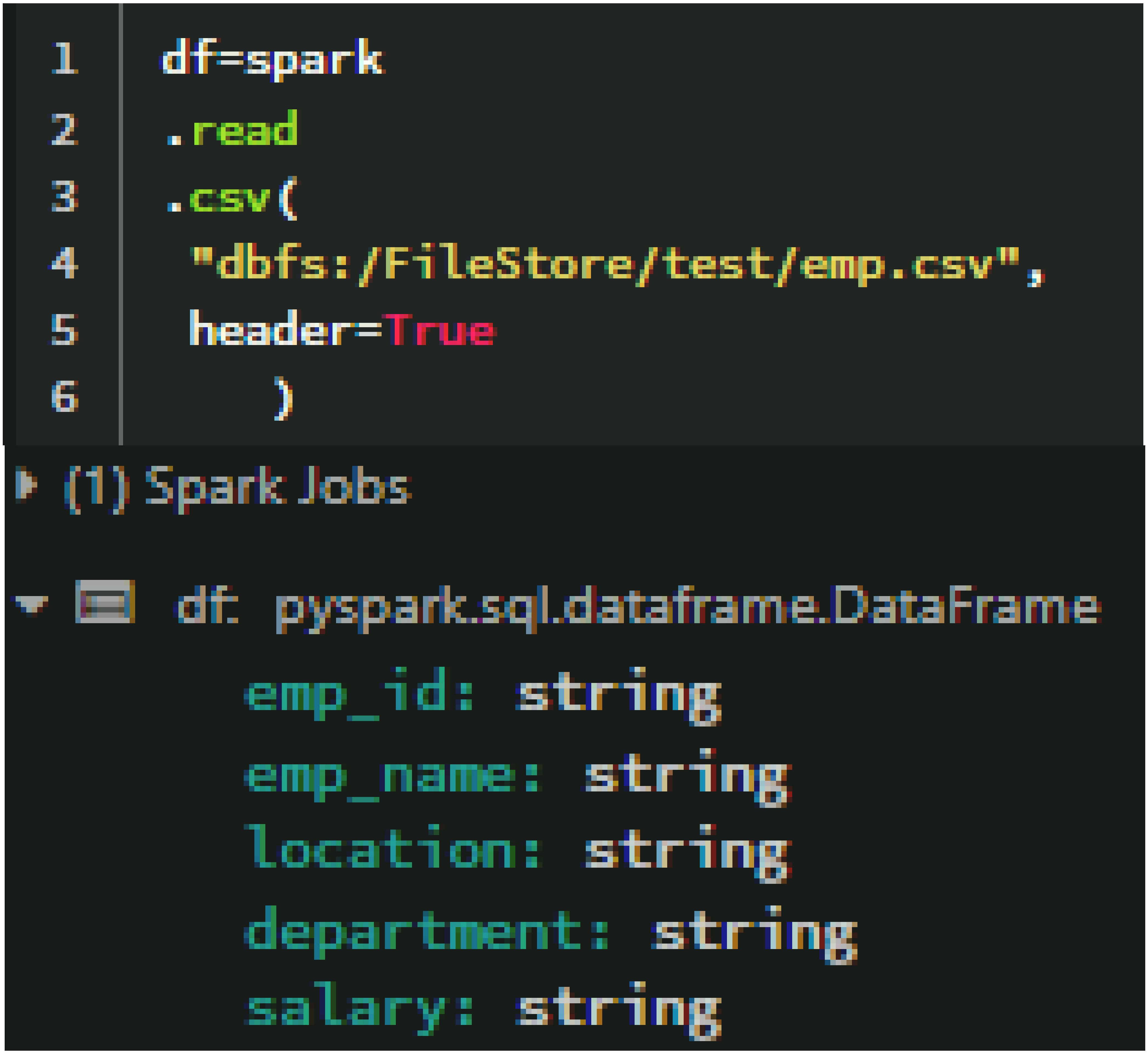
Read CSV with Apache Spark



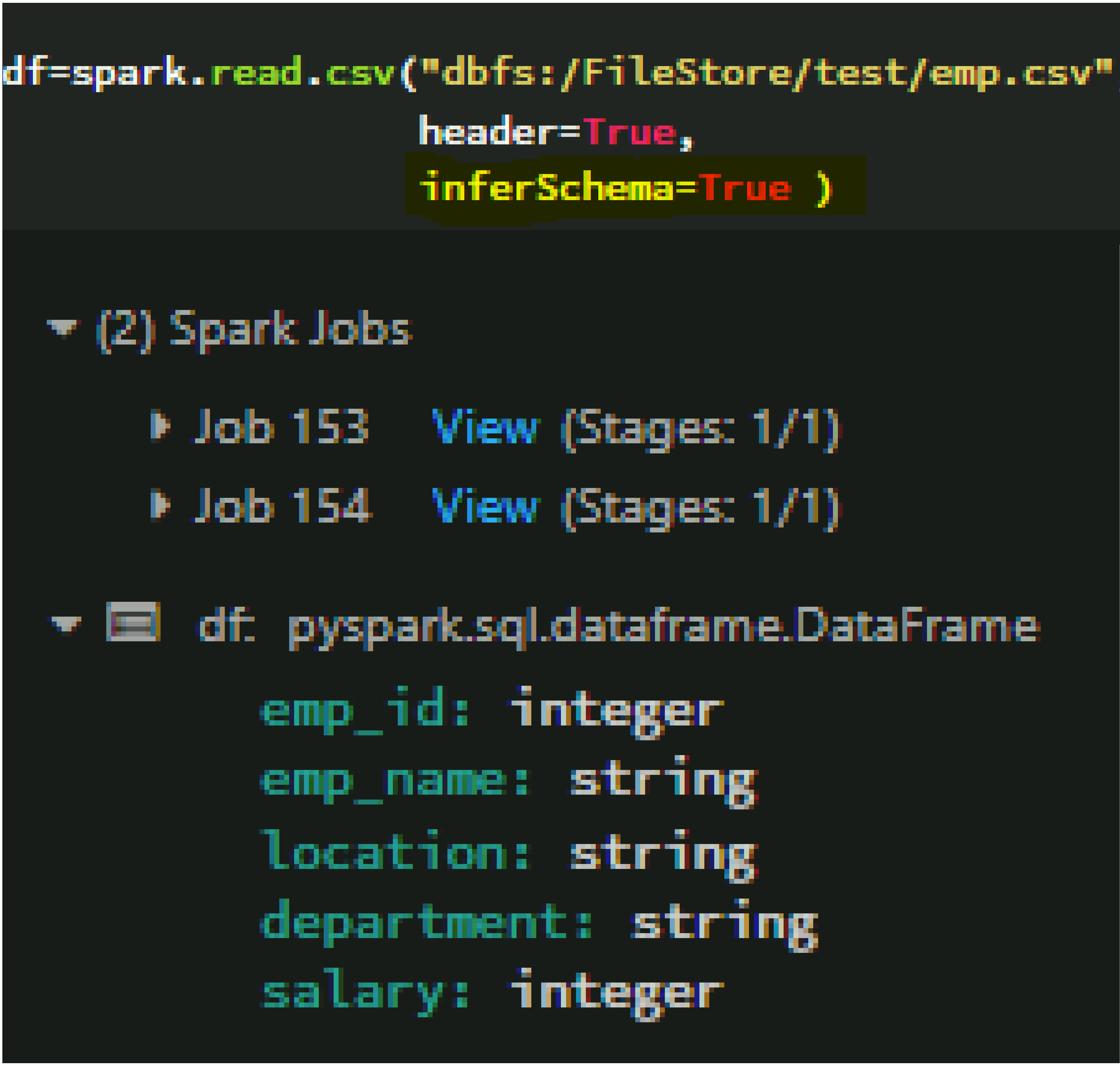
```
df=spark
.read
.csv(
 "dbfs:/FileStore/test/emp.csv",
 header=True
    )
```

```
(1) Spark Jobs

df:  pyspark.sql.dataframe.DataFrame
    emp_id: string
    emp_name: string
    location: string
    department: string
    salary: string
```

