Session 20:
SPARK SQL 1
Assignment 1

Input Files:

- S20_Dataset_Holidays.txt
- S20_Dataset_Transport.txt
- S20_Dataset_User_details.txt

Program:

Task2_03.scala | Task1.scala

```scala
package com.spark.assignment20

import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.log4j._
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.types._
import org.apache.spark.sql.types.{StructType, StructField, StringType}
import org.apache.spark.sql.Row

object Task1 {
  def main(args : Array[String]) : Unit = {
    Logger.getLogger("org").setLevel(Level.ERROR)

    val spark = SparkSession.builder.appName("Students").master("local").getOrCreate()
    import spark.implicits._

    val rddTransport = spark.sparkContext.textFile("E:/Acadgild/Data/S20_Dataset_Transport.txt")
    val rddUsers = spark.sparkContext.textFile("E:/Acadgild/Data/S20_Dataset_User_details.txt")
    val rddHolidays = spark.sparkContext.textFile("E:/Acadgild/Data/S20_Dataset_Holidays.txt")

    val headerUser = "id name age"
    val headerTransport = "transport_mode cost_per_unit"
    val headerHolidays = "id source destination transport_mode distance year"

    val schemaUser = StructType(headerUser.split(" ").map(fieldName => StructField(fieldName,StringType, true)))
    val schemaTransport = StructType(headerTransport.split(" ").map(fieldName => StructField(fieldName,StringType, true)))
    val schemaHolidays = StructType(headerHolidays.split(" ").map(fieldName => StructField(fieldName,StringType, true)))

    val rowUserDD = rddUsers.map(_.split(",")).map(x => Row(x(0), x(1), x(2)))
    val rowTransportDD = rddTransport.map(_.split(",")).map(x => Row(x(0), x(1)))
    val rowHolidaysDD = rddHolidays.map(_.split(",")).map(x => Row(x(0), x(1), x(2), x(3), x(4), x(5)))
```

```scala
    val userDF = spark.createDataFrame(rowUserDD, schemaUser)
    userDF.createOrReplaceTempView("users")

    val transportDF = spark.createDataFrame(rowTransportDD, schemaTransport)
    transportDF.createOrReplaceTempView("transport")

    val holidaysDF = spark.createDataFrame(rowHolidaysDD, schemaHolidays)
    holidaysDF.createOrReplaceTempView("holidays")

    // 1. What is the distribution of the total number of air-travelers per year
    println("1 --> DISTRIBUTION OF TOTAL NUMBER OF AIR TRAVELERS PER YEAR")
    val travellerDistribution = spark.sql("SELECT year, count(*) as Count FROM holidays GROUP BY year")
    travellerDistribution.show()

    // 2. What is the total air distance covered by each user per year
    println("2 --> TOTAL AIR DISTANCE COVERED BY EACH USER PER YEAR")
    val userHolidayData = spark.sql("SELECT u.name, h.year, SUM(h.distance) AS air_distance FROM users AS u INNER JOIN holidays AS h ON u.id = h.id GROUP BY u.name, h.year")
    userHolidayData.collect().foreach(println)

    // 3. Which user has travelled the largest distance till date
    println("3 --> USER WHO HAS TRAVELLED THE LARGEST DISTANCE TILL DATE")
    val userMaxAirDistance = spark.sql("SELECT u.name, SUM(h.distance) as air_distance FROM users AS u INNER JOIN holidays AS h ON u.id = h.id GROUP BY u.name ORDER BY air_di
    println(userMaxAirDistance.first())

    // 4. What is the most preferred destination for all users.
    println("4 --> MOST PREFERRED DESTINATIONS FOR ALL USERS")
    val groupedDestinations = spark.sql("SELECT destination, COUNT(destination) AS visit_count FROM holidays GROUP BY destination ORDER BY visit_count DESC")
    println("Visit Count by Destinations\n----------------------------")
    groupedDestinations.show()
    println("Most preferred destination is : " + groupedDestinations.first().get(0))
```

```
    // 5. Which route is generating the most revenue per year
    println("5 --> ROUTE GENERATING THE MOST REVENUE PER YEAR")
    val groupedRouteRevenue = spark.sql("SELECT h.source, h.destination, h.year, SUM((h.distance * t.cost_per_unit)) as yearly_revenue FROM holidays AS h INNER JOIN transport AS t C
    groupedRouteRevenue.show()

    // 6. What is the total amount spent by every user on air-travel per year
    println("6 --> TOTAL AMOUNT SPENT BY EVERY USER ON AIR PER YEAR")
    val userExpenditure = spark.sql("SELECT u.name, h.year, SUM(h.distance * t.cost_per_unit) as total_expense FROM users AS u INNER JOIN holidays AS h ON u.id = h.id INNER JOIN t
    userExpenditure.show()

    // 7. Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.
    println("7 --> AGE GROUP THAT TRAVELS MOST EVERY YEAR")
    val maxDistanceTravelled = spark.sql("SELECT h.year, CASE WHEN u.age < 20 THEN \"BELOW 20\" WHEN u.age > 35 THEN \"ABOVE 35\" ELSE \"BETWEEN 20 & 35\" END AS age_g
    maxDistanceTravelled.show()

    spark.stop()
  }
}
```

