**ASSIGNMENT 18.3**

**1) Considering age groups of < 20 , 20-35, 35 > ,Which age group spends the most**

**amount of money travelling.**

**val baseRDD1 = sc.textFile("/home/acadgild/Assignment-18/S18\_Dataset\_Holidays.txt")**

**val baseRDD2 = sc.textFile("/home/acadgild/Assignment-18/S18\_Dataset\_Transport.txt")**

**val baseRDD3 = sc.textFile("/home/acadgild/Assignment-18/S18\_Dataset\_User\_details.txt")**

**import org.apache.spark.storage.StorageLevel**

**baseRDD1.persist(StorageLevel.MEMORY\_ONLY)**

**baseRDD2.persist(StorageLevel.MEMORY\_ONLY)**

**baseRDD3.persist(StorageLevel.MEMORY\_ONLY)**

**val travel = baseRDD1.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2),x.split(",")(3),x.split(",")(4).toInt,x.split(",")(5).toInt))**

**val transport = baseRDD2.map(x => (x.split(",")(0),x.split(",")(1).toInt))**

**val user = baseRDD3.map(x => (x.split(",")(0).toInt,x.split(",")(1),x.split(",")(2).toInt))**

**val AgeMap = user.map(x => x.\_1 ->**

**| {**

**| if(x.\_3<20)**

**| "20"**

**| else if(x.\_3>35)**

**| "35"**

**| else "20-35"**

**| })**

**val userMap = travel.map(x => x.\_4 -> (x.\_1,x.\_5))**

**val transportmap = transport.map(x=> x.\_1 -> x.\_2)**

**val joinCost = userMap.join(transportmap)**

**val calCost = joinCost.map(x => x.\_2.\_1.\_1 -> x.\_2.\_1.\_2 \* x.\_2.\_2)**

**val groupCost = calCost.groupByKey().map(x => x.\_1 -> x.\_2.sum)**

**val groupAgeCost = AgeMap.join(groupCost).map(x => x.\_2.\_1 -> x.\_2.\_2)**

**val GroupSpend = groupAgeCost.group**

**val finalCost = groupAgeCost.groupByKey().map(x => x.\_1 -> x.\_2.sum)**

**val maxVal = finalCost.sortBy(x => -x.\_2).first()**

**2) What is the amount spent by each age-group, every year in travelling?**

**val UserYearMap = travel.map(x => x.\_4 -> (x.\_1,x.\_5,x.\_6))**

**val transportmap = transport.map(x=> x.\_1 -> x.\_2)**

**val UserCost = UserYearMap.join(transportmap)**

**val CalcCost = UserCost.map(x => x.\_2.\_1.\_1 -> (x.\_2.\_1.\_3,x.\_2.\_1.\_2 \* x.\_2.\_2))**

**val AgeMap = user.map(x => x.\_1 ->**

**| {**

**| if(x.\_3<20)**

**| "20"**

**| else if(x.\_3>35)**

**| "35"**

**| else "20-35"**

**| })**

**val CostMap = AgeMap.join(CalcCost).map(x => (x.\_2.\_1,x.\_2.\_2.\_1) -> x.\_2.\_2.\_2)**

**val ExpPeryear = CostMap.groupByKey().map(x => x.\_1 -> x.\_2.sum)**