Every fiield is read as STRING when schema not mentioned

Read CSV with Apache Spark



```
df=spark.read.csv("dbfs:/FileStore/test/emp.csv"
                        header=True,
                        inferSchema=True )
```

```
▾ (2) Spark Jobs
    ▸ Job 153    View (Stages: 1/1)
    ▸ Job 154    View (Stages: 1/1)

▾ df:  pyspark.sql.dataframe.DataFrame
        emp_id: integer
        emp_name: string
        location: string
        department: string
        salary: integer
```

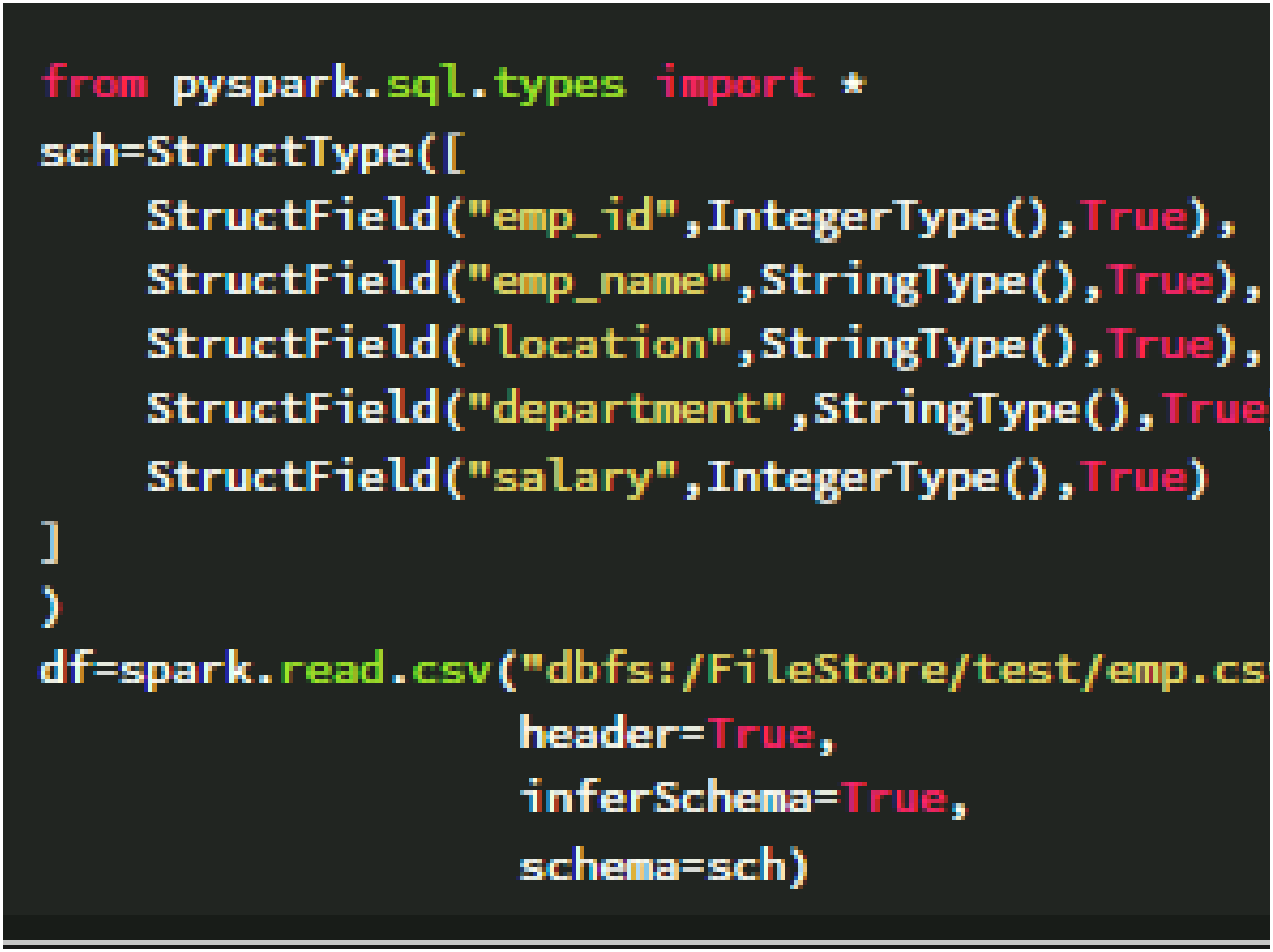
The CSV file is read with inferScehma Option. What happens? See next →

Read CSV with Apache Spark

When CSV is read with InferSchema option it runs an additional Spark job(Check that 2 Jobs run instead of 1) to infer the schema from a sample and that scema is enforced.

