ADVANCED SPARK ASSIGNMENT 21.2

PROBLEM STATEMENT:

1) Join of two or more data sets is one of the most widely used operations you do with your data, but in distributed systems it can be a huge headache. In general, since your data are distributed among many nodes, they have to be shuffled before a join that causes significant network I/O and slow performance.
2) Fortunately, if you need to join a large table with relatively small tables you can avoid sending all data of the large table over the network. This type of join is called map-side join in Hadoop community. In other distributed systems, it is often called replicated or broadcast join. The fact table can be very large, while dimension tables are often quite small.
3) Let's use the following sample data (one fact and two-dimension tables):

```
// Fact table
val flights = sc.parallelize(List(
("SEA", "JFK", "DL", "418", "7:00"),
("SFO", "LAX", "AA", "1250", "7:05"),
("SFO", "JFK", "VX", "12", "7:05"),
("JFK", "LAX", "DL", "424", "7:10"),
("LAX", "SEA", "DL", "5737", "7:10")
)
)

// Dimension table/broadcast table
val airports = sc.parallelize(List(
("JFK", "John F. Kennedy International Airport", "New York", "NY"),
("LAX", "Los Angeles International Airport", "Los Angeles", "CA"),
("SEA", "Seattle-Tacoma International Airport", "Seattle", "WA"),
("SFO", "San Francisco International Airport", "San Francisco", "CA")))

// Dimension table/broadcast table
val airlines = sc.parallelize(List(
("AA", "American Airlines"),
("DL", "Delta Airlines"),
("VX", "Virgin America")))
```

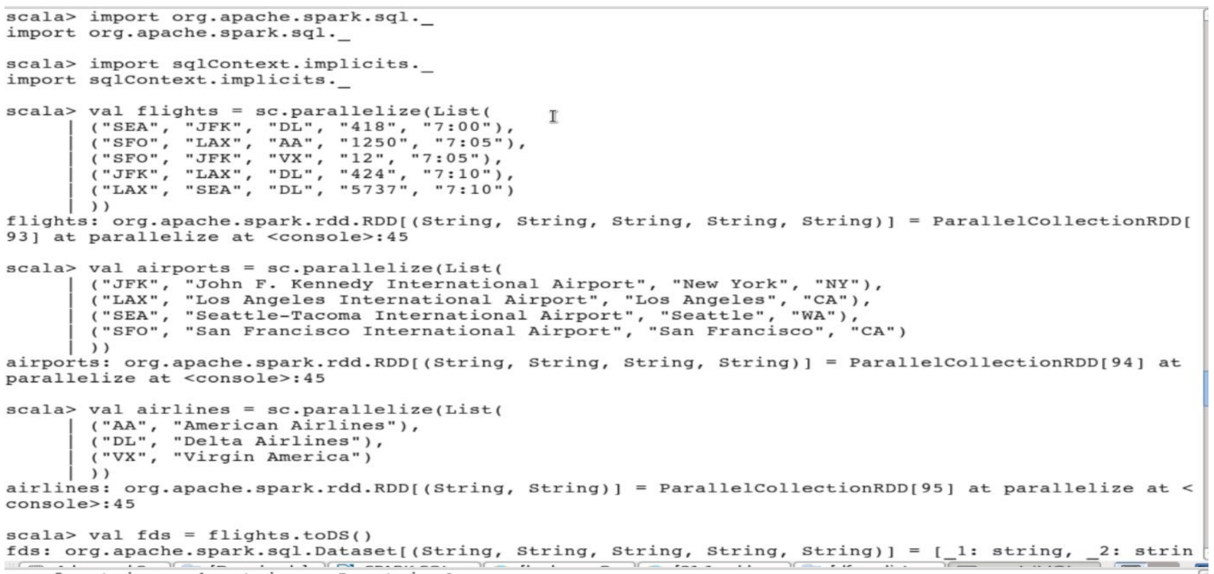
We need to join the fact and dimension tables to get the following result:

```
San Francisco   Los Angeles    American Airlines 1250  7:05
New York        Los Angeles    Delta Airlines    424   7:10
Seattle         New York       Delta Airlines    418   7:00
San Francisco   New York       Virgin America    12    7:05
Los Angeles     Seattle        Delta Airlines    5737  7:10
```

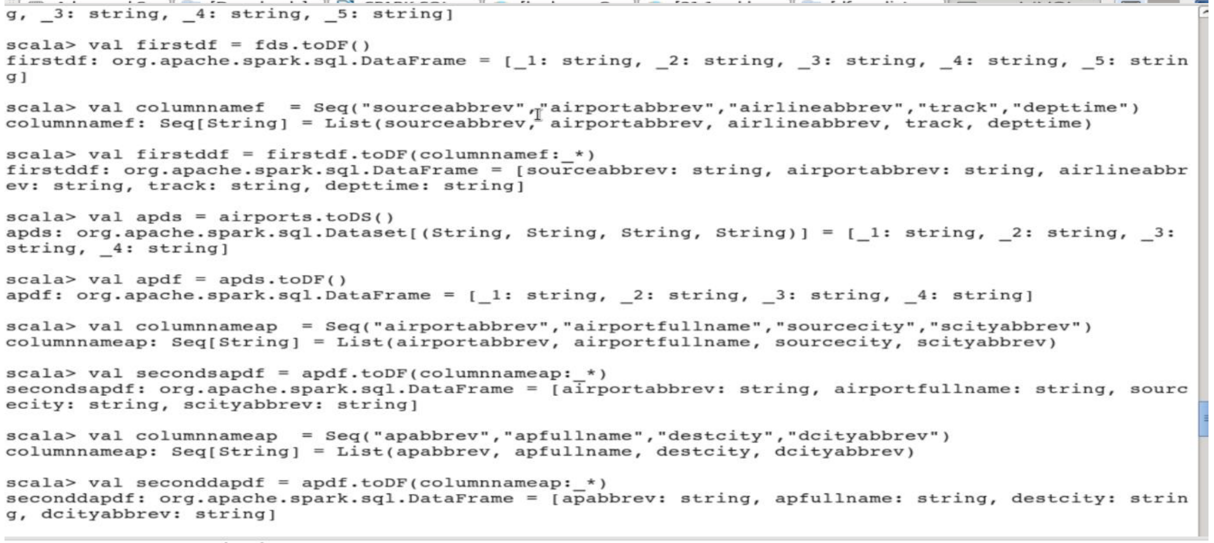
```
//Importing the spark sql packages
import org.apache.spark.sql._
import sqlContext.implicits._

//putting the flights data into SPARK RDD
val flights = sc.parallelize(List(
