Dataframes & Spark SQL

Spark SQL

Spark module
for
Structured Data Processing

Spark SQL

Integrated

- Provides **DataFrames**
- Mix SQL queries & Spark programs

Spark SQL

Uniform Data Access

- Source:
  - HDFS,
  - Hive
  - Relational Databases
- Avro, Parquet, ORC, JSON
- You can even join data across these sources.
- Hive Compatibility
- Standard Connectivity

DataFrames

RDD

1 sandeep
2 ted
3 thomas
4 priya
5 kush

Unstructured

Need code for processing

DataFrames

RDD

1 sandeep
2 ted
3 thomas
4 priya
5 kush

Unstructured

Need code for processing

Data Frame

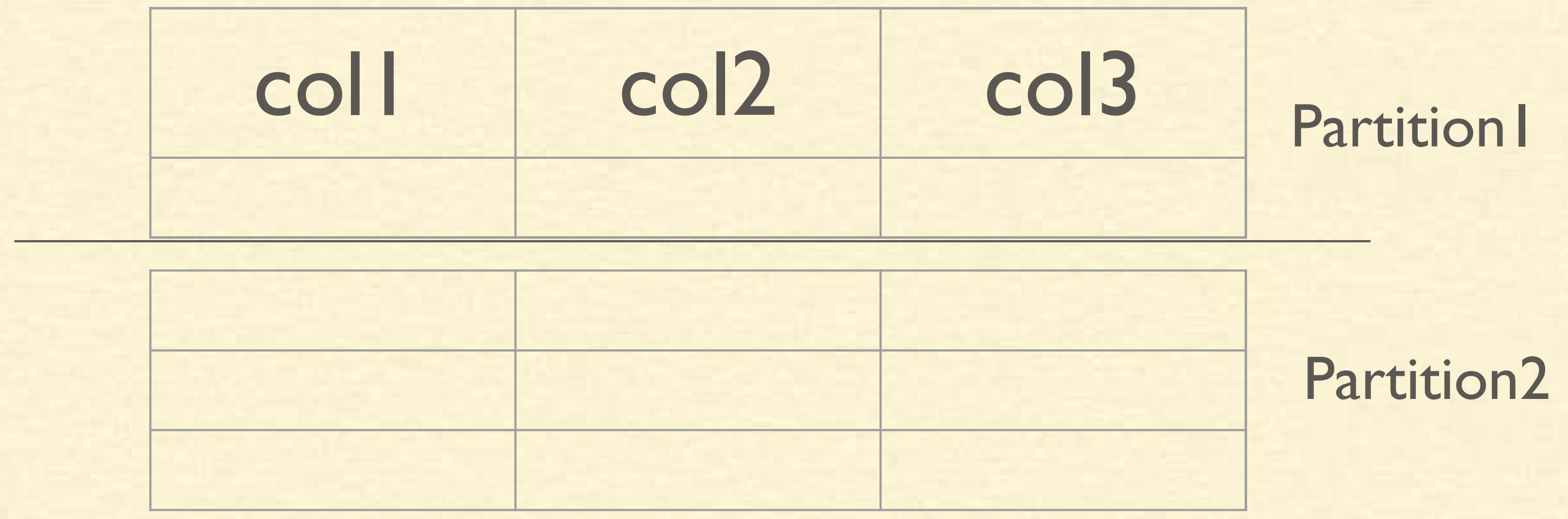
ID	Name
1	sandeep
2	ted
3	thomas
4	priya
5	kush

Structured

Can use SQL or R like syntax:
df.sql("select Id where name = 'priya'")