It is useful when we are not aware of the schema of the input file. But there is performance overhead due to additional job

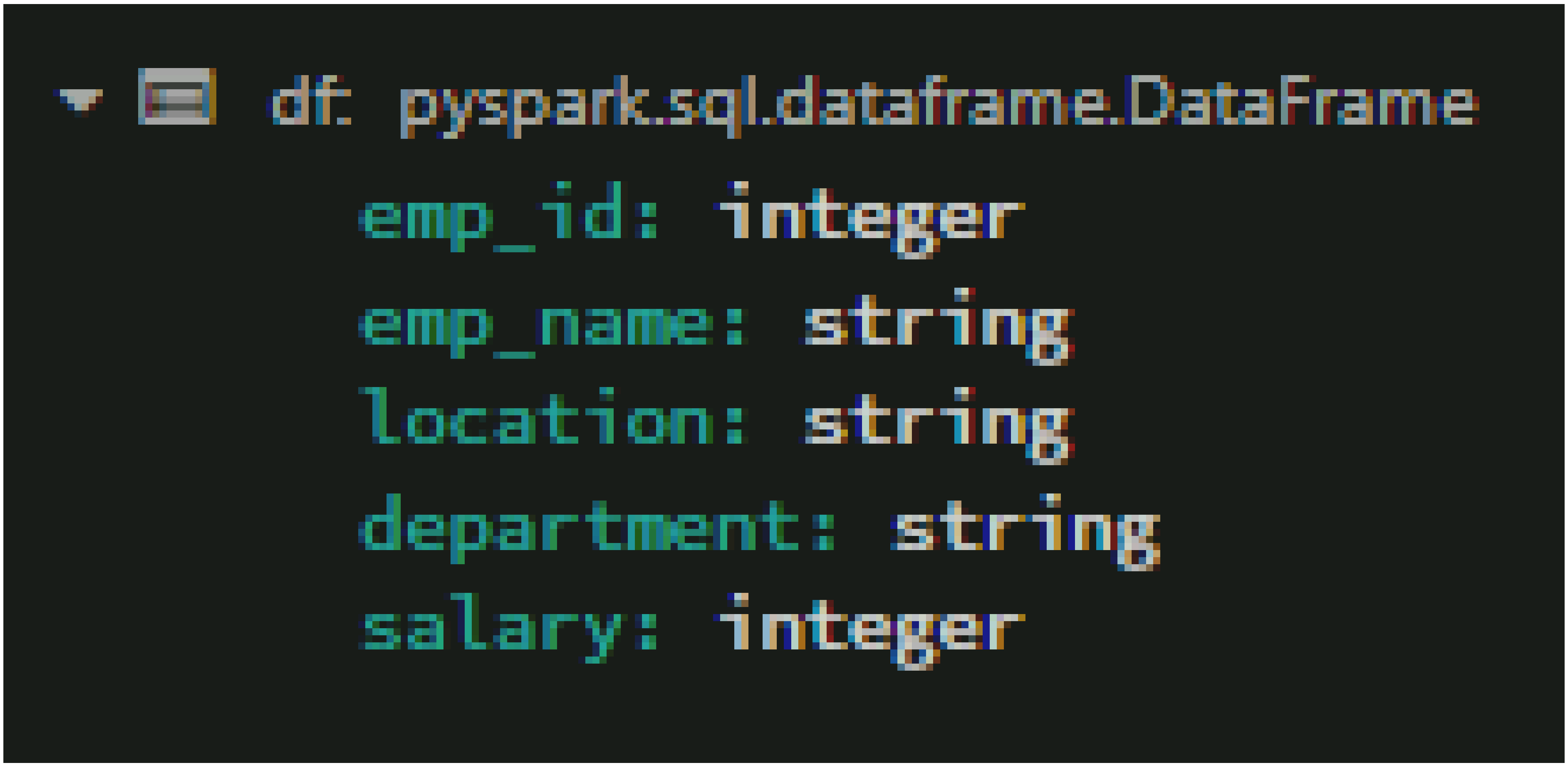
Read CSV with Apache Spark



```
from pyspark.sql.types import *
sch=StructType([
    StructField("emp_id",IntegerType(),True),
    StructField("emp_name",StringType(),True),
    StructField("location",StringType(),True),
    StructField("department",StringType(),True
    StructField("salary",IntegerType(),True)
]
)
df=spark.read.csv("dbfs:/FileStore/test/emp.cs
                  header=True,
                  inferSchema=True,
                  schema=sch)
```

