Spark Data Processing

**Creating a PySpark DataFrame**

**PySpark** helps in processing large datasets using its DataFrame structure. In this article, we will see different methods to create a [PySpark](https://www.geeksforgeeks.org/introduction-pyspark-distributed-computing-apache-spark/" \t "_blank) DataFrame. It starts with initialization of SparkSession which serves as the entry point for all PySpark applications which is shown below:

*from pyspark.sql import SparkSession  
spark = SparkSession.builder.getOrCreate()*

Lets see an example of creating DataFrame from a List of Rows. Here we can create a DataFrame from a list of rows where each row is represented as a Row object. This method is useful for small datasets that can fit into memory.

* **spark = SparkSession.builder.getOrCreate()**: Initializes a SparkSession which is the entry point for working with PySpark or retrieves an existing session if one is already created.
* **df = spark.createDataFrame([…])**: Creates a PySpark DataFrame using a list of Row objects where each row contains values for the columns a, b, c, d and e.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

df = spark.createDataFrame([

Row(a=1, b=4., c='GFG1', d=date(2000, 8, 1),

e=datetime(2000, 8, 1, 12, 0)),

Row(a=2, b=8., c='GFG2', d=date(2000, 6, 2),

e=datetime(2000, 6, 2, 12, 0)),

Row(a=4, b=5., c='GFG3', d=date(2000, 5, 3),

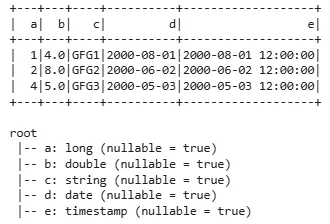
e=datetime(2000, 5, 3, 12, 0))

])

df.show()

df.printSchema()

**Output:**



*Basic example using List of rows*

**Syntax**

*pyspark.sql.SparkSession.createDataFrame(data, schema=None, samplingRatio=None, verifySchema=True)*

**Parameters:**

* **data**: Data we want to load into the DataFrame.
* **schema**: A string or list specifying column names and data types. It is optional.
* **samplingRatio**: Ratio of rows used for analysing schema. Default is None.
* **verifySchema**: Ensures data types of each row match the schema. Default is True.

**Returns:**Dataframe

**Different Methods to Create a PySpark DataFrame**

**1. Create PySpark DataFrame with an Explicit Schema**

Here we can specify the schema explicitly to define the structure of DataFrame which is useful when we want more control over data types.

* **df = spark.createDataFrame([…], schema=’a long, b double, c string, d date, e timestamp’)**: Creates a PySpark DataFrame using a list of tuples and an explicit schema that defines the column names and data types.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

df = spark.createDataFrame([

(1, 4., 'GFG1', date(2000, 8, 1),

datetime(2000, 8, 1, 12, 0)),

(2, 8., 'GFG2', date(2000, 6, 2),

datetime(2000, 6, 2, 12, 0)),

(3, 5., 'GFG3', date(2000, 5, 3),

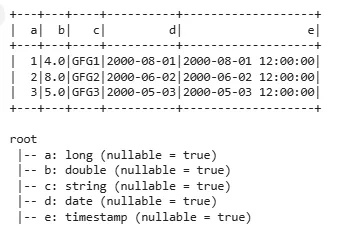
datetime(2000, 5, 3, 12, 0))

], schema='a long, b double, c string, d date, e timestamp')

df.show()

df.printSchema()

**Output:**



*Explicit Schema*

**2. Create DataFrame from a Pandas DataFrame**

We can convert a Pandas DataFrame into a PySpark DataFrame for large-scale data processing.

* **pandas\_df = pd.DataFrame({…})**: Creates a Pandas DataFrame pandas\_df with columns a, b, c, d, and e using sample data.
* **df = spark.createDataFrame(pandas\_df):** Converts Pandas DataFrame pandas\_df into a PySpark DataFrame df.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

pandas\_df = pd.DataFrame({

'a': [1, 2, 3],

'b': [4., 8., 5.],

'c': ['GFG1', 'GFG2', 'GFG3'],

'd': [date(2000, 8, 1), date(2000, 6, 2),

date(2000, 5, 3)],

'e': [datetime(2000, 8, 1, 12, 0),

datetime(2000, 6, 2, 12, 0),

datetime(2000, 5, 3, 12, 0)]

})

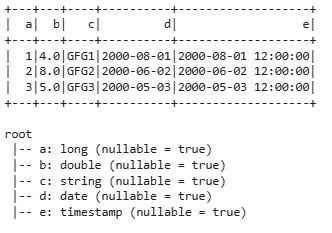
df = spark.createDataFrame(pandas\_df)

df

df.show()

df.printSchema()

**Output:**



*Using Pandas DataFrame*

**3. Create DataFrame from an RDD**

We can convert an existing RDD (Resilient Distributed Dataset) into a DataFrame for structured data processing.