("SEA", "JFK", "DL", "418", "7:00"),
("SFO", "LAX", "AA", "1250", "7:05"),
("SFO", "JFK", "VX", "12", "7:05"),
("JFK", "LAX", "DL", "424", "7:10"),
("LAX", "SEA", "DL", "5737", "7:10")
))

//putting the airports data into SPARK RDD
val airports = sc.parallelize(List(
("JFK", "John F. Kennedy International Airport", "New York", "NY"),
("LAX", "Los Angeles International Airport", "Los Angeles", "CA"),
("SEA", "Seattle-Tacoma International Airport", "Seattle", "WA"),
("SFO", "San Francisco International Airport", "San Francisco", "CA")
))

//putting the airlines data into SPARK RDD
val airlines = sc.parallelize(List(
("AA", "American Airlines"),
("DL", "Delta Airlines"),
("VX", "Virgin America")
))
---------------------------------------------------------------------------

//Conversion of flights RDD into Dataset
val fds = flights.toDS()

//Conversion of Dataset to SPARK Dataframe
val firstdf = fds.toDF()

//Naming the Columns of SPARK Dataframe into putting it into RDD
val columnnamef  =
Seq("sourceabbrev","airportabbrev","airlineabbrev","track","depttime")

//Renaming the columns of SPARK Dataframe with the help of SPARK RDD passed
as an argument
val firstddf = firstdf.toDF(columnnamef:_*)
---------------------------------------------------------------------------

//Conversion of airports RDD into Dataset
val apds = airports.toDS()

//Conversion of airport Dataset to SPARK Dataframe
val apdf = apds.toDF()

//Naming the Columns of SPARK Dataframe into RDD
val columnnameap  =
Seq("airportabbrev","airportfullname","sourcecity","scityabbrev")

//Renaming the columns of SPARK Dataframe with the help of SPARK RDD passed
as an argument
val secondsapdf = apdf.toDF(columnnameap:_*)

//Naming the Columns of SPARK Dataframe into RDD
val columnnameap  = Seq("apabbrev","apfullname","destcity","dcityabbrev")

//Renaming the columns of SPARK Dataframe with the help of SPARK RDD
val seconddapdf = apdf.toDF(columnnameap:_*)
---------------------------------------------------------------------------

//Conversion of airlines RDD into Dataset
val asds = airlines.toDS()
```

```
//Conversion of airlines Dataset to SPARK Dataframe
val aldf = asds.toDF()

//Naming the Columns of SPARK Dataframe into RDD