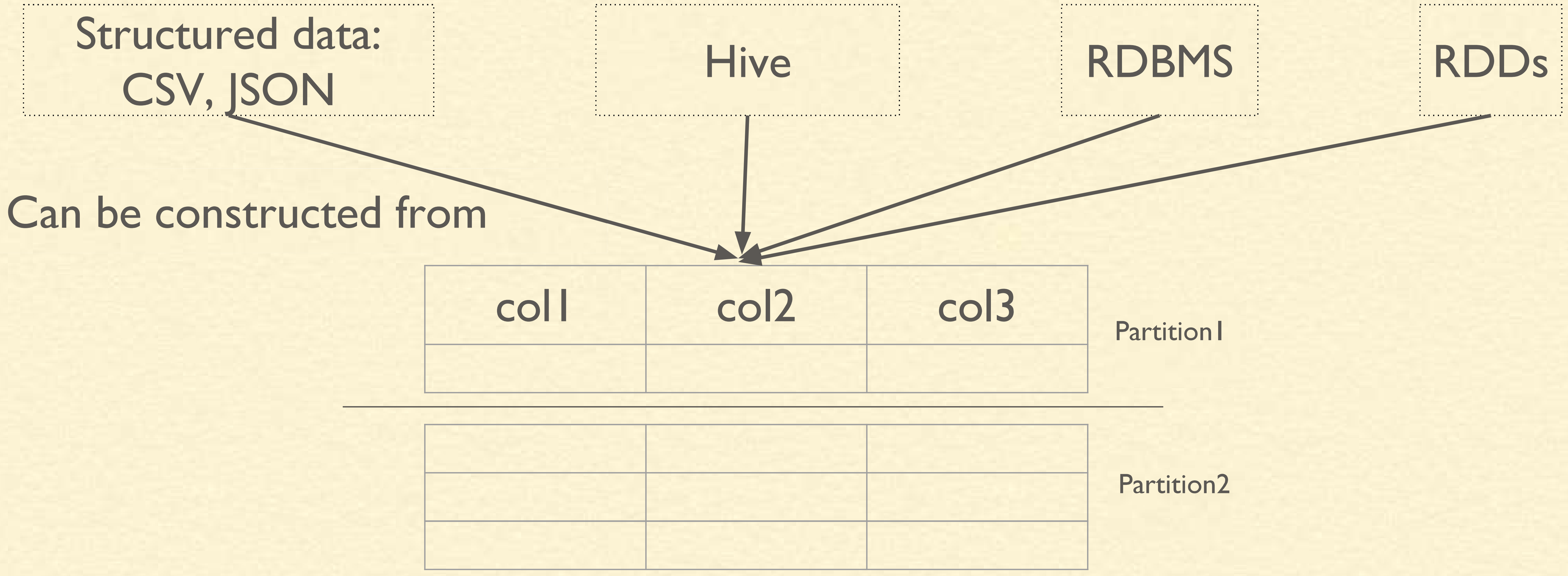
head(where(df, df$ID > 21))

Data Frames

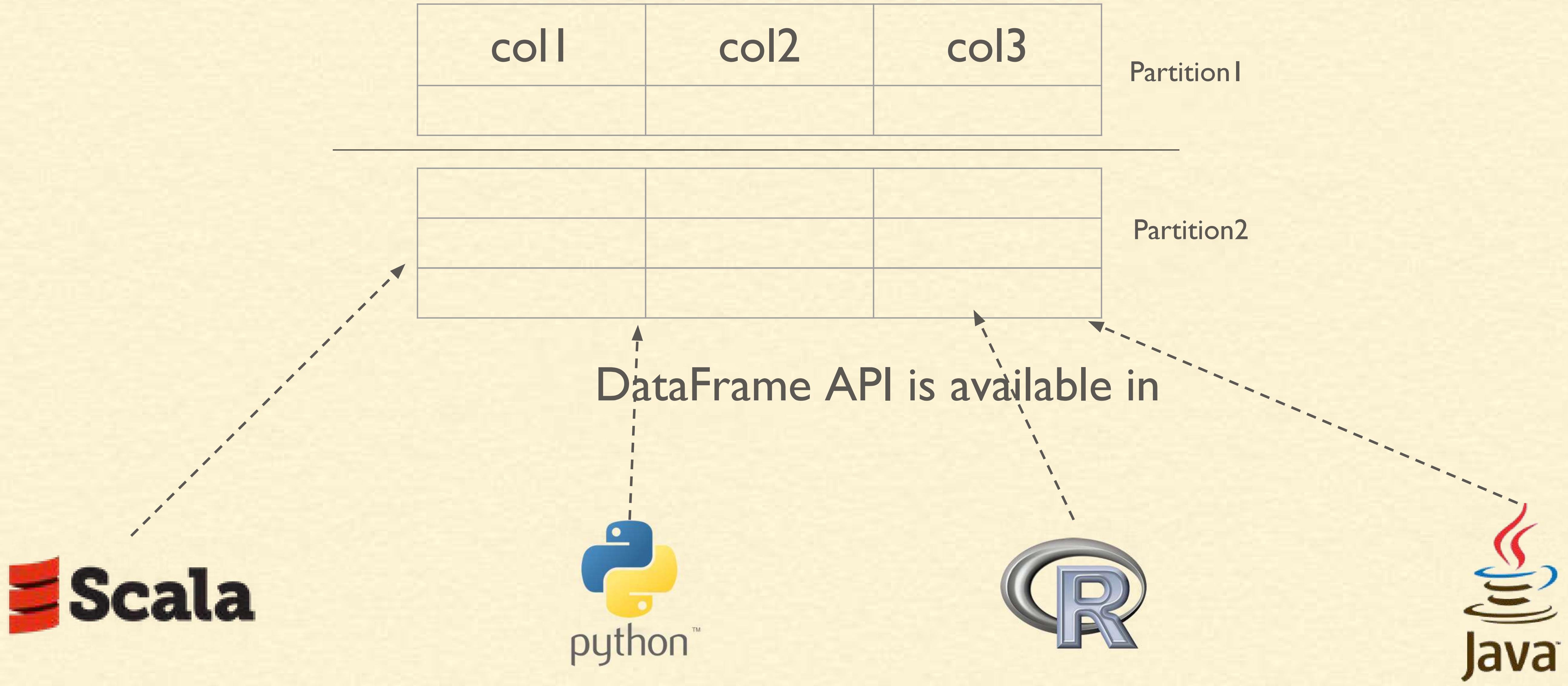


- Collection with named columns
- Distributed
- <> Same as database table
- <> A data frame in R/Python

