**Assignment 21.2 Spark**

**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.

Let’s use the following sample data (one fact and two dimension tables):

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

**Solution:**

**// 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"),

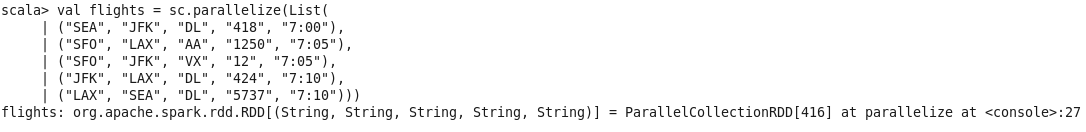
("JFK", "LAX", "DL", "424", "7:10"),

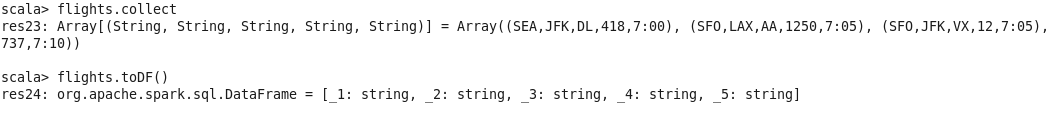
("LAX", "SEA", "DL", "5737", "7:10")))

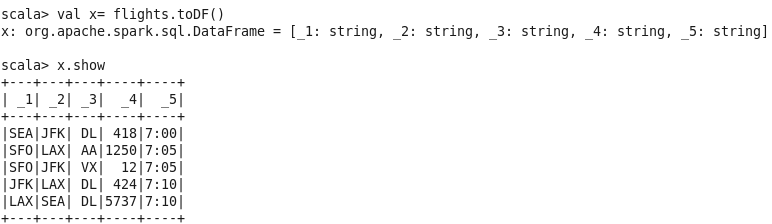
val x= flights.toDF("From\_AP","To\_AP","Airlines","Flightno","flighttime")

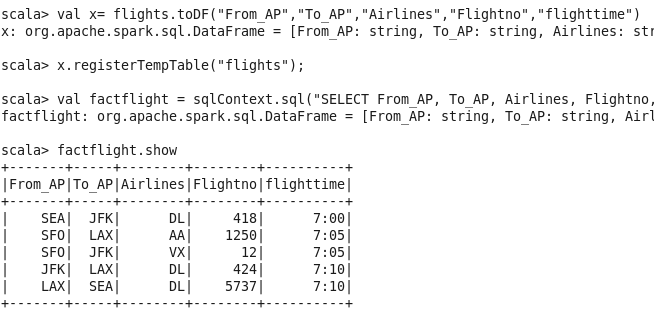
x.registerTempTable("flights");

val factflight = sqlContext.sql("SELECT From\_AP, To\_AP, Airlines, Flightno, flighttime FROM flights ")









**// Dimension table**

val airports = sc.parallelize(List(

("JFK", "John F. Kennedy International Airport", "New York", "NY"),

("LAX", "Los Angeles International Airport", "Los Angeles", "CA"),

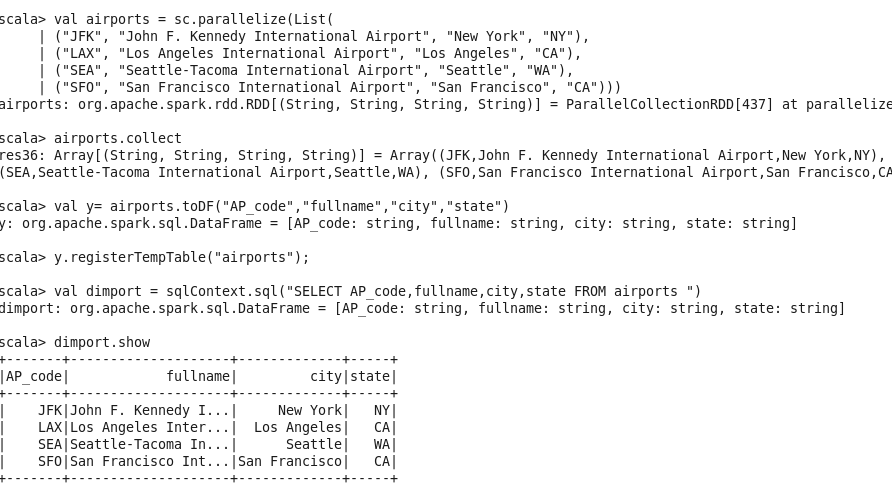
("SEA", "Seattle-Tacoma International Airport", "Seattle", "WA"),

("SFO", "San Francisco International Airport", "San Francisco", "CA")))

val y= airports.toDF("AP\_code","fullname","city","state")

y.registerTempTable("airports");

val dimport = sqlContext.sql("SELECT AP\_code,fullname,city,state FROM airports ")



