.0. It provides a way to interact with various spark's functionality with a lesser number of constructs. Instead of having a spark context, hive context, SQL context, now all of it is encapsulated in a Spark session.

Able to Initialize SparkSession: Creating SparkSession using Pycharm IDE.

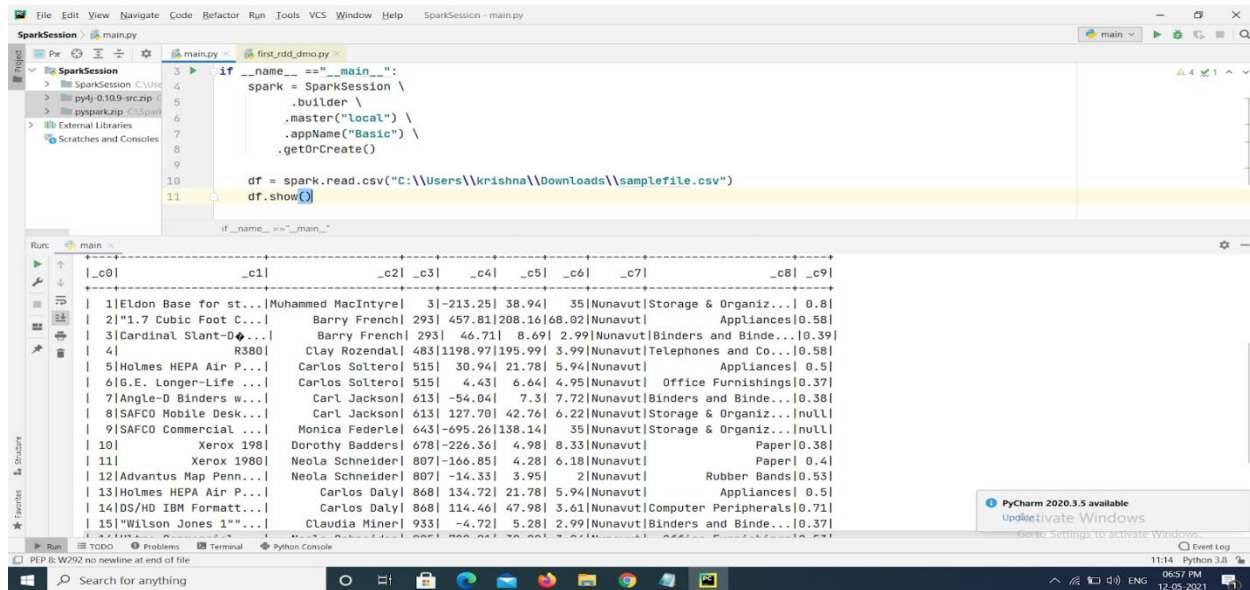
Import sparkSession and write a program for creating sparkSession.



Ability To Ingest Data

Able to Read the input data RDD / Data Frame

1. Read csv file and show using dataframe.



The screenshot shows a PyCharm IDE with a SparkSession project. The main.py file contains the following code:

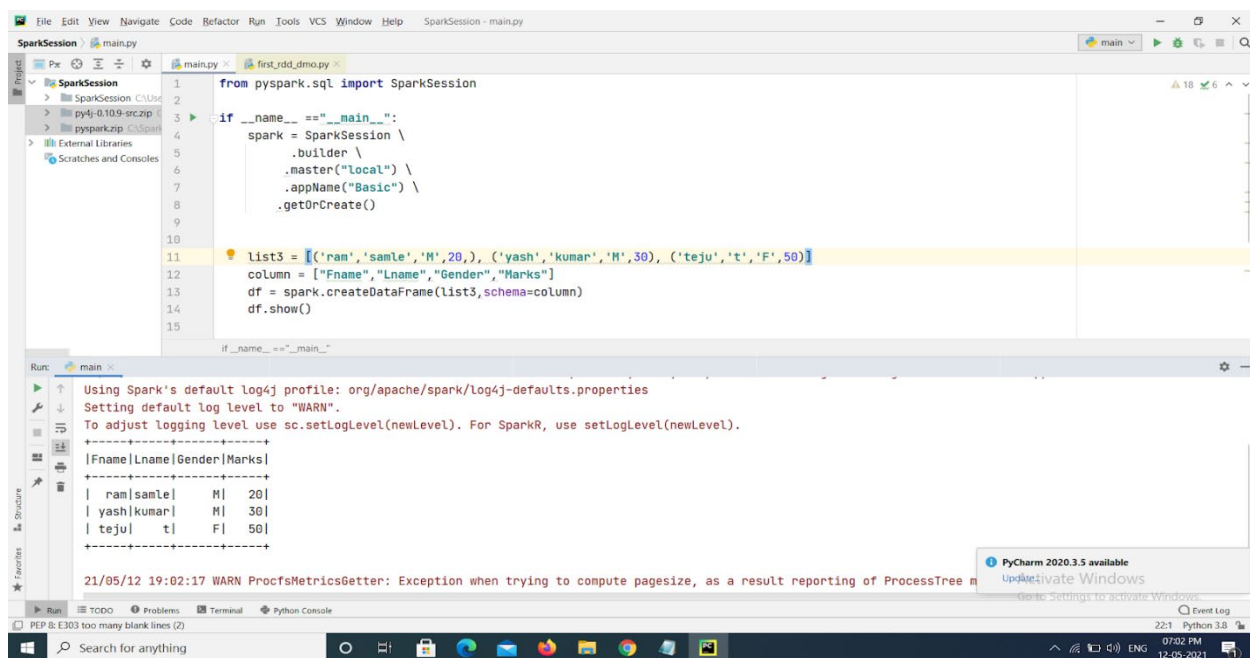
```
if __name__ == "__main__":
    spark = SparkSession \
        .builder \
        .master("local") \
        .appName("Basic") \
        .getOrCreate()

    df = spark.read.csv("C:\\Users\\Krishna\\Downloads\\samplefile.csv")
    df.show()
```

The Run console displays the output of the CSV file, showing a table with 15 rows and 9 columns (_c0 to _c9). The data includes names, IDs, and various numerical values.