val columnnameal  = Seq("airlineabbrev","airlinefullname")

//Renaming the columns of SPARK Dataframe with the help of SPARK RDD
val secondaldf = aldf.toDF(columnnameal:_*)
---------------------------------------------------------------------------

//Broadcast Join(Inner Join in this scenario) of fact table flight and
broadcast table airports for destination city
val
firstjointdf=firstddf.join(broadcast(seconddapdf),firstddf("airportabbrev")
=== seconddapdf("apabbrev"),"inner")

//Display the Schema of Spark Dataframe generated broadcast join
firstjointdf.printSchema()

//Broadcast Join(Inner Join in this scenario) of fact table flight and
broadcast table airports for source city

val
secondjointdf=firstjointdf.join(broadcast(secondsapdf),firstjointdf("source
abbrev") === secondsapdf("airportabbrev"),"inner")

//Display the Schema of Spark Dataframe generated broadcast join
secondjointdf.printSchema()

//Broadcast Join(Inner Join in this scenario) of fact table flight and
broadcast table airlines
val thirdjointdf =
secondjointdf.join(broadcast(secondaldf),secondjointdf("airlineabbrev") ===
secondaldf("airlineabbrev"),"inner")

//Display the Schema of Spark Dataframe generated broadcast join
thirdjointdf.printSchema()

//Display the final output of broadcast join of fact table and dimension
tables
thirdjointdf.select(
$"sourcecity",
$"destcity",
$"airlinefullname",
$"track",
$"depttime"
).show()
```

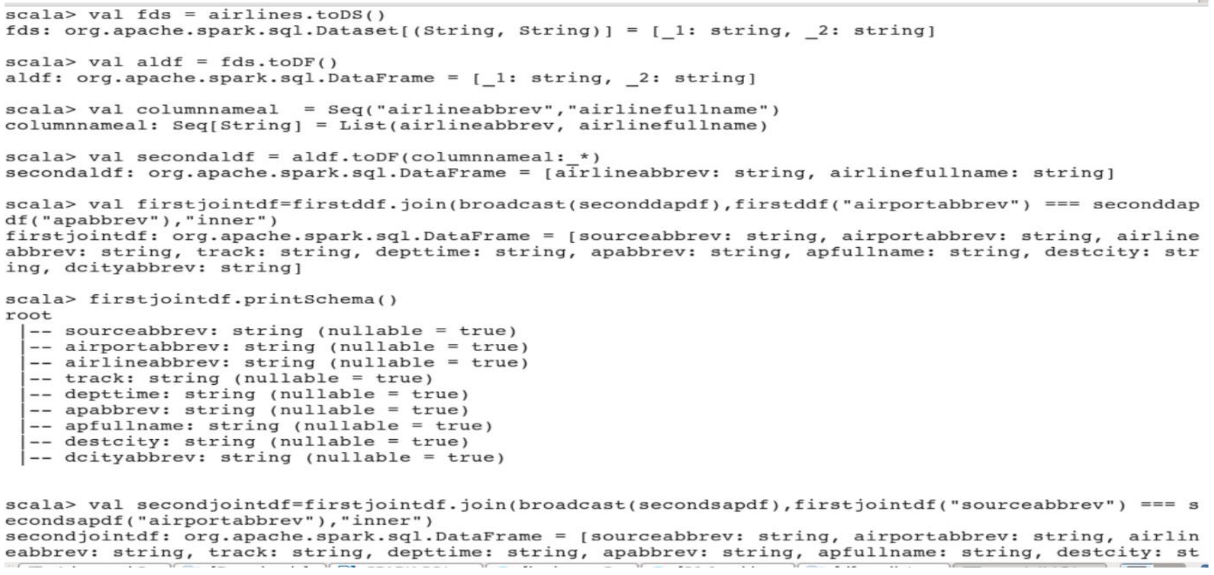
Screenshots:



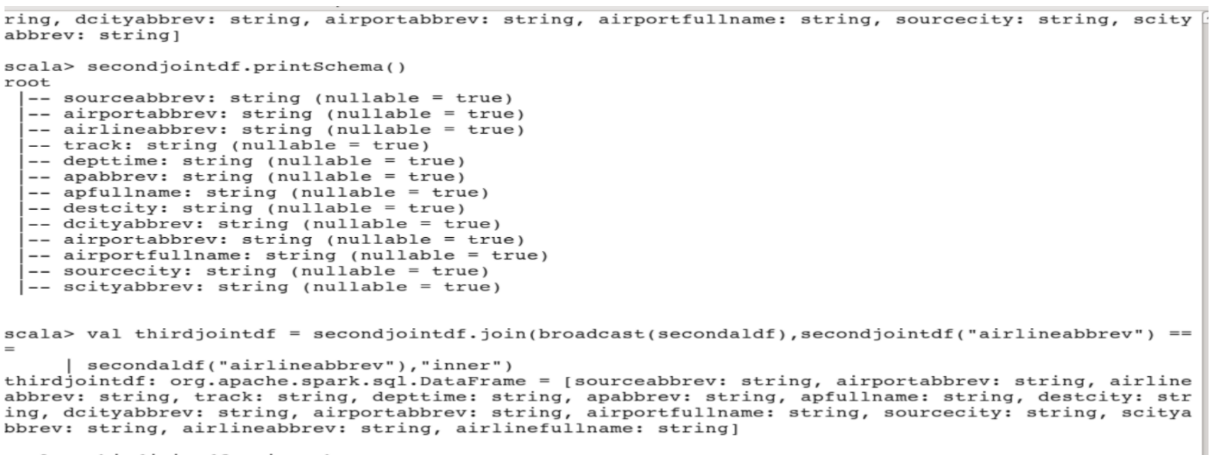
```
scala> import org.apache.spark.sql._
import org.apache.spark.sql._

scala> import sqlContext.implicits._
import sqlContext.implicits._

scala> val flights = sc.parallelize(List(
     | ("SEA", "JFK", "DL", "418", "7:00"),
     | ("SFO", "LAX", "AA", "1250", "7:05"),
     | ("SFO", "JFK", "VX", "12", "7:05"),
     | ("JFK", "LAX", "DL", "424", "7:10"),
     | ("LAX", "SEA", "DL", "5737", "7:10")
     | ))
flights: org.apache.spark.rdd.RDD[(String, String, String, String, String)] = ParallelCollectionRDD[
93] at parallelize at <console>:45

scala> val airports = sc.parallelize(List(
     | ("JFK", "John F. Kennedy International Airport", "New York", "NY"),