**// Dimension table**

val airlines = sc.parallelize(List(

("AA", "American Airlines"),

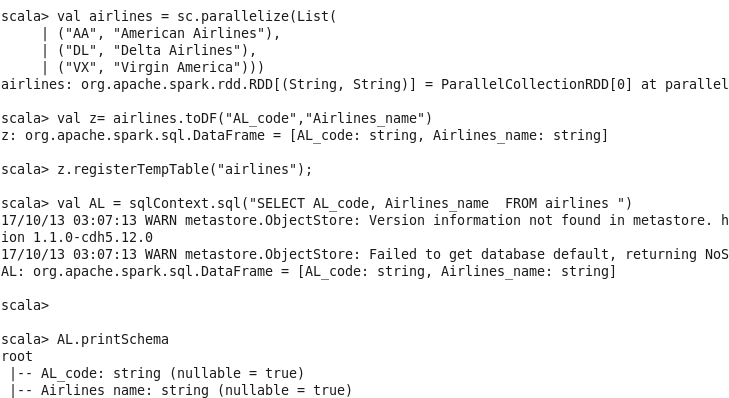
("DL", "Delta Airlines"),

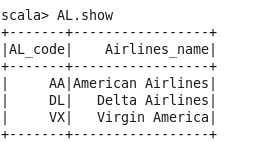
("VX", "Virgin America")))

val z= airlines.toDF("AL\_code","Airlines\_name")

z.registerTempTable("airlines");

val AL = sqlContext.sql("SELECT AL\_code, Airlines\_name FROM airlines ")





**Problem Statement:**

**PS-1:**

**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**

**New York Los Angeles Delta Airlines 424 7:10**

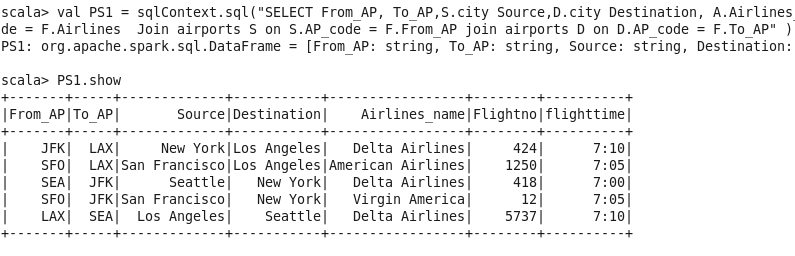
**Los Angeles Seattle Delta Airlines 5737 7:10**

**AL, factflight, dimport**

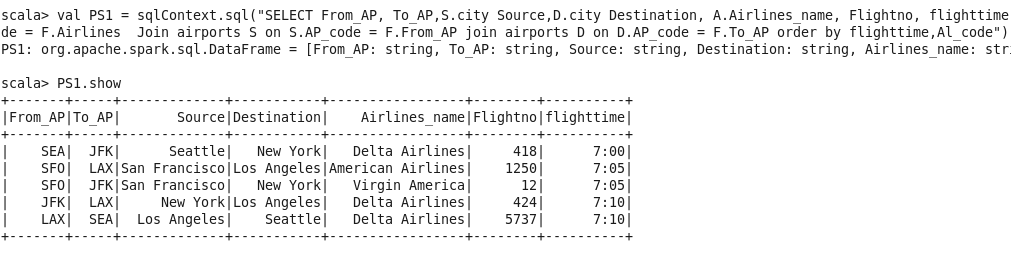
val AL = sqlContext.sql("SELECT AL\_code, Airlines\_name FROM airlines ")

val dimport = sqlContext.sql("SELECT AP\_code,fullname,city,state FROM airports ")

scala> val PS1 = sqlContext.sql("SELECT From\_AP, To\_AP,S.city Source,D.city Destination, A.Airlines\_name, Flightno, flighttime FROM flights F Join airlines A on A.AL\_code = F.Airlines Join airports S on S.AP\_code = F.From\_AP join airports D on D.AP\_code = F.To\_AP" )



val PS1 = sqlContext.sql("SELECT From\_AP, To\_AP,S.city Source,D.city Destination, A.Airlines\_name, Flightno, flighttime FROM flights F Join airlines A on A.AL\_code = F.Airlines Join airports S on S.AP\_code = F.From\_AP join airports D on D.AP\_code = F.To\_AP order by flighttime,Al\_code")



val PS1 = sqlContext.sql("SELECT From\_AP, To\_AP,S.city Source,D.city Destination, A.Airlines\_name, Flightno, flighttime FROM flights F Join airlines A on A.AL\_code = F.Airlines Join airports S on S.AP\_code = F.From\_AP join airports D on D.AP\_code = F.To\_AP WHERE S.city='Seattle' and D.city='New York' and Flightno='418' " )

