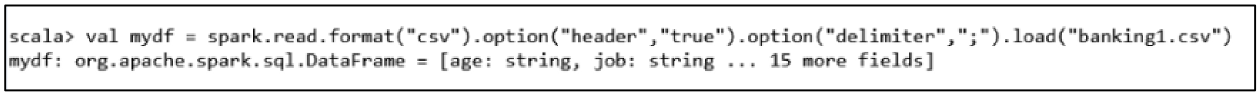
Step 1: Load the data and making a data frame.

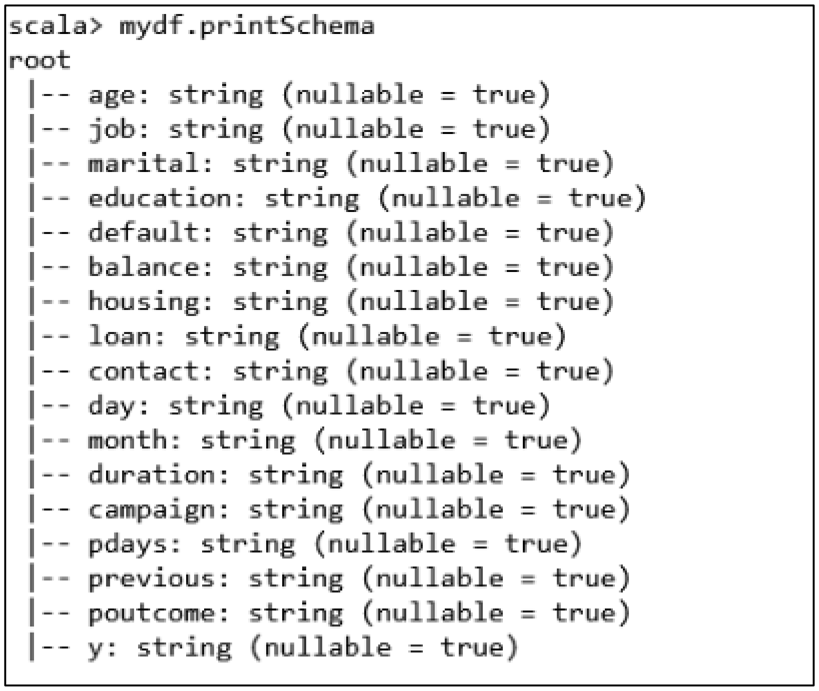
We need to read the csv file and choose the delimiter option so that features get separated by a semicolon.

```
scala> val mydf= spark.read.format("csv").option("header","true").
option("delimiter",";").load("banking.csv")
```



```
scala> val mydf = spark.read.format("csv").option("header","true").option("delimiter",";").load("banking1.csv")
mydf: org.apache.spark.sql.DataFrame = [age: string, job: string ... 15 more fields]
```

```
scala> data.printSchema
```



```
scala> mydf.printSchema
root
 |-- age: string (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: string (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- day: string (nullable = true)
 |-- month: string (nullable = true)
 |-- duration: string (nullable = true)
 |-- campaign: string (nullable = true)
 |-- pdays: string (nullable = true)
 |-- previous: string (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- y: string (nullable = true)
```

Step 2: Success rate

We need to calculate the total number of subscribed in records and total entries.

```
scala> val totalcount = mydf.count().toDouble
```

```
#output:
totalcount: Double = 45211.0
scala> val subscribed = mydf.filter($"y" ===
"yes").count().toDouble
#output:
subscribed = Double = 5289.0
scala> val success_rate = subscribed/totalcount
#output:
success_rate: Double = 0.1169
```

The success rate comes around 11.5%.

Step 3: The Failure rate

For this, we need to calculate the total number of not subscribed in records and total entries.

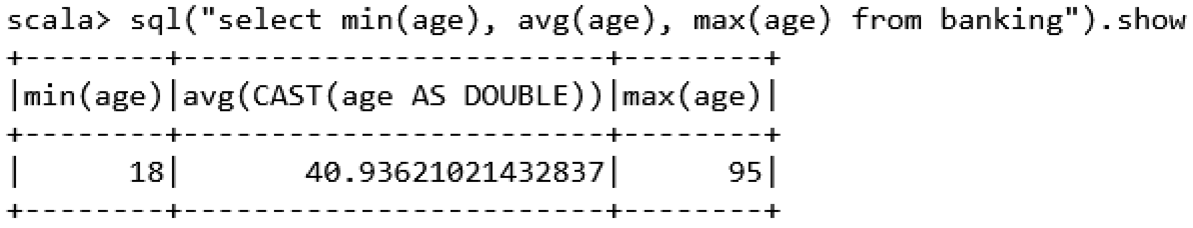
```
scala> val not_subscribed = mydf.filter($"n" ===
"no").count().toDouble
#output:
not_subscribed = Double = 39922.0
scala> val failure_rate = not_subscribed/totalcount
#output:
Failure_rate: Double = 0.8830
```

The failure rate comes around 88.3%.