     | ("LAX", "Los Angeles International Airport", "Los Angeles", "CA"),
     | ("SEA", "Seattle-Tacoma International Airport", "Seattle", "WA"),
     | ("SFO", "San Francisco International Airport", "San Francisco", "CA")
     | ))
airports: org.apache.spark.rdd.RDD[(String, String, String, String)] = ParallelCollectionRDD[94] at
parallelize at <console>:45

scala> val airlines = sc.parallelize(List(
     | ("AA", "American Airlines"),
     | ("DL", "Delta Airlines"),
     | ("VX", "Virgin America")
     | ))
airlines: org.apache.spark.rdd.RDD[(String, String)] = ParallelCollectionRDD[95] at parallelize at <
console>:45

scala> val fds = flights.toDS()
fds: org.apache.spark.sql.Dataset[(String, String, String, String, String)] = [_1: string, _2: strin
```



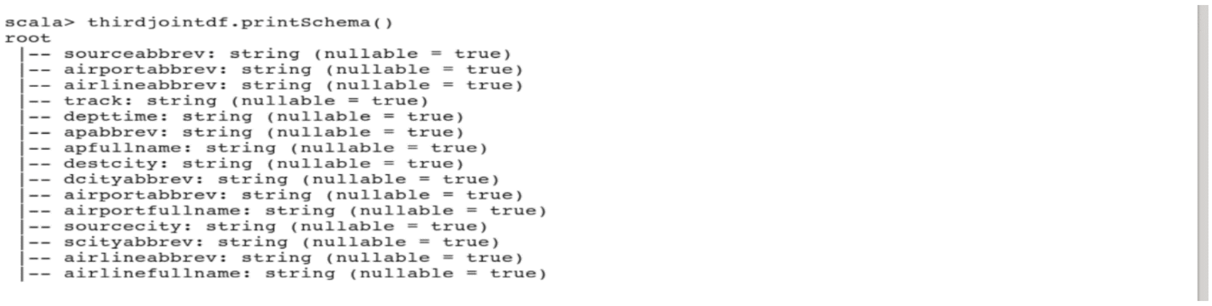
```
g, _3: string, _4: string, _5: string]

scala> val firstdf = fds.toDF()
firstdf: org.apache.spark.sql.DataFrame = [_1: string, _2: string, _3: string, _4: string, _5: strin
g]

scala> val columnnamef  = Seq("sourceabbrev","airportabbrev","airlineabbrev","track","depttime")
columnnamef: Seq[String] = List(sourceabbrev, airportabbrev, airlineabbrev, track, depttime)

scala> val firstddf = firstdf.toDF(columnnamef:_*)
firstddf: org.apache.spark.sql.DataFrame = [sourceabbrev: string, airportabbrev: string, airlineabbr
ev: string, track: string, depttime: string]

scala> val apds = airports.toDS()
apds: org.apache.spark.sql.Dataset[(String, String, String, String)] = [_1: string, _2: string, _3:
string, _4: string]

scala> val apdf = apds.toDF()
apdf: org.apache.spark.sql.DataFrame = [_1: string, _2: string, _3: string, _4: string]

scala> val columnnameap  = Seq("airportabbrev","airportfullname","sourcecity","scityabbrev")
columnnameap: Seq[String] = List(airportabbrev, airportfullname, sourcecity, scityabbrev)

scala> val secondsapdf = apdf.toDF(columnnameap:_*)
secondsapdf: org.apache.spark.sql.DataFrame = [airportabbrev: string, airportfullname: string, sourc
ecity: string, scityabbrev: string]

scala> val columnnameap  = Seq("apabbrev","apfullname","destcity","dcityabbrev")
columnnameap: Seq[String] = List(apabbrev, apfullname, destcity, dcityabbrev)

scala> val seconddapdf = apdf.toDF(columnnameap:_*)
