Spark DataFrames

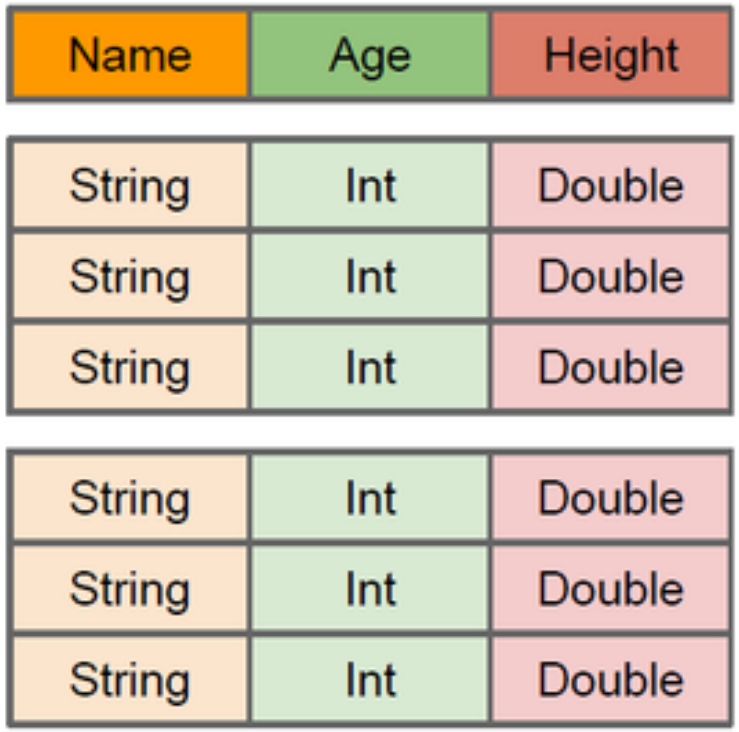
Anurag Nagar

Big Data Class

Outline

Introduction

- DataFrames are part of **Spark SQL**.
- Like RDDs, DataFrames (DF) are **immutable, distributed, partitioned** collection of data
- They have all the properties of RDDs, such as lazy evaluation, recovery through lineage graphs, etc.
- They contain specialized APIs for working with **tabular** data, and have **named columns.**

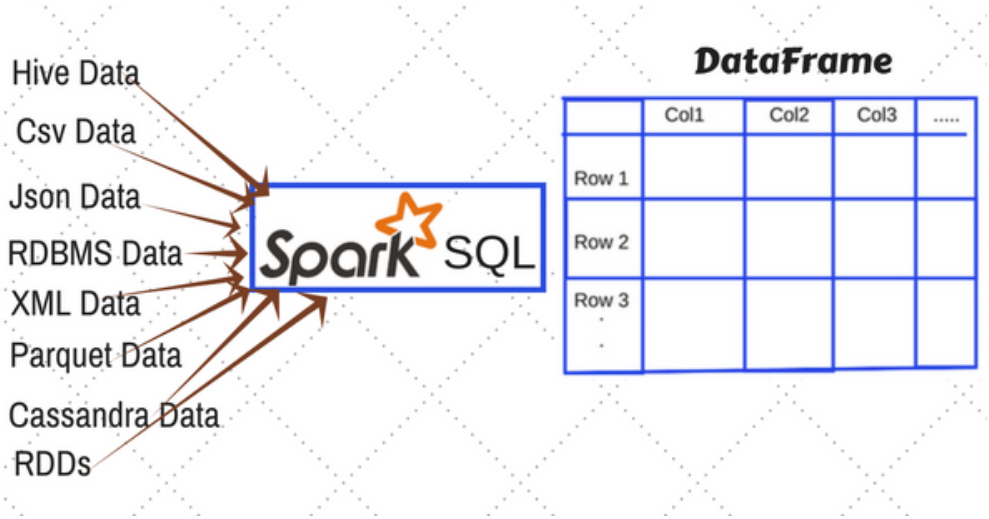


DataFrame

Outline

Creating DataFrames

- DataFrames are well suited for large structured or semi-structured data.
- Data can be loaded easily from a wide variety of sources
- DF contain named columns, and a list of tuples



Outline

Loading Data into DataFrames

spark.read is the starting point to read data into DF. More details can be found at this link.

- To read a simple CSV file with header

```
df = spark.read.load("PATH", format="csv", sep=",",
     inferSchema="true", header="true")
```

Loading Data into DataFrames

spark.read is the starting point to read data into DF. More details can be found at this link.

- To read a simple CSV file with header

```
df = spark.read.load("PATH", format="csv", sep=",",
     inferSchema="true", header="true")
```

- To see schema

```
df.printSchema()
```

Loading Data into DataFrames

spark.read is the starting point to read data into DF. More details can be found at this link.

- To read a simple CSV file with header

```
df = spark.read.load("PATH", format="csv", sep=",",
     inferSchema="true", header="true")
```

- To see schema

```
df.printSchema()
```

- To see first 10 rows

```
df.take(10)
```

Outline

Selection and Projection

- To extract few columns

```
filtered = df.select(["column1", "column2"])
```

Selection and Projection

- To extract few columns

```
filtered = df.select(["column1", "column2"])
```

- To filter data with conditions:

```
selected = df.filter(df['column'] > condition)
# example
df.filter(df["age"] > 21).show()
```

Outline

Ordering

- To order by a column

```
from pyspark.sql.functions import desc, asc
from pyspark.sql.functions import col, column
df.orderBy(expr("count desc")).show(2)
# another way
df.orderBy(col(" first ").desc(),
      col("second").asc()).show(2)
# another way
df.orderBy("age", desc("name").show()
```

Outline

Grouping Data

- To group by a column and get count of groups:

```
df.groupBy("age").count()
```

Grouping Data

- To group by a column and get count of groups:

```
df.groupBy("age").count()
```

- To group by a column and show average of another column by group

```
df.groupBy("department").avg("salary")
```

Grouping Data

- To group by a column and get count of groups:

```
df.groupBy("age").count()
```

- To group by a column and show average of another column by group

```
df.groupBy("department").avg("salary")
```

- To find other stats

```
df.groupBy("department")
    .agg(sum("salary").alias("sum_salary"),
         avg("salary").alias("avg_salary"),
         sum("bonus").alias("sum_bonus"),
         max("bonus").alias("max_bonus"))
```

Outline

Joining Data

- To join two DF

```
df = left.join(right, left.name == right.name, "inner")
```

Joining Data

- To join two DF

```
df = left.join(right, left.name == right.name, "inner")
```

- To do left/right outer join

```
df = left.join(right, left["name"] == right["name"],
    "leftOuter")
```

the last parameter can be *inner*, *outer*, *leftOuter*, *rightOuter*