Execution Result:

Task – 1:

```
1 --> DISTRIBUTION OF TOTAL NUMBER OF AIR TRAVELERS PER YEAR
+----+-----+
|year|Count|
+----+-----+
|1992|    7|
|1994|    1|
|1993|    7|
|1990|    8|
|1991|    9|
+----+-----+
```

Task – 2:

```
2 --> TOTAL AIR DISTANCE COVERED BY EACH USER PER YEAR
[mark,1994,200.0]
[andrew,1990,200.0]
[annie,1993,200.0]
[lisa,1991,200.0]
[andrew,1992,200.0]
[annie,1990,200.0]
[john,1993,200.0]
[peter,1993,200.0]
[mark,1990,200.0]
[lisa,1990,400.0]
[annie,1992,200.0]
[mark,1993,600.0]
[james,1990,600.0]
[luke,1991,200.0]
[peter,1991,400.0]
[thomas,1991,200.0]
[luke,1993,200.0]
[john,1991,400.0]
[luke,1992,200.0]
[thomas,1992,400.0]
[mark,1991,200.0]
[mark,1992,400.0]
[andrew,1991,200.0]
```

Task – 3 & Task 4:

```
3 --> USER WHO HAS TRAVELLED THE LARGEST DISTANCE TILL DATE
[mark,1600.0]
4 --> MOST PREFERRED DESTINATIONS FOR ALL USERS
Visit Count by Destinations
-----------------------------
+-----------+-----------+
|destination|visit_count|
+-----------+-----------+
|        IND|          9|
|        CHN|          7|
|        RUS|          6|
|        AUS|          5|
|        PAK|          5|
+-----------+-----------+

Most preferred destination is : IND
```

Task – 5:

```
5 --> ROUTE GENERATING THE MOST REVENUE PER YEAR
+------+-----------+----+--------------+
|source|destination|year|yearly_revenue|
+------+-----------+----+--------------+
|   CHN|        IND|1990|       68000.0|
|   RUS|        IND|1992|       68000.0|
|   AUS|        CHN|1993|       68000.0|
|   CHN|        RUS|1992|       68000.0|
|   IND|        AUS|1991|       68000.0|
|   CHN|        IND|1993|       68000.0|
|   IND|        RUS|1991|       68000.0|
|   IND|        CHN|1992|       34000.0|
|   RUS|        AUS|1990|       34000.0|
|   CHN|        RUS|1990|       34000.0|
|   RUS|        CHN|1992|       34000.0|
|   IND|        PAK|1991|       34000.0|
|   IND|        CHN|1991|       34000.0|
|   CHN|        AUS|1990|       34000.0|
|   CHN|        PAK|1990|       34000.0|
|   PAK|        IND|1993|       34000.0|
|   CHN|        PAK|1991|       34000.0|
|   PAK|        AUS|1993|       34000.0|
|   RUS|        IND|1990|       34000.0|
|   RUS|        CHN|1993|       34000.0|
+------+-----------+----+--------------+
only showing top 20 rows
```

Task – 6:

```
6 --> TOTAL AMOUNT SPENT BY EVERY USER ON AIR PER YEAR
+------+----+-------------+
|  name|year|total_expense|
+------+----+-------------+
|  mark|1993|     102000.0|
| james|1990|     102000.0|
| peter|1991|      68000.0|
|  lisa|1990|      68000.0|
|  john|1991|      68000.0|
|thomas|1992|      68000.0|
|  mark|1992|      68000.0|
|andrew|1990|      34000.0|
|  lisa|1991|      34000.0|
| annie|1993|      34000.0|
| peter|1993|      34000.0|
|andrew|1992|      34000.0|
| annie|1990|      34000.0|
|  john|1993|      34000.0|
|thomas|1991|      34000.0|
|  mark|1990|      34000.0|
|  mark|1991|      34000.0|
| annie|1992|      34000.0|
|  luke|1991|      34000.0|
|  luke|1993|      34000.0|
+------+----+-------------+
only showing top 20 rows
```

Task – 7:

```
7 --> AGE GROUP THAT TRAVELS MOST EVERY YEAR
+----+---------------+------------------+
|year|      age_group|travelled_distance|
+----+---------------+------------------+
|1990|BETWEEN 20 & 35|            1000.0|
|1990|       ABOVE 35|             400.0|
|1990|       BELOW 20|             200.0|
|1991|BETWEEN 20 & 35|             800.0|
|1991|       BELOW 20|             600.0|
|1991|       ABOVE 35|             400.0|
|1992|       ABOVE 35|             800.0|
|1992|BETWEEN 20 & 35|             400.0|
|1992|       BELOW 20|             200.0|
|1993|       BELOW 20|            1000.0|
|1993|BETWEEN 20 & 35|             200.0|
|1993|       ABOVE 35|             200.0|
|1994|BETWEEN 20 & 35|             200.0|
+----+---------------+------------------+
```