_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9
1 Eldon Base for st...	Muhammed MacIntyre	3	-213.25	38.94	35	Nunavut	Storage & Organiz...	0.8	
2 1.7 Cubic Foot C...	Barry French	293	457.81	208.16	68.02	Nunavut	Appliances	0.58	
3 Cardinal Slant-D...	Barry French	293	46.71	8.69	2.99	Nunavut	Binders and Binde...	0.39	
4	Clay Rozendal	483	1198.97	195.99	3.99	Nunavut	Telephones and Co...	0.58	
5 Holmes HEPA Air P...	Carlos Soltero	515	38.94	21.78	5.94	Nunavut	Appliances	0.5	
6 G.E. Longer-Life ...	Carlos Soltero	515	4.43	6.64	4.95	Nunavut	Office Furnishings	0.37	
7 Angle-D Binders w...	Carl Jackson	613	-54.04	7.3	7.72	Nunavut	Binders and Binde...	0.38	
8 SAFCO Mobile Desk...	Carl Jackson	613	127.78	42.76	6.22	Nunavut	Storage & Organiz...	0.37	
9 SAFCO Commercial ...	Monica Federle	643	-695.26	138.14	35	Nunavut	Storage & Organiz...	0.37	
10	Xerox 198	Dorothy Badders	678	-226.36	4.98	8.33	Nunavut	Paper	0.38
11	Xerox 198	Neola Schneider	807	-166.85	4.28	6.18	Nunavut	Paper	0.4
12 Advantus Map Penn...	Neola Schneider	807	-14.33	3.95	2	Nunavut	Rubber Bands	0.53	
13 Holmes HEPA Air P...	Carlos Daly	868	134.72	21.78	5.94	Nunavut	Appliances	0.5	
14 OS/HD IBM Formatt...	Carlos Daly	868	114.46	47.98	3.61	Nunavut	Computer Peripherals	0.71	
15 Wilson Jones 1"	Claudia Miner	933	-4.72	5.28	2.99	Nunavut	Binders and Binde...	0.37	

2. Create dataframe using list



The screenshot shows a PyCharm IDE with a SparkSession project. The main.py file contains the following code:

```
from pyspark.sql import SparkSession

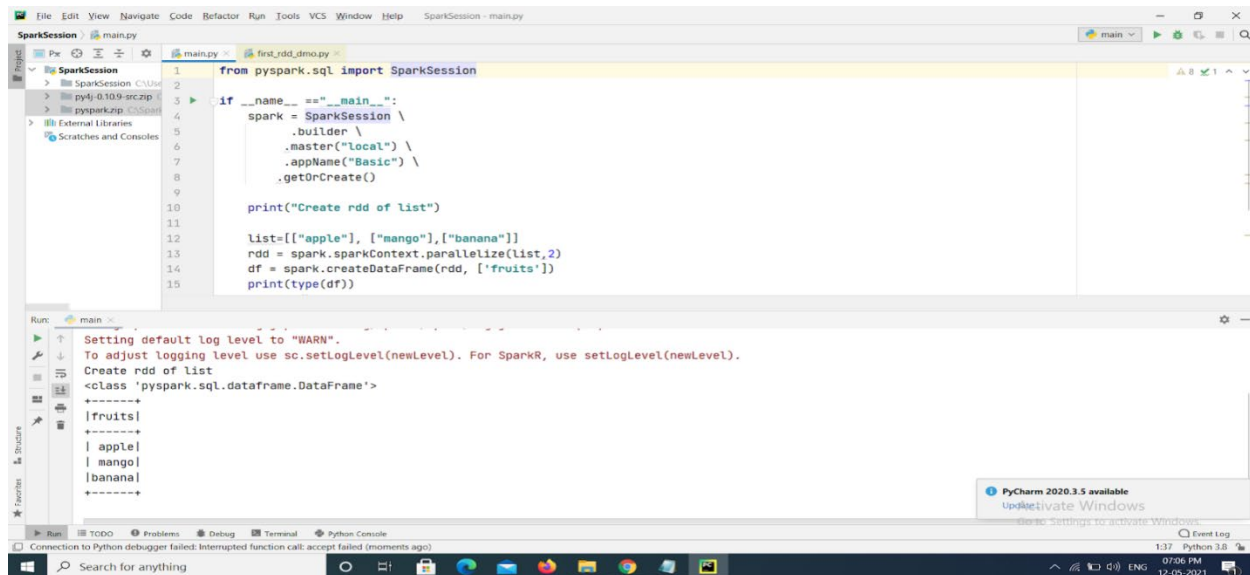
if __name__ == "__main__":
    spark = SparkSession \
        .builder \
        .master("local") \
        .appName("Basic") \
        .getOrCreate()

    list3 = [('ram', 'samle', 'M', 20), ('yash', 'kumar', 'M', 30), ('teju', 't', 'F', 50)]
    column = ["Fname", "Lname", "Gender", "Marks"]
    df = spark.createDataFrame(list3, schema=column)
    df.show()
```

The Run console displays the output of the DataFrame, showing a table with 3 rows and 4 columns (Fname, Lname, Gender, Marks). The data includes names, genders, and marks.

Fname	Lname	Gender	Marks
ram	samle	M	20
yash	kumar	M	30
teju	t	F	50

3. Convert RDD to Dataframe



The image shows a PyCharm IDE window with a Python script for creating a SparkSession and converting an RDD to a DataFrame. The script is as follows:

```
1 from pyspark.sql import SparkSession
2
3 if __name__ == '__main__':
4     spark = SparkSession \
5         .builder \
6         .master("local") \
7         .appName("Basic") \
8         .getOrCreate()
9
10    print("Create rdd of list")
11
12    List=[["apple"], ["mango"], ["banana"]]
13    rdd = spark.sparkContext.parallelize(List,2)
14    df = spark.createDataFrame(rdd, ['fruits'])
15    print(type(df))
```

The Run console shows the output of the script:

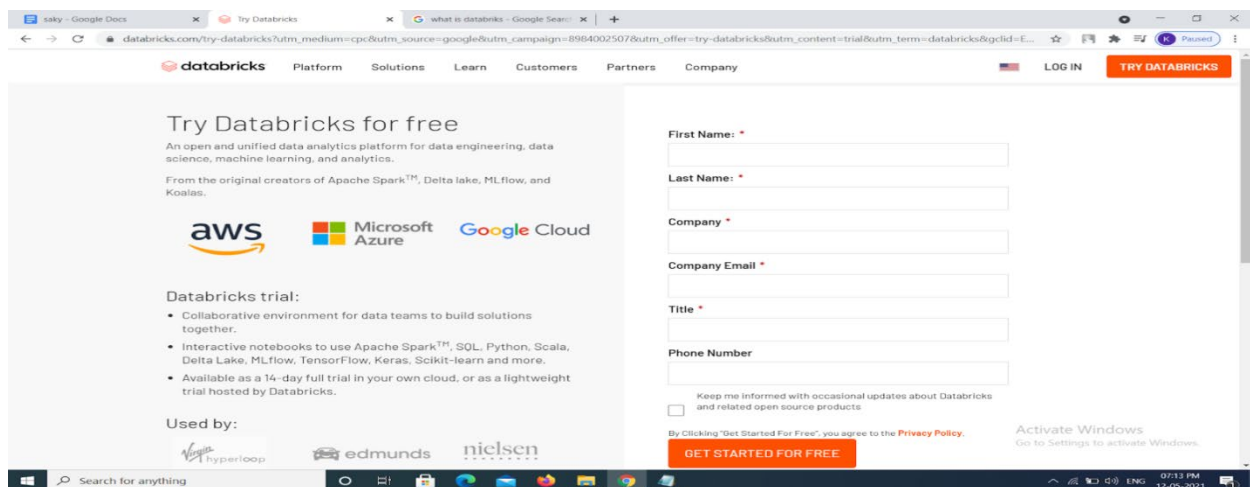
```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Create rdd of list
<class 'pyspark.sql.dataframe.DataFrame'>
+-----+
|fruits|
+-----+
| apple|
| mango|
| banana|
+-----+
```