seconddapdf: org.apache.spark.sql.DataFrame = [apabbrev: string, apfullname: string, destcity: strin
g, dcityabbrev: string]
```



```
scala> val fds = airlines.toDS()
fds: org.apache.spark.sql.Dataset[(String, String)] = [_1: string, _2: string]

scala> val aldf = fds.toDF()
aldf: org.apache.spark.sql.DataFrame = [_1: string, _2: string]

scala> val columnnameal  = Seq("airlineabbrev","airlinefullname")
columnnameal: Seq[String] = List(airlineabbrev, airlinefullname)

scala> val secondaldf = aldf.toDF(columnnameal:_*)
secondaldf: org.apache.spark.sql.DataFrame = [airlineabbrev: string, airlinefullname: string]

scala> val firstjointdf=firstddf.join(broadcast(seconddapdf),firstddf("airportabbrev") === seconddap
df("apabbrev"),"inner")
firstjointdf: org.apache.spark.sql.DataFrame = [sourceabbrev: string, airportabbrev: string, airline
abbrev: string, track: string, depttime: string, apabbrev: string, apfullname: string, destcity: str
ing, dcityabbrev: string]

scala> firstjointdf.printSchema()
root
 |-- sourceabbrev: string (nullable = true)
 |-- airportabbrev: string (nullable = true)
 |-- airlineabbrev: string (nullable = true)
 |-- track: string (nullable = true)
 |-- depttime: string (nullable = true)
 |-- apabbrev: string (nullable = true)
 |-- apfullname: string (nullable = true)
 |-- destcity: string (nullable = true)
 |-- dcityabbrev: string (nullable = true)


scala> val secondjointdf=firstjointdf.join(broadcast(secondsapdf),firstjointdf("sourceabbrev") === s
econdsapdf("airportabbrev"),"inner")
secondjointdf: org.apache.spark.sql.DataFrame = [sourceabbrev: string, airportabbrev: string, airlin
eabbrev: string, track: string, depttime: string, apabbrev: string, apfullname: string, destcity: st
```



```
ring, dcityabbrev: string, airportabbrev: string, airportfullname: string, sourcecity: string, scity
abbrev: string]

scala> secondjointdf.printSchema()
root
 |-- sourceabbrev: string (nullable = true)
 |-- airportabbrev: string (nullable = true)
 |-- airlineabbrev: string (nullable = true)
 |-- track: string (nullable = true)
 |-- depttime: string (nullable = true)
 |-- apabbrev: string (nullable = true)
 |-- apfullname: string (nullable = true)
