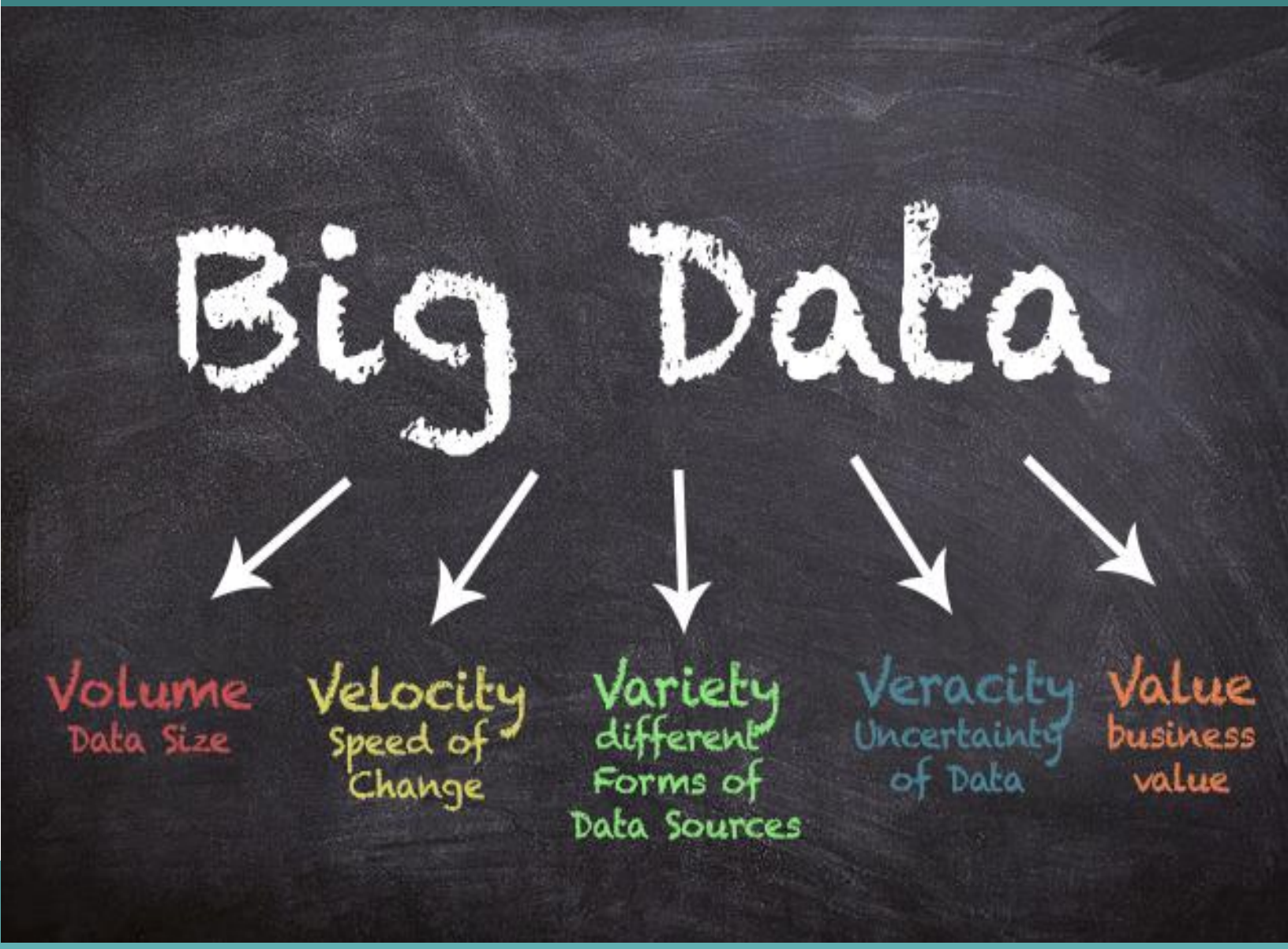
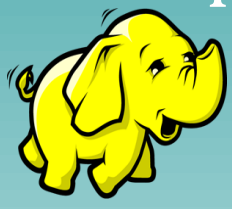
09/09/2018 | By: Ajit K Prasad



Big Data Engineering with Hadoop & Spark

Assignment on Spark SQL



Acadgild

Session 20: Assignment 20.1

This assignment is aimed at consolidating the concepts that was learnt during the Apache Spark SQL session of the course.

Associated Data Files

Datasets can be downloaded from this ***link***.

- Jump to the ***Source Code***.
- Jump to the ***output*** of the datasets.

Task 1:

Problem Statement:

1. What is the distribution of the total number of air-travelers per year?
2. What is the total air distance covered by each user per year?
3. Which user has travelled the largest distance till date?
4. What is the most preferred destination for all users?
5. Which route is generating the most revenue per year?
6. What is the total amount spent by every user on air-travel per year?
7. Considering age groups of < 20, 20-35, 35 >, which age group is travelling the most every year?