Databricks Platform

Databricks is an enterprise software company founded by the original creators of Apache Spark. The company has also created Delta Lake, MLflow and Koalas, popular open source projects that span data engineering, data science and machine learning. We can also create databricks account in azure.

Azure Databricks is based on Apache Spark and provides in memory compute with language support for Scala, R, Python and SQL.

Step 1: Create account and login in databricks <https://www.databricks.com/>



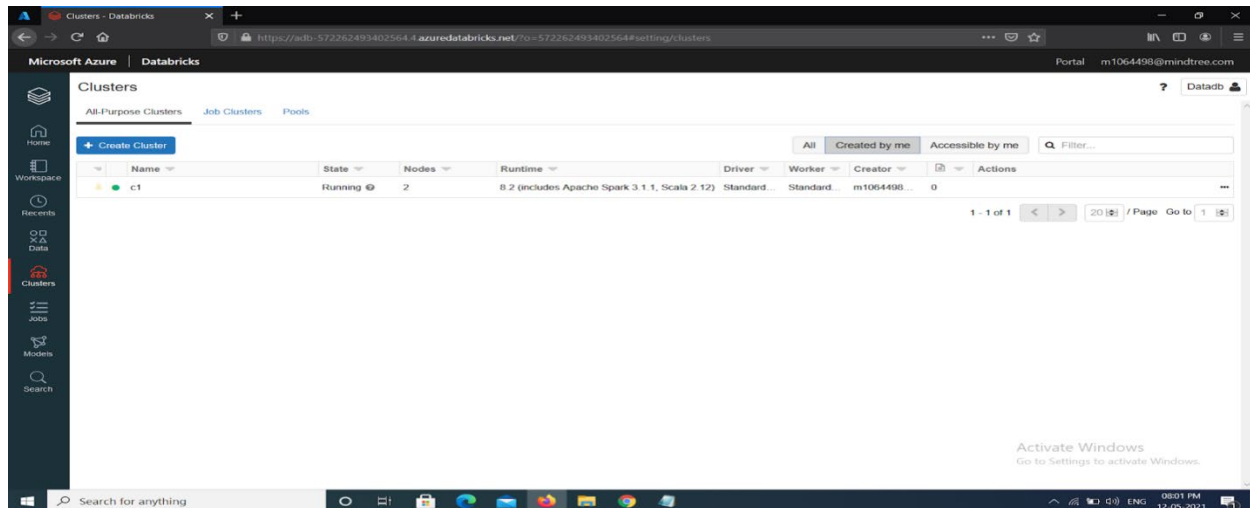
The image shows the Databricks website's 'Try Databricks for free' page. The page features the Databricks logo and navigation links (Platform, Solutions, Learn, Customers, Partners, Company). It includes a 'LOG IN' button and a 'TRY DATABRICKS' button. The main content area describes the platform as an open and unified data analytics platform for data engineering, data science, machine learning, and analytics. It mentions that Databricks is from the original creators of Apache Spark, Delta Lake, MLflow, and Koalas. Below this, there are logos for AWS, Microsoft Azure, and Google Cloud. A 'Databricks trial' section lists the following features:

- Collaborative environment for data teams to build solutions together.
- Interactive notebooks to use Apache Spark™, SQL, Python, Scala, Delta Lake, MLflow, TensorFlow, Keras, Scikit-learn and more.
- Available as a 14-day full trial in your own cloud, or as a lightweight trial hosted by Databricks.

The 'Used by:' section shows logos for Virgin Hyperloop, edmunds, and nielsen. On the right side, there is a form to sign up for the trial, with fields for First Name, Last Name, Company, Company Email, Title, and Phone Number. There is also a checkbox for 'Keep me informed with occasional updates about Databricks and related open source products.' and a 'GET STARTED FOR FREE' button. At the bottom, there is a 'Privacy Policy' link and an 'Activate Windows' watermark.

Step 2: Create clusters and it is use job clusters to run fast and robust automated jobs.

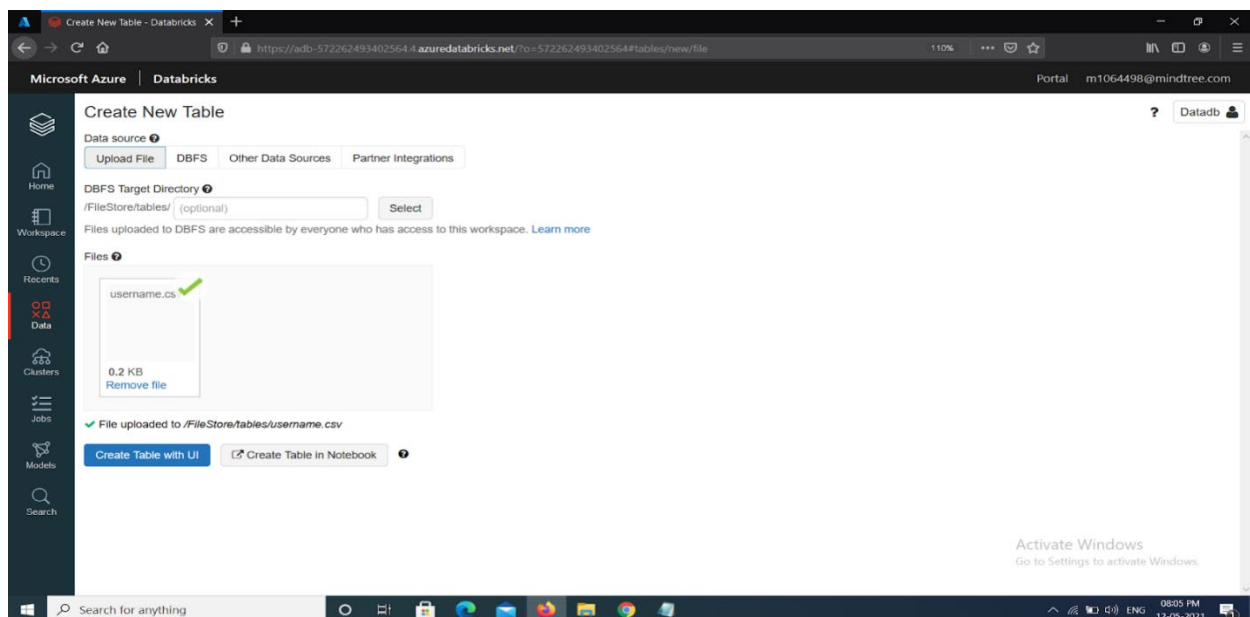
Click Clusters >> Create Cluster >> Cluster Name >> Cluster Mode (Standard) >> Create Cluster



