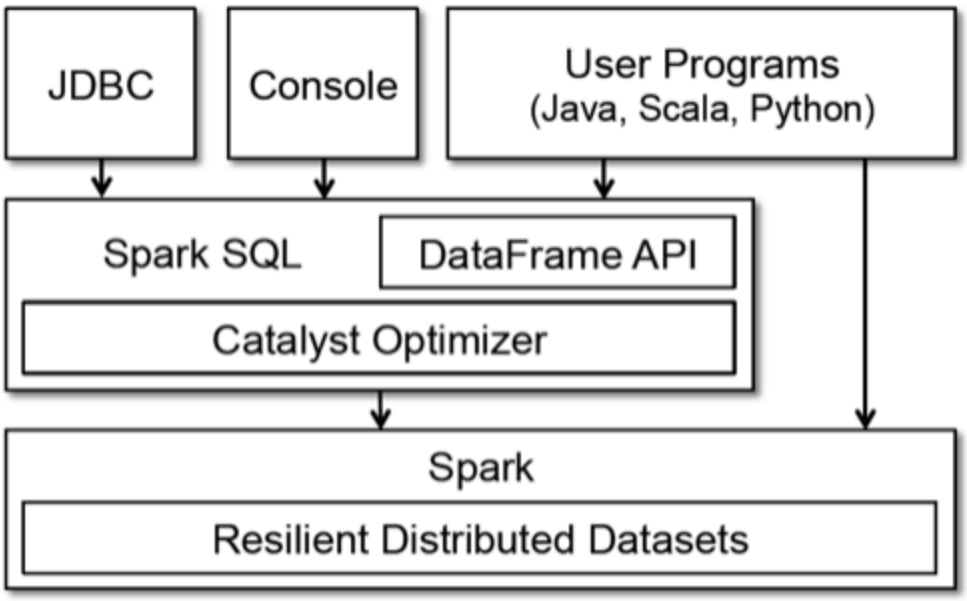
- SparkSQL is used to query data on spark easily
- especially for structured data with schemas
- No need for tedious spark RDDs
- simple SQL queries automatically translated to spark RDDs

GOALS:

- support relational processing both within spark programs and external data sources using friendly API
- high performance using established DBMS techniques
- support new data sources, including semi-structured data and external databases amenable to query federation
- enable graph processing (advanced analytics) and external databases
- enable the use of advanced analytics algorithms, like graph processing and ml

SPARK SQL ARCHITECTURE



DataFrame:

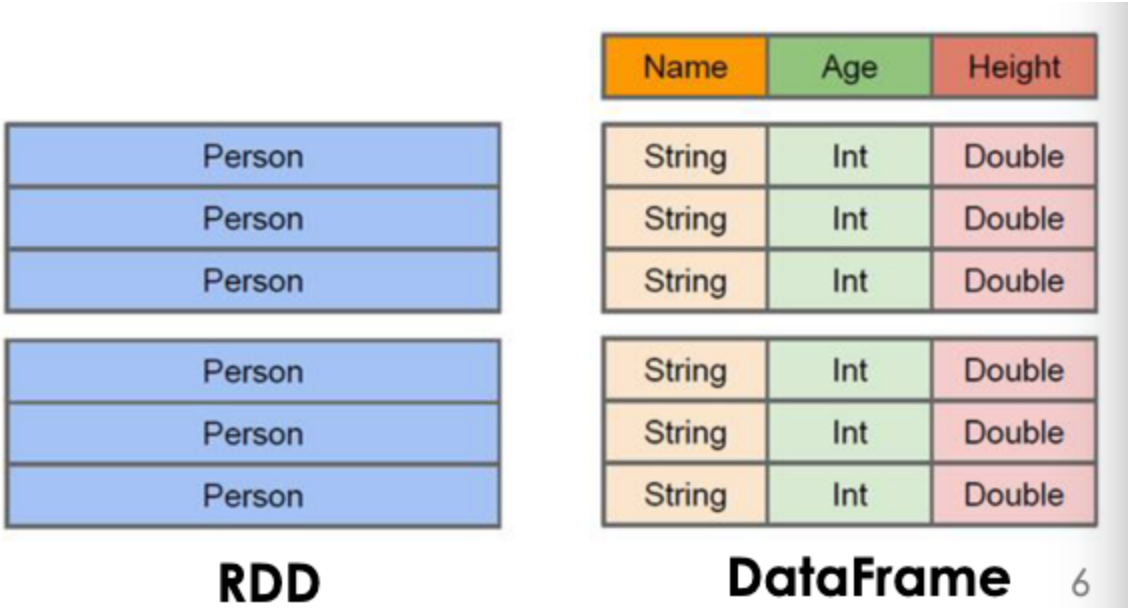
- new concept to abstract RDDs for structured data (like a table)

Types of interfaces:

- DataFrame API
- SQL over DataFrame

DataFrame is designed for handling structured, distributed data in a table-like representation with named columns and declared column types. It is a higher-level abstraction than RDD.

- has schema(types), allows more meaningful operations/queries over columns and rows.
- For RDD we only know that there is a collection of items (not knowing data type of each fiels)



We can create DFs from a csv, json

- Schema will be automatically inferred
- Can specify options("inferSchema","true")

We can print out the dataframe `dfs.printSchema()` note nullable means the column can be null.

Can also create data frame from RDD with/without schema

Two approaches for query DataFrame:

- DataFrame operations
- SQL queries over DataFrame

Operations:

- select one or more columns
- limit(k) to print out first k results
- filter to add some condition like age > 23
- GroupBy and use max(),min(),avg()
- sort(), orderby(), (by default we have ascending order)

- join two dataframes based on common attributes

```
employees
  .join(dept, employees("deptId") === dept("id"))
  .where(employees("gender") === "female")
  .groupBy(dept("id"), dept("name"))
  .agg(count("name"))
```

SQL Queries over DataFrame

- DFs can be registered as temporary tables in the system catalog and queried using SQL

```
users.where(users("age") < 21)
     .registerTempTable("young")
ctx.sql("SELECT count(*), avg(age) FROM young'
```