**pyspark.sql.types**

* [**ArrayType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html)
* [BinaryType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.BinaryType.html)
* [**BooleanType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.BooleanType.html)
* [ByteType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ByteType.html)
* [**DataType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DataType.html)
* [**DateType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DateType.html)
* [DecimalType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DecimalType.html)
* [DoubleType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DoubleType.html)
* [**FloatType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.FloatType.html)
* [**IntegerType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.IntegerType.html)
* [**LongType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.LongType.html)
* [MapType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.MapType.html)
* [NullType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.NullType.html)
* [ShortType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ShortType.html)
* [**StringType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.StringType.html)
* [**StructField**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.StructField.html)
* [**StructType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.StructType.html)
* [**TimestampType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.TimestampType.html)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Methods**   |  |  | | --- | --- | | [**fromInternal**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.fromInternal)(obj) | Converts an internal SQL object into a native Python object. | | [**fromJson**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.fromJson)(json) |  | | [**json**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.json)() |  | | [**jsonValue**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.jsonValue)() |  | | [**needConversion**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.needConversion)() | Does this type needs conversion between Python object and internal SQL object. | | [**simpleString**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.simpleString)() |  | | [**toInternal**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.toInternal)(obj) | Converts a Python object into an internal SQL object. | | [**typeName**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html#pyspark.sql.types.ArrayType.typeName)() |  | |

|  |
| --- |
| **PySpark StructType & StructField**  **These classes are used to programmatically specify the schema to the DataFrame and creating complex columns like nested struct, array and map columns.**   * **StructType class to define the structure of the DataFrame. StructType is a collection or list of StructField objects. StructType is a collection of StructFields.** * **StructField class to define the columns which includes:**   + **column name(String),**   + **column type (DataType),**   + **nullable column (Boolean) and**   + **metadata (MetaData)**   import pyspark  from pyspark.sql import SparkSession  from pyspark.sql.types import StructType,StructField, StringType, IntegerType  spark = SparkSession.builder.master("local[1]") \  .appName('SparkByExamples.com') \  .getOrCreate()  data = [("James","","Smith","36636","M",3000),  ]  schema = StructType([ \  StructField("firstname",StringType(),True), \  StructField("middlename",StringType(),True), \  StructField("lastname",StringType(),True), \  StructField("id", StringType(), True), \  StructField("gender", StringType(), True), \  StructField("salary", IntegerType(), True) \  ])  df = spark.createDataFrame(data=data,schema=schema)  df.printSchema()  df.show(truncate=False)  **If defining Nested StructType object struct:**  structureData = [  (("James","","Smith"),"36636","M",3100),  (("Michael","Rose",""),"40288","M",4300),  (("Robert","","Williams"),"42114","M",1400),  (("Maria","Anne","Jones"),"39192","F",5500),  (("Jen","Mary","Brown"),"","F",-1)  ]  structureSchema = StructType([  StructField('name', StructType([  StructField('firstname', StringType(), True),  StructField('middlename', StringType(), True),  StructField('lastname', StringType(), True)  ])),  StructField('id', StringType(), True),  StructField('gender', StringType(), True),  StructField('salary', IntegerType(), True)  ])  df2 = spark.createDataFrame(data=structureData,schema=structureSchema)  df2.printSchema()  🡪root  |-- name: struct (nullable = true)  | |-- firstname: string (nullable = true)  | |-- middlename: string (nullable = true)  | |-- lastname: string (nullable = true)  |-- id: string (nullable = true)  |-- gender: string (nullable = true)  |-- salary: integer (nullable = true)  **Using PySpark SQL function struct(), we can change the struct of the existing DataFrame and add a new StructType to it.**  from pyspark.sql.functions import col,struct,when  updatedDF = df2.withColumn(  "OtherInfo",  struct(  col("id").alias("identifier"),  col("gender").alias("gender"),  col("salary").alias("salary"),  when(col("salary").cast(IntegerType()) < 2000,"Low")  .when(col("salary").cast(IntegerType()) < 4000,"Medium")  .otherwise("High").alias("Salary\_Grade")  )  ).drop("id","gender","salary")  updatedDF.printSchema()  updatedDF.show(truncate=False)  🡪root  |-- name: struct (nullable = true)  | |-- firstname: string (nullable = true)  | |-- middlename: string (nullable = true)  | |-- lastname: string (nullable = true)  |-- OtherInfo: struct (nullable = false)  | |-- identifier: string (nullable = true)  | |-- gender: string (nullable = true)  | |-- salary: integer (nullable = true)  | |-- Salary\_Grade: string (nullable = false)  **Using SQL ArrayType and MapType: On the below example, column hobbies defined as ArrayType(StringType) and properties defined as MapType(StringType,StringType) meaning both key and value as String.**  arrayStructureSchema = StructType([  StructField('name',  StructType([  StructField('firstname', StringType(), True),  StructField('middlename', StringType(), True),  StructField('lastname', StringType(), True)  ])  ),  StructField('hobbies', ArrayType(StringType()), True),  StructField('properties', MapType(StringType(),StringType()), True)  ])  root  |-- name: struct (nullable = true)  | |-- firstname: string (nullable = true)  | |-- middlename: string (nullable = true)  | |-- lastname: string (nullable = true)  |-- hobbies: array (nullable = true)  | |-- element: string (containsNull = true)  |-- properties: map (nullable = true)  | |-- key: string  | |-- value: string (valueContainsNull = true)  **Checking if a Column Exists in a DataFrame:**  print(df.schema.fieldNames.contains("firstname"))  # To check every property in a field:  print(df.schema.contains(StructField("firstname",StringType,true)))  **Complete Example of PySpark StructType & StructField:**  import pyspark  from pyspark.sql import SparkSession  from pyspark.sql.types import StructType,StructField, StringType, IntegerType,ArrayType,MapType  from pyspark.sql.functions import col,struct,when  spark = SparkSession.builder.master("local[1]") \  .appName('SparkByExamples.com') \  .getOrCreate()  data = [("James","","Smith","36636","M",3000),  ("Michael","Rose","","40288","M",4000),  ("Robert","","Williams","42114","M",4000),  ("Maria","Anne","Jones","39192","F",4000),  ("Jen","Mary","Brown","","F",-1)  ]  schema = StructType([  StructField("firstname",StringType(),True),  StructField("middlename",StringType(),True),  StructField("lastname",StringType(),True),  StructField("id", StringType(), True),  StructField("gender", StringType(), True),  StructField("salary", IntegerType(), True)  ])    df = spark.createDataFrame(data=data,schema=schema)  df.printSchema()  df.show(truncate=False)  structureData = [  (("James","","Smith"),"36636","M",3100),  (("Michael","Rose",""),"40288","M",4300),  (("Robert","","Williams"),"42114","M",1400),  (("Maria","Anne","Jones"),"39192","F",5500),  (("Jen","Mary","Brown"),"","F",-1)  ]  structureSchema = StructType([  StructField('name', StructType([  StructField('firstname', StringType(), True),  StructField('middlename', StringType(), True),  StructField('lastname', StringType(), True)  ])),  StructField('id', StringType(), True),  StructField('gender', StringType(), True),  StructField('salary', IntegerType(), True)  ])  df2 = spark.createDataFrame(data=structureData,schema=structureSchema)  df2.printSchema()  df2.show(truncate=False)  updatedDF = df2.withColumn("OtherInfo",  struct(col("id").alias("identifier"),  col("gender").alias("gender"),  col("salary").alias("salary"),  when(col("salary").cast(IntegerType()) < 2000,"Low")  .when(col("salary").cast(IntegerType()) < 4000,"Medium")  .otherwise("High").alias("Salary\_Grade")  )).drop("id","gender","salary")  updatedDF.printSchema()  updatedDF.show(truncate=False)  """ Array & Map"""  arrayStructureSchema = StructType([  StructField('name', StructType([  StructField('firstname', StringType(), True),  StructField('middlename', StringType(), True),  StructField('lastname', StringType(), True)  ])),  StructField('hobbies', ArrayType(StringType()), True),  StructField('properties', MapType(StringType(),StringType()), True)  ]) |