Step 3: Create New Table for upload file in databricks.

Click on Create New Table >> drag and drop file >> Create table with UI >> Select a Cluster to Preview the Table >> preview table >> create table.

If you have small data files on your local machine that you want to analyze with Databricks, you can import them to DBFS using the UI.



Step 4: Open the notebook for getting the automatic generated query from file loaded in the cluster.

```
2 file_location = "/FileStore/tables/sample.csv"
3 file_type = "csv"
4
5 # CSV options
6 infer_schema = "false"
7 first_row_is_header = "false"
8 delimiter = ","
9
10 # The applied options are for CSV files. For other file types, these will be ignored.
11 df = spark.read.format(file_type) \
12     .option("inferSchema", infer_schema) \
13     .option("header", first_row_is_header) \
14     .option("sep", delimiter) \
15     .load(file_location)
16
17 display(df)
```

▶ (2) Spark Jobs

▶  df: pyspark.sql.dataframe.DataFrame = [_c0: string, _c1: string ... 8 more fields]

	_c0	_c1	_c2	_c3	_c4
1	1	Eldon Base for stackable storage shelf, platinum	Muhammed MacIntyre	3	-213.25
2	2	"1.7 Cubic Foot Compact ""Cube"" Office Refrigerators"	Barry French	293	457.81
3	3	Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl	Barry French	293	46.71
4	4	R380	Clay Rozendal	483	1198.97
5	5	Holmes HEPA Air Purifier	Carlos Soltero	515	30.94
6	6	G.E. Longer-Life Indoor Recessed Floodlight Bulbs	Carlos Soltero	515	4.43

Step 5: We can run each cell of the notebook for the result or also create new cell for new query.

```
2
3 temp_table_name = "sample_csv"
4
5 df.createOrReplaceTempView(temp_table_name)
```

nd 4

```
1 %sql
2
3 /* Query the created temp table in a SQL cell */
4
5 select * from `sample_csv`
```

▶ (1) Spark Jobs

	_c0	_c1	_c2	_c3	_c4
1	1	Eldon Base for stackable storage shelf, platinum	Muhammed MacIntyre	3	-213.25
2	2	"1.7 Cubic Foot Compact ""Cube"" Office Refrigerators"	Barry French	293	457.81
3	3	Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl	Barry French	293	46.71
4	4	R380	Clay Rozendal	483	1198.97
5	5	Holmes HEPA Air Purifier	Carlos Soltero	515	30.94
6	6	G.E. Longer-Life Indoor Recessed Floodlight Bulbs	Carlos Soltero	515	4.43
7	7	Analog Binders with Locking Rings, Label Holders	Carl Jackson	613	-54.04

Showing all 100 rows.

Step 6: There are options for getting result of query in graph formats.

```
1 # Create a view or table
2
3 temp_table_name = "sample_csv"
4
5 df.createOrReplaceTempView(temp_table_name)
```



Metadata

Metadata is a data about data or information that describes the data (summary of data). There are few methods that provide metadata .

1. show(): This shows all the data in dataframe.

display data ¶

```
In [2]: df=spark.read.csv("C:/Spark/sample.csv")
```

```
In [10]: df.show()
```

_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9
1 Eldon Base for st...	Muhammed MacIntyre	3	-213.25	38.94	35	Nunavut	Storage & Organiz...	0.8	
2 "1.7 Cubic Foot C...	Barry French	293	457.81	208.16	68.02	Nunavut	Appliances	0.58	
3 Cardinal Slant-D...	Barry French	293	46.71	8.69	2.99	Nunavut	Binders and Binde...	0.39	
4	R380	Clay Rozendal	483	1198.97	195.99	3.99	Nunavut	Telephones and Co...	0.58
5 Holmes HEPA Air P...	Carlos Soltero	515	30.94	21.78	5.94	Nunavut	Appliances	0.5	
6 G.E. Longer-Life ...	Carlos Soltero	515	4.43	6.64	4.95	Nunavut	Office Furnishings	0.37	
7 Angle-D Binders w...	Carl Jackson	613	-54.04	7.3	7.72	Nunavut	Binders and Binde...	0.38	
8 SAFCO Mobile Desk...	Carl Jackson	613	127.70	42.76	6.22	Nunavut	Storage & Organiz...	null	
9 SAFCO Commercial ...	Monica Federle	643	-695.26	138.14	35	Nunavut	Storage & Organiz...	null	
10	Xerox 198	Dorothy Badders	678	-226.36	4.98	8.33	Nunavut	Paper	0.38
11	Xerox 1980	Neola Schneider	807	-166.85	4.28	6.18	Nunavut	Paper	0.4
12 Advantus Map Penn...	Neola Schneider	807	-14.33	3.95	2	Nunavut	Rubber Bands	0.53	
13 Holmes HEPA Air P...	Carlos Daly	868	134.72	21.78	5.94	Nunavut	Appliances	0.5	
14 DS/HD IBM Formatt...	Carlos Daly	868	114.46	47.98	3.61	Nunavut	Computer Peripherals	0.71	
15 "Wilson Jones 1""...	Claudia Miner	933	-4.72	5.28	2.99	Nunavut	Binders and Binde...	0.37	
16 Ultra Commercial ...	Neola Schneider	995	782.91	39.89	3.04	Nunavut	Office Furnishings	0.53	
17 "#10-4 1/8"" x 9 ...	Allen Rosenblatt	998	93.80	15.74	1.39	Nunavut	Envelopes	0.4	
18 Hon 4-Shelf Metal...	Sylvia Foulston	1154	440.72	100.98	26.22	Nunavut	Bookcases	0.6	
19 Lesro Sheffield C...	Sylvia Foulston	1154	-481.04	71.37	69	Nunavut	Tables	0.68	

Using `show(truncate=False)`, the truncation of data can be avoided.