Data Frames


Structured data: CSV, JSON
Hive
RDBMS
RDDs
Can be constructed from
col1
col2
col3
Partition1
Partition2

Data Frames


col1
col2
col3
Partition1
Partition2
DataFrame API is available in
Scala
python
R
Java

Getting Started

- Available in Spark 2.0x onwards.
- Using usual interfaces
    - Spark-shell
    - Spark Application
    - Pyspark
    - Java
    - etc.

Getting Started

```
$ export HADOOP_CONF_DIR=/etc/hadoop/conf/
$ export YARN_CONF_DIR=/etc/hadoop/conf/
```

Getting Started

```
$ export HADOOP_CONF_DIR=/etc/hadoop/conf/
$ export YARN_CONF_DIR=/etc/hadoop/conf/
$ ls /usr/
bin   games   include   jdk64   lib64     local   share         spark1.6
spark2.0.2   tmp etc   hdp     java      lib     libexec   sbin    spark1.2.1
spark2.0.1   src
```

Getting Started

```
$ export HADOOP_CONF_DIR=/etc/hadoop/conf/
$ export YARN_CONF_DIR=/etc/hadoop/conf/
$ ls /usr/
bin   games   include   jdk64   lib64     local   share       spark1.6
spark2.0.2   tmp etc   hdp     java      lib     libexec   sbin    spark1.2.1
spark2.0.1   src
$ /usr/spark2.0.2/bin/spark-shell
```

Getting Started

```
Spark context Web UI available at http://172.31.60.179:4040
Spark context available as 'sc' (master = local[*], app id = local-1498489557917).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.2
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Getting Started

```
Spark context Web UI available at http://172.31.60.179:4040
Spark context available as 'sc' (master = local[*], app id = local-1498489557917).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.2
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Starting Point: SparkSession

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession
  .builder()
  .appName("Spark SQL basic example")
  .config("spark.some.config.option", "some-value")
  .getOrCreate()
```

Starting Point: SparkSession

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession
  .builder()
  .appName("Spark SQL basic example")
  .config("spark.some.config.option", "some-value")
  .getOrCreate()

//For implicit conversions, e.g. RDDs to DataFrames
import spark.implicits._
```

Creating DataFrames from JSON

```
In web console or ssh:
$ hadoop fs -cat /data/spark/people.json
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

Creating DataFrames from JSON

```
var df = spark.read.json("/data/spark/people.json")

# Displays the content of the DataFrame to stdout
df.show()
```

```
scala> df.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

Creating DataFrames from JSON

```
var df = spark.read.json("/data/spark/people.json")

# Displays the content of the DataFrame to stdout
df.show()
```

```
scala> df.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

Original JSON:

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

DataFrame Operations

```
# Print the schema in a tree format
df.printSchema()

root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

DataFrame Operations

```
# Select only the "name" column
df.select("name").show()
+--------+
|    name|
+--------+
|Michael|
|    Andy|
|  Justin|
+--------+
```

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

DataFrame Operations

```
# Increment the age by 1
df.select($"name",$"age" + 1).show()

+-------+---------+
|   name|(age + 1)|
+-------+---------+
|Michael|     null|
|   Andy|       31|
| Justin|       20|
+-------+---------+
```

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

DataFrame Operations

```
# Select people older than 21
df.filter($"age"> 21).show()
+---+-----+
|age|name|
+---+-----+
| 30|Andy|
+---+-----+
```

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

DataFrame Operations

```
# Count people by age
df.groupBy("age").count().show()
+----+------+
| age|count|
+----+------+
|  19|     1|
|null|     1|
|  30|     1|
+----+------+


#SQL Equivalent
Select age, count(*) from df group by age
```

```
{"name":"Michael"}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
```

Running SQL Queries Programmatically

Running SQL Queries Programmatically

```
// Register the DataFrame as a SQL temporary view
df.createOrReplaceTempView("people")
```

Running SQL Queries Programmatically

```
// Register the DataFrame as a SQL temporary view
df.createOrReplaceTempView("people")

val sqlDF = spark.sql("SELECT * FROM people")
```

Running SQL Queries Programmatically

```
// Register the DataFrame as a SQL temporary view
df.createOrReplaceTempView("people")

val sqlDF = spark.sql("SELECT * FROM people")
sqlDF.show()
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```