* **rdd = spark.sparkContext.parallelize([ … ]):** Creates an RDD from a list of tuples where each tuple represents a row of data.
* **df = spark.createDataFrame(rdd, schema=[‘a’, ‘b’, ‘c’, ‘d’, ‘e’])**: Converts RDD into a PySpark DataFrame and assigns column names (a, b, c, d, e) to DataFrame.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

rdd = spark.sparkContext.parallelize([

(1, 4., 'GFG1', date(2000, 8, 1), datetime(2000, 8, 1, 12, 0)),

(2, 8., 'GFG2', date(2000, 6, 2), datetime(2000, 6, 2, 12, 0)),

(3, 5., 'GFG3', date(2000, 5, 3), datetime(2000, 5, 3, 12, 0))

])

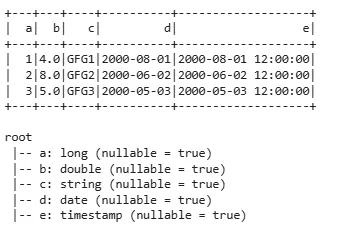
df = spark.createDataFrame(rdd, schema=['a', 'b', 'c', 'd', 'e'])

df

df.show()

df.printSchema()

**Output:**



*Using RDD*

**4. Create DataFrame from a CSV File**

PySpark can easily load data from a CSV file into a DataFrame. Here we are using random dataset for its implementation. Download the dataset from [train\_dataset](https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210512205547/train_dataset-1.csv).

* **df = spark.createDataFrame(pd.read\_csv(‘/content/train\_dataset-1.csv’)):** Reads a CSV file using Pandas **read\_csv()**function and then converts resulting Pandas DataFrame into PySpark DataFrame.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

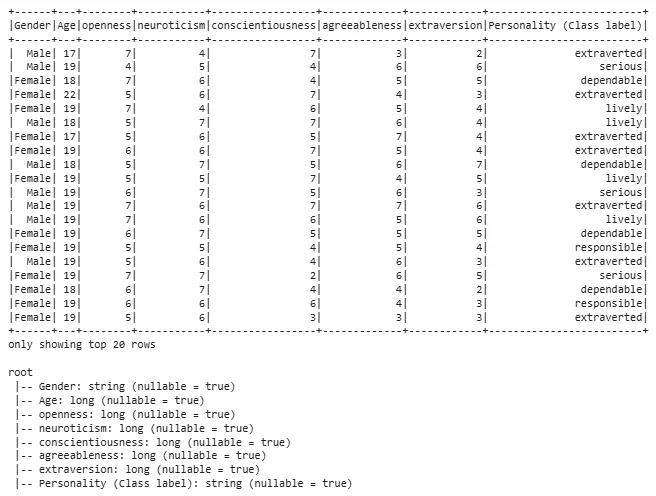
df = spark.createDataFrame(pd.read\_csv('/content/train\_dataset-1.csv'))

df

df.show()

df.printSchema()

**Output:**



*Using a CSV File*

**5. Create PySpark DataFrame from Text file**

If our data is stored in a plain text file we can load each line as a row using the **read.text()** method. Here we are using a random .txt file which can be downloaded from [here](https://media.geeksforgeeks.org/wp-content/uploads/20250503172613340152/text_file.zip).

* **df = spark.createDataFrame(pd.read\_csv(‘/content/text\_file.txt’, delimiter=”\t”)):** Reads text file using **pandas.read\_csv()**to load it into Pandas DataFrame.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

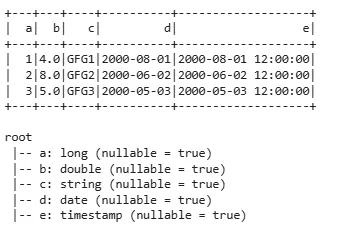
df = spark.createDataFrame(pd.read\_csv('/content/text\_file.txt', delimiter="**\t**"))

df

df.show()

df.printSchema()

**Output:**



*Using aext file*

**6. Create DataFrame from JSON**

JSON is a common format used for structured data. We can use **read.json()** to load data from JSON files directly into a PySpark DataFrame. The file we are using can be downloaded from [here](https://media.geeksforgeeks.org/wp-content/uploads/20250503172936988905/json_data.zip).

* **df = spark.createDataFrame(pd.read\_json(‘/content/json\_data.json’)):** Reads a JSON file using **pandas.read\_json()**to load it into a Pandas DataFrame.

**from** **datetime** **import** datetime, date

**import** **pandas** **as** **pd**

**from** **pyspark.sql** **import** Row

**from** **pyspark.sql** **import** SparkSession

spark = SparkSession.builder.getOrCreate()

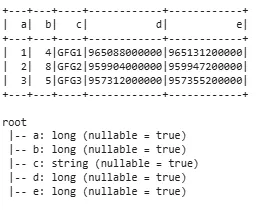
df = spark.createDataFrame(pd.read\_json('/content/json\_data.json'))

df

df.show()

df.printSchema()

**Output:**



*Using a JSON file*

PySpark’s process large-scale datasets using DataFrames and its integration with Spark’s distributed computing framework makes it important for data science work.

Refer link for PySpark Documentation for more function on Spark SQL, Spark Dataframe:

<https://spark.apache.org/docs/2.2.1/sql-programming-guide.html>

Practice applying all those function on different dataset