To avoid truncation

```
In [25]: df.show(truncate=False)
```

```
+-----+-----+-----+-----+-----+-----+
|_c0|_c1|_c2|_c3|_c4|_c5|
+-----+-----+-----+-----+-----+-----+
|1|Eldon Base for stackable storage shelf, platinum|Muhammed MacIntyre|3|-213.25|38.9|
|4|35|Nunavut|Storage & Organization|0.8|||
|2|"1.7 Cubic Foot Compact ""Cube"" Office Refrigerators"|Barry French|293|457.81|208.
|16|68.02|Nunavut|Appliances|0.58|||
|3|Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl|Barry French|293|46.71|8.69|
|2.99|Nunavut|Binders and Binder Accessories|0.39|||
|4|R380|||
|99|3.99|Nunavut|Telephones and Communication|0.58|||
|5|Holmes HEPA Air Purifier|Carlos Soltero|515|30.94|21.7|
|8|5.94|Nunavut|Appliances|0.5|||
|6|G.E. Longer-Life Indoor Recessed Floodlight Bulbs|Carlos Soltero|515|4.43|6.64|
|4.95|Nunavut|Office Furnishings|0.37|||
|7|Angle-D Binders with Locking Rings, Label Holders|Carl Jackson|613|-54.04|7.3|
|7.72|Nunavut|Binders and Binder Accessories|0.38|||
|8|SAFCO Mobile Desk Side File, Wire Frame|Carl Jackson|613|127.70|42.7|
|6|6.22|Nunavut|Storage & Organization|null|||
|9|SAFCO Commercial Wire Shelving, Black|Monica Federle|643|-695.26|138.
|14|35|Nunavut|Storage & Organization|null|||
|10|Xerox 198|Dorothy Badders|678|-226.36|4.98|
|8.33|Nunavut|Paper|0.38|||
```

2. columns: This prints the name of columns in dataframe.

Jupyter spark Last Checkpoint: 5 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

Run Code

To print names of columns

```
In [10]: df.columns
```

```
Out[10]:['_c0', '_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9']
```

3. **describe():** This provides statistical insight on the data present in dataframe.

To show statistical info on data

```
In [12]: df.describe().show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
|summary|      _c0|      _c1|      _c2|      _c3|      _c4|      _c5|
|_c6|      _c7|      _c8|      _c9|      _c3|      _c4|      _c5|
+-----+-----+-----+-----+-----+-----+-----+
| count|      100|      100|      100|      100|      100|      100|
100|      100|      100|      97|
| mean|      50.5|      6160.0|      null|      5069.0|      326.58259999999996|      76.13869999999991|      13.875
900000000001|      4.076|      null|      0.5152173913043478|
| stddev|      29.011491975882016|      NaN|      null|      3453.319445329286|      1369.8326563759458|      185.10559463986982|      18.292
605068494733|      1.413552262917788|      null|      0.14419521460159157|
| min|      1|"#10-4 1/8"" x 9 ...|      10/Pack"|      10437|      -0.06|      -144.55|
0.49|      2.27|      Appliances|      0.35|
| max|      99|      g520|      Sylvia Foulston|      Eugene Barchas|      93.80|      95.99|
9.03|      Nunavut|      Telephones and Co...|      Paper|
+-----+-----+-----+-----+-----+-----+-----+
```

4. **dtypes:** This provides the datatypes of the columns.

To print datatype of columns

```
In [12]: df.dtypes
```

```
Out[12]: [('_c0', 'string'),
          ('_c1', 'string'),
          ('_c2', 'string'),
          ('_c3', 'string'),
          ('_c4', 'string'),
          ('_c5', 'string'),
          ('_c6', 'string'),
          ('_c7', 'string'),
          ('_c8', 'string'),
          ('_c9', 'string')]
```

5. printSchema(): This provides the schema of the dataframe in tree format.

To print schema in tree format

```
In [13]: df.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
```

6. schema: This provides the schema of the dataframe in structure format.

To print schema in structure format

```
In [14]: df.schema
```

```
Out[14]: StructType(List(StructField(_c0,StringType,true),StructField(_c1,StringType,true),StructField(_c2,StringType,true),StructField(_c3,StringType,true),StructField(_c4,StringType,true),StructField(_c5,StringType,true),StructField(_c6,StringType,true),StructField(_c7,StringType,true),StructField(_c8,StringType,true),StructField(_c9,StringType,true)))
```

Try And Except

The usage of try and except blocks for reading data is similar to the usage of try and except blocks in python to handle the errors.

Try and Except for reading the data

```
In [25]: from pyspark.sql import SparkSession
spark=SparkSession \
    .builder \
    .appName("hello") \
    .master("local[*]") \
    .enableHiveSupport() \
    .getOrCreate()
class myclasss:
    def myfuncs():
        try:
            df = spark.read.csv("C:/Spark/samples.csv")
            df.show()
        except Exception as error:
            print("There is error in reading dataframe")

m=myclasss
m.myfuncs()
```

There is error in reading dataframe

Try and Except for reading the data

```
In [27]: from pyspark.sql import SparkSession
spark=SparkSession \
    .builder \
    .appName("hello") \
    .master("local[*]") \
    .enableHiveSupport() \
    .getOrCreate()
class myclasss:
    def myfuncs():
        try:
            df = spark.read.csv("C:/Spark/samples.csv")
            df.show()

        except Exception as error:
            print(error)

m=myclasss
m.myfuncs()
```

Path does not exist: file:/C:/Spark/samples.csv;

Similar to python any number of excepts blocks along with finally can be used in pyspark.