Step 4: Find the minimum, maximum and average age of the people

Sometimes we use SQL operations, to make a data frame in form of a table we have to use the same method as shown below:

```
scala> mydf.createOrReplaceTempView("banking")scala> sql("select
min(age), avg(age), max(age) from banking").show
```

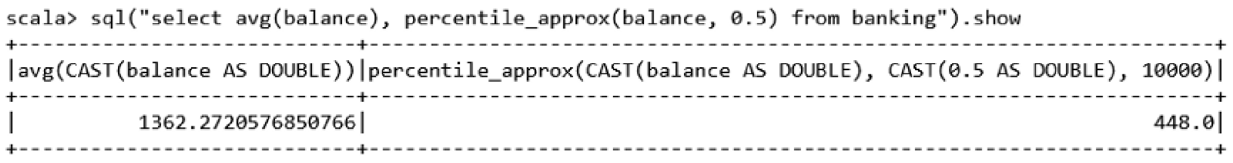


```
scala> sql("select min(age), avg(age), max(age) from banking").show
+--------+------------------------+--------+
|min(age)|avg(CAST(age AS DOUBLE))|max(age)|
+--------+------------------------+--------+
|      18|       40.93621021432837|      95|
+--------+------------------------+--------+
```

Step 5: To observe the customers through their bank balances

The analyst is always trying to find insights from different features so that they do analysis and visualize it. The analysis by checking the bank balances of the customers.

```
scala> sql("select avg(balance), percentile_approx(balance, 0.5)
from banking").show
```

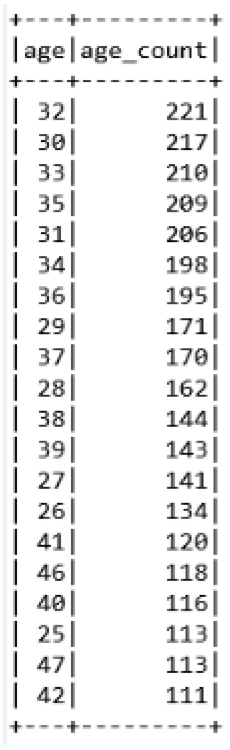


```
scala> sql("select avg(balance), percentile_approx(balance, 0.5) from banking").show
+----------------------------+----------------------------------------------------------------------+
|avg(CAST(balance AS DOUBLE))|percentile_approx(CAST(balance AS DOUBLE), CAST(0.5 AS DOUBLE), 10000)|
+----------------------------+----------------------------------------------------------------------+
|          1362.2720576850766|                                                                 448.0|
+----------------------------+----------------------------------------------------------------------+
```

Step 6: Find the number of people by age of customers who subscribed to the scheme

The result is showing only the top 20 rows.

```
scala> sql("select age, count(*) as age_count from banking where
y = 'yes' group by age order by age_count desc").show
```



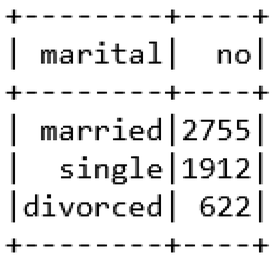
```
+---+---------+
|age|age_count|
+---+---------+
| 32|      221|
| 30|      217|
| 33|      210|
| 35|      209|
| 31|      206|
| 34|      198|
| 36|      195|
| 29|      171|
| 37|      170|
| 28|      162|
| 38|      144|
| 39|      143|
| 27|      141|
| 26|      134|
| 41|      120|
| 46|      118|
| 40|      116|
| 25|      113|
| 47|      113|
| 42|      111|
+---+---------+
```

Step 7: Know the marital status count categories

This query in scala gives the count of all categories in marital status.

```
scala> sql("select marital, count(*) as count from banking where

y = 'yes' group by marital order by count desc").show
```



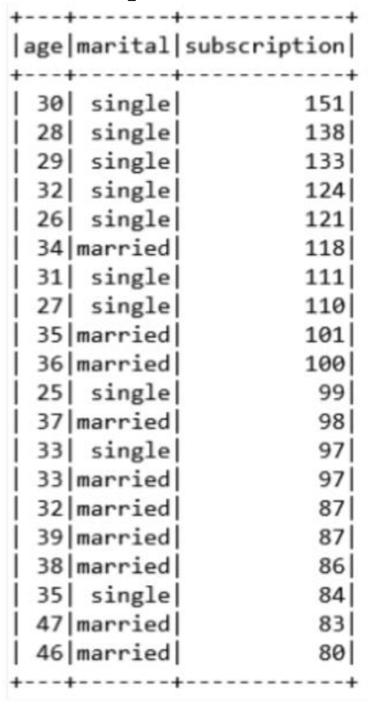
```
+--------+----+
| marital|  no|
+--------+----+
| married|2755|
|  single|1912|
|divorced| 622|
+--------+----+
```

Step 8: Effect of age and marital status together

How a combination of age and marital status gives insight for subscription or not.

```
scala> sql("select age, marital, count(*) as subscription from

banking where y ='yes' group by age, marital order by
subscription desc").show
```

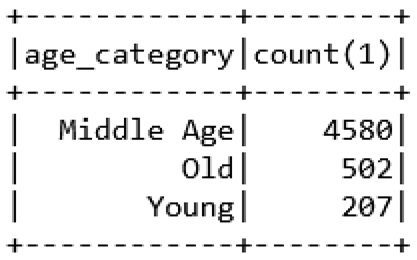


```
+---+-------+------------+
|age|marital|subscription|
+---+-------+------------+
| 30| single|         151|
| 28| single|         138|
| 29| single|         133|
| 32| single|         124|
| 26| single|         121|
| 34|married|         118|
| 31| single|         111|
| 27| single|         110|
| 35|married|         101|
| 36|married|         100|
| 25| single|          99|
| 37|married|          98|
| 33| single|          97|
| 33|married|          97|
| 32|married|          87|
| 39|married|          87|
| 38|married|          86|
| 35| single|          84|
| 47|married|          83|
| 46|married|          80|
+---+-------+------------+
```

Step 9: A little feature engineering to find the right age for subscription

To know the right age of the customers who subscribed and find the age group who subscribed the more.

```
scala> sql("select case when age<25 then 'Young' when age
between 25 and 60 then 'Middle age' when age >=60 then 'Old' end
as age_category,count(1) from banking where y='yes' group by
           age_category order by 2 desc").show
```



```
+------------+--------+
|age_category|count(1)|
+------------+--------+
|  Middle Age|    4580|
|         Old|     502|
|       Young|     207|
+------------+--------+
```