|  |
| --- |
| **pyspark.sql.types.ArrayType(elementType, containsNull=True)**  same for:   * [ArrayType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ArrayType.html) * [BinaryType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.BinaryType.html) * [BooleanType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.BooleanType.html) * [ByteType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ByteType.html) * [DataType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DataType.html) * [DateType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DateType.html) * [DecimalType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DecimalType.html) * [DoubleType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DoubleType.html) * [FloatType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.FloatType.html) * [IntegerType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.IntegerType.html) * [LongType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.LongType.html) * [MapType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.MapType.html) * [NullType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.NullType.html) * [ShortType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.ShortType.html) * [StringType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.StringType.html) * TimestampType   elementType[*DataType*](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DataType.html#pyspark.sql.types.DataType)  [DataType](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DataType.html#pyspark.sql.types.DataType) of each element in the array.  containsNull*bool, optional*  whether the array can contain null (None) values.  >>>ArrayType(StringType()) == ArrayType(StringType(), True)  True  >>>ArrayType(StringType(), False) == ArrayType(StringType())  False |

|  |
| --- |
| **pyspark.sql.types.StructField(name, dataType, nullable=True, metadata=None)**  A field in StructType.  **Name: *str***  name of the field.  **dataType:** [***DataType***](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DataType.html#pyspark.sql.types.DataType)  [**DataType**](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.types.DataType.html#pyspark.sql.types.DataType) of the field.  **Nullable: *bool, optional***  whether the field can be null (None) or not.  **Metadat: a*dict, optional***  a dict from string to simple type that can be toInternald to JSON automatically  >>>(StructField("f1", StringType(), True)  == StructField("f1", StringType(), True))  True  >>>(StructField("f1", StringType(), True)  == StructField("f2", StringType(), True))  False |

|  |
| --- |
| **pyspark.sql.types.StructType(fields=None)**  Struct type, consisting of a list of StructField.  This is the data type representing a Row.  Iterating a StructType will iterate over its StructFields. A contained StructField can be accessed by its name or position.  >>>struct1 = StructType([StructField("f1", StringType(), True)])  >>>struct1["f1"]  StructField(f1,StringType,true)  >>>struct1[0]  StructField(f1,StringType,true)  >>>struct1 = StructType([StructField("f1", StringType(), True)])  >>>struct2 = StructType([StructField("f1", StringType(), True)])  >>>struct1 == struct2  True  >>>struct1 = StructType([StructField("f1", StringType(), True)])  >>>struct2 = StructType([StructField("f1", StringType(), True),  StructField("f2", IntegerType(), False)])  >>>struct1 == struct2  False  >>>struct1 = StructType().add("f1", StringType(), True).add("f2", StringType(), True, None)  >>>struct2 = StructType([StructField("f1", StringType(), True), \  StructField("f2", StringType(), True, None)])  >>>struct1 == struct2  True  >>>struct1 = StructType().add(StructField("f1", StringType(), True))  >>>struct2 = StructType([StructField("f1", StringType(), True)])  >>>struct1 == struct2  True  >>>struct1 = StructType().add("f1", "string", True)  >>>struct2 = StructType([StructField("f1", StringType(), True)])  >>>struct1 == struct2  True |