Multiple Try and Except for reading the data

```
In [30]: from pyspark.sql import SparkSession
spark=SparkSession \
    .builder \
    .appName("hello") \
    .master("local[*]") \
    .enableHiveSupport() \
    .getOrCreate()
class myclasss:
    def myfuncs():
        try:
            df = spark.read.txt("C:/Spark/sahana.csv")
            df.show()
        except IOError as error:
            print(error)
        except FileNotFoundError as error:
            print(error)
        except Exception as error:
            print(error)
        finally:
            print("pyspark running completed")
if __name__=="__main__":
    m=myclasss
    m.myfuncs()
```

```
'DataFrameReader' object has no attribute 'txt'
pyspark running completed
```


Ability To Apply Cleansing Of Data On The Data Provided

Regular Expression

In pyspark, the regular expression is found in `pyspark.sql.function`. It has two modules

1. `regexp_replace`
2. `regexp_extract`

regexp_extract: extract the characters mentioned in `regex_string` (string that contain regular expression)

syntax:

```
dataframe.select(regexp_extract(col("columnname"),regex_string,0).alias("cleaned")).show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0|      |_c1|      |_c2|_c3|    |_c4|    |_c5|    |_c6|    |_c7|      |_c8|_c9|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1|Eldon Base for st...|Muhammed MacIntyre| 3|-213.25| 38.94| 35|Nunavut|Storage & Organiz...| 0.8|
| 2|"1.7 Cubic Foot C...|Barry French|293| 457.81|208.16|68.02|Nunavut|Appliances|0.58|
| 3|Cardinal Slant-D...|Barry French|293| 46.71| 8.69| 2.99|Nunavut|Binders and Binde...|0.39|
| 4|      R380|Clay Rozenda|483|1198.97|195.99| 3.99|Nunavut|Telephones and Co...|0.58|
| 5|Holmes HEPA Air P...|Carlos Soltero|515| 30.94| 21.78| 5.94|Nunavut|Appliances| 0.5|
+-----+-----+-----+-----+-----+-----+-----+-----+
```

only showing top 5 rows

Regular exp for cleaning data by fetching only integer part in column _c4

```
In [12]: from pyspark.sql.functions import regexp_extract
from pyspark.sql.functions import col
regex_string="\\d+"
dataframe.select("_c4",regexp_extract(col("_c4"),regex_string,0).alias ("cleaned _c4")).show(5)
```

```
+-----+-----+
|      |_c4|cleaned _c4|
+-----+-----+
|-213.25|      213|
| 457.81|      457|
|  46.71|       46|
|1198.97|     1198|
|  30.94|       30|
+-----+-----+
```

regexp_replace: find the matching regex_string (string that contain regular expression) and replaces with the other given value.

syntax:

```
dataframe.select(regexp_replace(col("columnname"),regex_string,"").alias("cleaned")).show()
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|_c9|
+-----+-----+-----+-----+-----+-----+-----+-----+
|1|Eldon Base for st...|Muhammed MacIntyre|3|-213.25|38.94|35|Nunavut|Storage & Organiz...|0.8|
|2|"1.7 Cubic Foot C...|Barry French|293|457.81|208.16|68.02|Nunavut|Appliances|0.58|
|3|Cardinal Slant-D...|Barry French|293|46.71|8.69|2.99|Nunavut|Binders and Binde...|0.39|
|4|R380|Clay Rozendal|483|1198.97|195.99|3.99|Nunavut|Telephones and Co...|0.58|
|5|Holmes HEPA Air P...|Carlos Soltero|515|30.94|21.78|5.94|Nunavut|Appliances|0.5|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

Regular exp for cleaning data by replacing with new value in column _c2

```
In [21]: from pyspark.sql.functions import regexp_replace
from pyspark.sql.functions import col
regex_string="Barry"
dataframe.select("_c2",regexp_replace(col("_c2"),regex_string,"Arm").alias ("cleaned")).show(5)
```

```

+-----+-----+
|_c2|cleaned|
+-----+-----+
|Muhammed MacIntyre|Muhammed MacIntyre|
|Barry French|Arm French|
|Barry French|Arm French|
|Clay Rozendal|Clay Rozendal|
|Carlos Soltero|Carlos Soltero|
+-----+-----+
only showing top 5 rows
```

The cleaned column can also be added as a column to the existing dataframe using withColumn().

Reg Exp cleaned data adding in dataframe

```
In [23]: dataframe=spark.read.csv("C:/Spark/sample.csv")
dataframe.show(5)
dataframe=dataframe.withColumn("cleaned_c4",regexp_replace(col("_c4"),"\\-", ""))
dataframe.show(5)
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|_c9|
+-----+-----+-----+-----+-----+-----+-----+-----+
|1|Eldon Base for st...|Muhammed MacIntyre|3|-213.25|38.94|35|Nunavut|Storage & Organiz...|0.8|
|2|"1.7 Cubic Foot C...|Barry French|293|457.81|208.16|68.02|Nunavut|Appliances|0.58|
|3|Cardinal Slant-D...|Barry French|293|46.71|8.69|2.99|Nunavut|Binders and Binde...|0.39|
|4|R380|Clay Rozendal|483|1198.97|195.99|3.99|Nunavut|Telephones and Co...|0.58|
|5|Holmes HEPA Air P...|Carlos Soltero|515|30.94|21.78|5.94|Nunavut|Appliances|0.5|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|_c9|cleaned_c4|
+-----+-----+-----+-----+-----+-----+-----+-----+
|1|Eldon Base for st...|Muhammed MacIntyre|3|-213.25|38.94|35|Nunavut|Storage & Organiz...|0.8|213.25|
|2|"1.7 Cubic Foot C...|Barry French|293|457.81|208.16|68.02|Nunavut|Appliances|0.58|457.81|
|3|Cardinal Slant-D...|Barry French|293|46.71|8.69|2.99|Nunavut|Binders and Binde...|0.39|46.71|
|4|R380|Clay Rozendal|483|1198.97|195.99|3.99|Nunavut|Telephones and Co...|0.58|1198.97|
|5|Holmes HEPA Air P...|Carlos Soltero|515|30.94|21.78|5.94|Nunavut|Appliances|0.5|30.94|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

Union To Merge All The Input Data

PySpark UNION is a transformation in PySpark that is used to merge two or more data frames in a PySpark application.

The union operation is applied to spark data frames with the same schema and structure. This is a very important condition for the union operation to be performed in any PySpark application.

The union operation can be carried out with two or more PySpark data frames and can be used to combine the data frame to get the defined result.

It returns a new Spark Data Frame that contains the union of rows of the data frames used.

The syntax for the PYSPARK UNION function is:

```
Df = df1.union(df2)
```

where,

Df = DataFrame post union.

Df1 = DataFrame1 used for union operation.

Df2 = DataFrame2 used for union operation.

.union :- The union transformation

The Union is a transformation in Spark that is used to work with multiple data frames in Spark. It takes the data frame as the input and the return type is a new data frame containing the elements that are in data frame1 as well as in data frame2.

This transformation takes out all the elements whether its duplicate or not and appends them making them into a single data frame for further operational purposes.

We can also apply the union operation to more than one data frame in a spark application.

PySpark UNION is a transformation in PySpark that is used to merge two or more data frames in a PySpark application.

The union operation is applied to spark data frames with the same schema and structure.

This is a very important condition for the union operation to be performed in any PySpark application.