 |-- destcity: string (nullable = true)
 |-- dcityabbrev: string (nullable = true)
 |-- airportabbrev: string (nullable = true)
 |-- airportfullname: string (nullable = true)
 |-- sourcecity: string (nullable = true)
 |-- scityabbrev: string (nullable = true)


scala> val thirdjointdf = secondjointdf.join(broadcast(secondaldf),secondjointdf("airlineabbrev") ==
=
     | secondaldf("airlineabbrev"),"inner")
thirdjointdf: org.apache.spark.sql.DataFrame = [sourceabbrev: string, airportabbrev: string, airline
abbrev: string, track: string, depttime: string, apabbrev: string, apfullname: string, destcity: str
ing, dcityabbrev: string, airportabbrev: string, airportfullname: string, sourcecity: string, scitya
bbrev: string, airlineabbrev: string, airlinefullname: string]
```



```
scala> thirdjointdf.printSchema()
root
 |-- sourceabbrev: string (nullable = true)
 |-- airportabbrev: string (nullable = true)
 |-- airlineabbrev: string (nullable = true)
 |-- track: string (nullable = true)
 |-- depttime: string (nullable = true)
 |-- apabbrev: string (nullable = true)
 |-- apfullname: string (nullable = true)
 |-- destcity: string (nullable = true)
 |-- dcityabbrev: string (nullable = true)
 |-- airportabbrev: string (nullable = true)
 |-- airportfullname: string (nullable = true)
 |-- sourcecity: string (nullable = true)
 |-- scityabbrev: string (nullable = true)
 |-- airlineabbrev: string (nullable = true)
 |-- airlinefullname: string (nullable = true)
```

Final Output:

```
scala> thirdjointdf.select(
     | $"sourcecity",
     | $"destcity",
     | $"airlinefullname",
     | $"track",
     | $"depttime"
     | ).show()
+-------------+-----------+-----------------+-----+--------+
|   sourcecity|   destcity|  airlinefullname|track|depttime|
+-------------+-----------+-----------------+-----+--------+
|  Los Angeles|    Seattle|   Delta Airlines| 5737|    7:10|
|     New York|Los Angeles|   Delta Airlines|  424|    7:10|
|      Seattle|   New York|   Delta Airlines|  418|    7:00|
|San Francisco|Los Angeles|American Airlines| 1250|    7:05|
|San Francisco|   New York|   Virgin America|   12|    7:05|
+-------------+-----------+-----------------+-----+--------+
```