We have enforced custom schema using StructType. What happens? See next →

Read CSV with Apache Spark



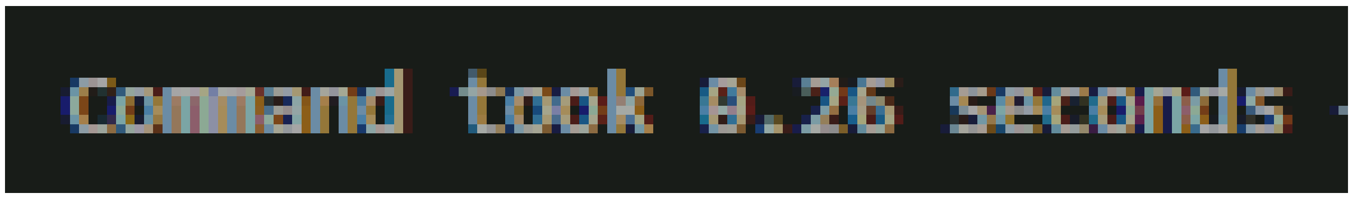
```
▾ df:  pyspark.sql.dataframe.DataFrame
        emp_id: integer
        emp_name: string
        location: string
        department: string
        salary: integer
```

The schema of the dataframe is same as what mentioned in StructType and only one Spark Job runs. So better performance.

Performance Comparison

Method 1:

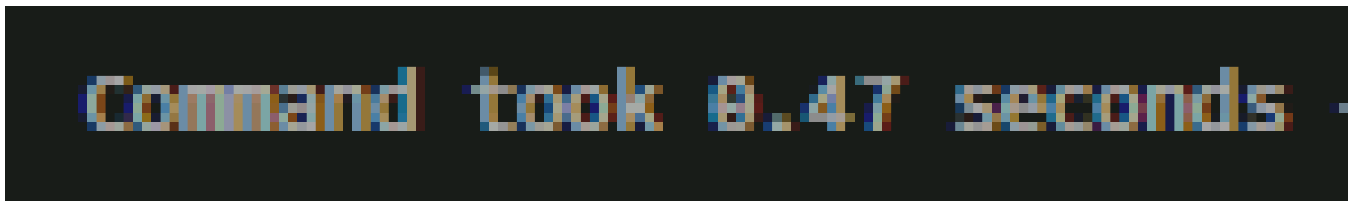
With No Schema. Everything read as STRING



```
Command took 0.26 seconds
```

Method 2:

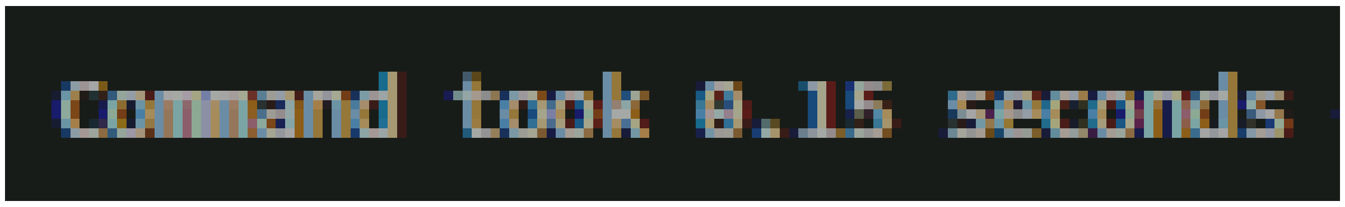
With InferSchema. Additional Saprk Job



```
Command took 0.47 seconds
```

Method 3:

With Custom Schema. Most Performant



```
Command took 0.15 seconds
```