The union operation can be carried out with two or more PySpark data frames and can be used to combine the data frame to get the defined result.

It returns a new Spark Data Frame that contains the union of rows of the data frames used.

The syntax for the PYSPARK UNION function is:

```
Df = df1.union(df2)
```

where,

Df = DataFrame post union.

Df1 = DataFrame1 used for union operation.

Df2 = DataFrame2 used for union operation.

.union :- The union transformation

The Union is a transformation in Spark that is used to work with multiple data frames in Spark. It takes the data frame as the input and the return type is a new data frame containing the elements that are in data frame1 as well as in data frame2.

This transformation takes out all the elements whether its duplicate or not and appends them making them into a single data frame for further operational purposes.

We can also apply the union operation to more than one data frame in a spark application.

```
import SparkContext, SparkConf, SparkSession from pyspark
```

```
In [1]: > import findspark
findspark.init()
```

```
In [2]: > findspark.find()
import pyspark
findspark.find()
```

```
Out[2]: 'C:\\bigData\\spark'
```

```
In [3]: > from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
```

```
In [4]: > conf=pyspark.SparkConf().setAppName('appName').setMaster('local')
sc=pyspark.SparkContext.getOrCreate(conf=conf)
spark=SparkSession(sc)
```

1. To merge two data frame having having same schema and structure. The output will append both the data frames together and the result will have both the data Frames together.

```
In [7]: M a= spark.createDataFrame(["SAM", "JOHN", "AND", "ROBIN", "ANAND"], "string").toDF("Name")
```

```
In [8]: M b= spark.createDataFrame(["DAN", "JACK", "AND"], "string").toDF("Name")
```

```
In [9]: M c = a.union(b).show()
```

```
+-----+
| Name |
+-----+
| SAM  |
| JOHN |
| AND  |
| ROBIN|
| ANAND|
| DAN  |
| JACK |
| AND  |
+-----+
```

2. To perform multiple union operations over the PySpark Data Frame. The same union operation can be applied to all the data frames to merge more than two data frame having same schema and structure.

```
In [15]: M c= spark.createDataFrame(["DATN", "JACKj", "AND"], "string").toDF("Name")
```

```
In [16]: M d=a.union(b).union(c)
```

```
In [18]: M d.show()
```

```
+-----+
| Name |
+-----+
| SAM  |
| JOHN |
| AND  |
| ROBIN|
| ANAND|
| DAN  |
| JACK |
| AND  |
| DATN |
| JACKj|
| AND  |
+-----+
```

3. The same union operation can be done with Data Frame with Integer type as the data type also.

```
In [20]: c = spark.createDataFrame([23,98,67], "integer").toDF("Number")
b = spark.createDataFrame([22,12,12], "integer").toDF("Number1")
d = c.union(b)
d.show()
```

```
+-----+
|Number|
+-----+
| 23|
| 98|
| 67|
| 22|
| 12|
| 12|
+-----+
```

```
In [21]: d.distinct().show()
```

```
+-----+
|Number|
+-----+
| 12|
| 22|
| 23|
| 98|
| 67|
+-----+
```

4. Another way to merge more than two dataframe having the same schema and structure is by unionAll() function in combination with the reduce() function from the funtools module.

```
In [5]: # import modules
from functools import reduce
```

```
In [6]: from pyspark.sql import DataFrame
```

```
In [8]: df1 = sc.parallelize([[1., 'age 18-25']])
df2 = sc.parallelize([[2., 'age 26-30']])
df3 = sc.parallelize([[3., 'age 31-35']])
```

```
In [9]: df1=df1.toDF(["f1","age"])
df1.show()
```

```
+---+-----+
| f1|    age|
+---+-----+
|1.0|age 18-25|
+---+-----+
```

```
In [10]: df2=df2.toDF(["f1", "age"])
df2.show()
```

```
+---+-----+
| f1|    age|
+---+-----+
|2.0|age 26-30|
+---+-----+
```

```
In [11]: df3=df3.toDF(["f1", "age"])
df3.show()
```

```
+---+-----+
| f1|    age|
+---+-----+
|3.0|age 31-35|
+---+-----+
```

5. `reduce()` takes two arguments, a function and the input arguments for the function. Instead of two input arguments, we can provide a list.

In this case, `reduce` will apply the function subsequently to the list.

```
In [12]: M dfs = [df1, df2, df3]
```

```
In [13]: M df_complete = reduce(DataFrame.unionAll, dfs)
```

```
In [14]: M df_complete.collect()
```

```
Out[14]: [Row(f1=1.0, age='age 18-25'),
          Row(f1=2.0, age='age 26-30'),
          Row(f1=3.0, age='age 31-35')]
```

There are 3 ways to merge two dataframe having different schema.

- 1.withColumn,Union
- 2.Outer Join
- 3.automate the process

1. withColumn,Union

`import lit` from `pyspark.sql.functions`. Dataframe with missing column has to be filled with `null`. After having two dataframe with same schema use `union` function.

withColumn,Union

```
In [48]: M from pyspark.sql.functions import lit
```

```
In [49]: M df1 = sc.parallelize([[1., 'age 18-25', 'mite']])
          M df2 = sc.parallelize([[3., 'age 31-35']])
```

```
In [51]: M df1=df1.toDF(["f1","age","college"])
          M df1.show()
```

```
+---+-----+
| f1|    age|college|
+---+-----+
|1.0|age 18-25| mite|
+---+-----+
```

```
In [53]: M df2=df2.toDF(["f1", "age"])
          M df2.show()
```

```
+---+-----+
| f1|    age|
+---+-----+
|3.0|age 31-35|
+---+-----+
```

```
In [57]: M df3=df2.withColumn("college",lit("null")).show()
```

```
+---+-----+
| f1|    age|college|
+---+-----+
|3.0|age 31-35|  null|
+---+-----+
```

```
In [56]: M df3.union(df1).show()
```

```
+---+-----+
| f1|    age|college|
+---+-----+
|3.0|age 31-35|  null|
|1.0|age 18-25| mite|
+---+-----+
```


2. Outer Join

By using the outer join it will automatically fill null values to the missing column.

Outer Join

```
In [97]: df_output=df1.join(df2,on=["f1","age"],how="Outer")
```

```
In [98]: df_output.show()
```

```
+---+-----+-----+
| f1|    age|college|
+---+-----+-----+
|3.0|age 31-35|  null|
|1.0|age 18-25| mite|
+---+-----+-----+
```

3. Automate the process

This is the best way to merge two dataframe having different schema.

It will take set of columns and makes a list of missing columns and each missing column will be filled with null values. After having same schema union function is applied on both dataframe.

automate the process

```
In [100]: listA=list(set(df1.columns)-set(df2.columns))
listB=list(set(df2.columns)-set(df1.columns))
```

```
In [102]: for i in listA:
df2=df2.withColumn(i,lit("null"))

for i in listB:
df1=df1.withColumn(i,lit("null"))
```

```
In [104]: df1.union(df2).show()
```

```
+---+-----+-----+
| f1|    age|college|
+---+-----+-----+
|1.0|age 18-25| mite|
|3.0|age 31-35|  null|
+---+-----+-----+
```

Print The Result In Console And Databrick Platform

1. Printing the result in console

To open scala in command prompt, we use

>>>spark-shell

In scala we use scala programming language to create variable, where variable is prefixed with val keyword. We can perform the same operation with the same command in console also that we have used in jupyter.

```
scala> val x=sc.parallelize(Array(1,2,3))
x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> val y=sc.parallelize(Array(6,3,5))
y: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at <console>:24

scala> val z=x.union(y)
z: org.apache.spark.rdd.RDD[Int] = UnionRDD[2] at union at <console>:27

scala> z.collect.foreach(println)
1
2
3
6
3
5

scala> val p=z.distinct()
p: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[5] at distinct at <console>:25

scala> p.collect.foreach(println)
1
2
3
5
6
```

2. Print the result in databrick platform

In databrick platform create a cluster, then in the notebook create a dataframe then perform the action on that dataframe.

```
1 df1 = sc.parallelize([[1., 'age 18-25','mite']])
2 df2 = sc.parallelize([[2., 'age 26-30',82]])
3 df3 = sc.parallelize([[3., 'age 31-35']])
```

Command took 0.21 seconds -- by amithahegde98@gmail.com at 5/14/2021, 10:24:32 AM on demo

Cmd 3

```
1 df1=df1.toDF(["f1","age","college"])
2 df1.show()
```

▶ (6) Spark Jobs

▶ df1: pyspark.sql.dataframe.DataFrame = [f1: double, age: string ... 1 more fields]

```
+---+-----+-----+
| f1|      age|college|
+---+-----+-----+
|1.0|age 18-25|  mite|
+---+-----+-----+
```

...

```
1 df2=df2.toDF(["f1", "age","marks"])
2 df2.show()
```

▶ (6) Spark Jobs

▶ df2: pyspark.sql.dataframe.DataFrame = [f1: double, age: string ... 1 more fields]

```
+---+-----+-----+
| f1|      age|marks|
+---+-----+-----+
|2.0|age 26-30|   82|
+---+-----+-----+
```

Command took 1.15 seconds -- by amithahegde98@gmail.com at 4/22/2021, 1:53:56 PM on spark_demo

d 5

```
1 df3=df3.toDF(["f1", "age"])
2 df3.show()
```

▶ (6) Spark Jobs

▶ df3: pyspark.sql.dataframe.DataFrame = [f1: double, age: string]

```
+---+-----+
| f1|      age|
+---+-----+
|3.0|age 31-35|
+---+-----+
```

Here in the following screenshot the union function on two dataframes is shown. The databricks not only displays output on the console but it also provides a feature that help in visualizing the result of the query in the graphical manner.

```
1 #from pyspark.sql.functions import lit
2 df3_add=df3.withColumn("college",lit("null"))
```

▶ df3_add: pyspark.sql.dataframe.DataFrame = [f1: double, age: string ... 1 more fields]

Command took 0.10 seconds -- by amithahegde98@gmail.com at 4/22/2021, 1:54:40 PM on spark_demo

Cmd 7

```
1 df3_add.union(df1).show()
```

▶ (3) Spark Jobs

```
+---+-----+-----+
| f1|      age|college|
+---+-----+-----+
|3.0|age 31-35|  null|
|1.0|age 18-25|  mite|
+---+-----+-----+
```

Ability to Aggregate and Summarizing Data into Useful Reports

Aggregate Function

Pyspark provides built-in standard Aggregate functions defines in Dataframe API, these comes in handy when we need to make aggregate operations on Dataframe columns.

Aggregate functions operate on a group of rows and calculate a single return value for every group. Some of the aggregate functions are avg, count, max, mean, min, sum.

create a data frame by taking input from csv file

```
In [1]: from pyspark.sql import SparkSession

In [2]: spark = SparkSession.builder.appName("groupbyagg").getOrCreate()

In [29]: df = spark.read.csv(r'C:\Users\Lenovo\Downloads\sample1.csv', inferSchema=True, header=True)

In [31]: #Showing the data
df.show()
```

Phone	User Name	First Name	Last Name	Display Name	Job Title	Department	Office Number	Office Phone	Mobile
-6641 123-555-9821 1	chris@contoso.com	Chris	Green	Chris Green	IT Manager	Information Techn...	123451	123-555-1211	123-555
-6642 123-555-9822 1	ben@contoso.com	Ben	Andrews	Ben Andrews	IT Manager	Information Techn...	123452	123-555-1212	123-555
-6643 123-555-9823 1	david@contoso.com	David	Longmuir	David Longmuir	IT Manager	Information Techn...	123453	123-555-1213	123-555
-6644 123-555-9824 1	cynthia@contoso.com	Cynthia	Carey	Cynthia Carey	IT Manager	Information Techn...	123454	123-555-1214	123-555
-6645 123-555-9825 1	melissa@contoso.com	Melissa	MacBeth	Melissa MacBeth	IT Manager	Information Techn...	123455	123-555-1215	123-555

Usage of group by function on a dataframe and apply aggregate function on it.

Syntax for aggregate function is:-

Dataframe.agg({'column': 'aggregate function'})

aggregate function

agg: avg, count, max, mean, min, sum

```
In [43]: # Sum by Agg
group_data = df.groupBy("City")
group_data.agg({'salary': 'sum'}).show()
```

City	sum(salary)
Redmond	157000

```
In [44]: # Max by agg
group_data.agg({'salary': 'max'}).show()
```

City	max(salary)
Redmond	60000

Usage of aggregate function directly on dataframe.

```
In [45]: ▶ # Sum  
df.agg({'salary': 'sum'}).show()
```

```
+-----+  
|sum(salary)|  
+-----+  
|      157000|  
+-----+
```

```
In [46]: ▶ df.agg({'salary': 'max'}).show()
```

```
+-----+  
|max(salary)|  
+-----+  
|      60000|  
+-----+
```

```
In [47]: ▶ df.agg({'salary': 'avg'}).show()
```

```
+-----+  
|avg(salary)|  
+-----+  
|      31400.0|  
+-----+
```