Source Code for all the task to be performed

```
package SQL

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.{col}

object TravelSQL {
  case class HolidaysDetails(UID: Int, Arrival: String, Destination: String,
Transport_Mode: String, Distance: Int, Year: Long)

  case class TransportMode(Trans_Mode: String, Trans_Expense: Int)

  case class UserID(userID: Int, userName: String, userAge: Int)

  def main (args: Array[String]): Unit = {

    println("hello Scala, this is Travel Data Analysis using Spark SQL")

    /***Created a spark session object for spark application***/
    val spark = SparkSession
      .builder()
      .master("local")
      .appName("Spark SQL Travel Example")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

    //Set the log level as warning
    spark.sparkContext.setLogLevel("WARN")

    println("Spark Session Object Created")

    /***For implicit conversions like converting RDDs and sequences to DataFrames***/
    import spark.implicits._

    /***Loading Holidays Dataset***/
    val HolidaysFromFile = spark.sparkContext
      .textFile("D:\\AcadGild\\ScalaCaseStudies\\Datasets\\Travel\\s20\\Holidays.txt")
      .map(_.split(","))
      .map(x => HolidaysDetails(x(0).toInt, x(1), x(2), x(3), x(4).toInt, x(5).toInt))
      .toDF()
    HolidaysFromFile.show()
    print("Holidays dataframe displayed\n\n\n")

    /***Loading Transport Dataset***/
    val TransportFromFile = spark.sparkContext

.textFile("D:\\AcadGild\\ScalaCaseStudies\\Datasets\\Travel\\s20\\TransportMode.txt")
      .map(_.split(","))
      .map(x => TransportMode(x(0), x(1).toInt))
      .toDF()
    TransportFromFile.show()
    print("Transport mode dataframe displayed\n\n\n")

    /***Loading User Details Dataset***/
    val UserIDFromFile = spark.sparkContext

.textFile("D:\\AcadGild\\ScalaCaseStudies\\Datasets\\Travel\\s20\\UserDetails.txt")
      .map(_.split(","))
      .map(x => UserID(x(0).toInt, x(1), x(2).toInt))
      .toDF()
    UserIDFromFile.show()
    print("User Details dataframe displayed\n\n\n")

    /***What is the distribution of the total number of air-travelers per year?***/
```

```
    /***Create a view on the Data Frames called "HolidayData"***/
    print("HolidayData View created\n")
    print("Distribution of total number of air-travelers per year analyzed\n")
    HolidaysFromFile.createOrReplaceTempView("HolidayData")
    spark.sql(""" select Year, count("Year") from HolidayData Group by Year""")
      .show()

    /***What is the total air distance covered by each user per year?***/
    print("Users total air distance per year analyzed\n")
    val joindf = HolidaysFromFile.as ('a)
      .join(UserIDFromFile.toDF().as('b), $"a.UID" === $"b.userID")
    joindf.createOrReplaceTempView("joinView")
    spark.sql(""" select UID,userName,Year,Sum(Distance)as TotalDistance from joinView
        |Group by Year,UID,userName
        |order by TotalDistance desc""".stripMargin)
      .show()

    /***Which user has travelled the largest distance till date?***/
    print("Users with largest distance analyzed\n")
    val task1 = spark.sql("""select UID,userName,Year,Sum(Distance)as TotalDistance
from joinView
        |Group by Year,UID,userName""".stripMargin)
    task1.toDF().createOrReplaceTempView("DistanceView")

    spark.sql(""" select UID,userName,Year,Max(TotalDistance) as MaximumDistance from
DistanceView
        |Group by Year,UID,userName
        |order by MaximumDistance desc """.stripMargin)
      .show(1)

    /***What is the most preferred destination for all users?***/
    print("Most preferred destination analyzed\n")
    spark.sql("""select Destination,count(Destination) as mostPrefDest
        |from joinView group by Destination
        |order by mostPrefDest desc """.stripMargin)
      .show(1)

    /***Which route is generating the most revenue per year?***/
    print("revenueView Created\n")
    val joineddf = HolidaysFromFile.as('c)
      .join(TransportFromFile.toDF().as('d),$"c.Transport_Mode" === $"d.Trans_Mode",
joinType = "left_outer")
    joineddf.createOrReplaceTempView("revenueView")


    print("maxRevenueView Created\n")
    val revenue = spark.sql("""select
UID,Destination,Transport_Mode,Year,count(Transport_Mode) * sum(Trans_Expense)
        |as revenueExpense from revenueView
        |group by UID,Year,Destination,Transport_Mode""".stripMargin)
    revenue.toDF().createOrReplaceTempView("maxRevenue")

    print("Route generating most revenue per year analyzed\n")
    spark.sql("""select Destination, Transport_Mode,Year,max(revenueExpense) as
maximumRevenue from maxRevenue
        |group by Destination,Transport_Mode,Year
        |order by maximumRevenue desc""".stripMargin)
      .show()


    /***What is the total amount spent by every user on air-travel per year?***/
    val expense = spark.sql(""" select
UID,Destination,Transport_Mode,Year,sum(Trans_Expense) as totalExpense from
revenueView
        |group by UID,Year,Destination,Transport_Mode""".stripMargin)
      .filter(col("Transport_Mode") === "airplane")
```

```
    println("Transportation Expense calculated and filtered for Air Travel")

    val newJoindf = UserIDFromFile.toDF().as('e).join(expense.as('f), $"e.userID" ===
$"f.UID")

    newJoindf.toDF().createOrReplaceTempView("expenseView")
    println("ExpenseView Created\n")
    print("Total amount spent by every user on Air Travel per year analyzed\n")
    spark.sql(""" select UID, Transport_Mode, Year, totalExpense from expenseView
        |group by UID,Year,totalExpense,Transport_Mode""".stripMargin)
      .show()

    /***Considering age groups of < 20, 20-35, 35 >, which age group is travelling the
most every year?***/
    print("Age wise grouping of travel data every year analyzed\n")
    spark.sql("""select userAge,count(UID) as countTravel from joinView WHERE userAge
>= 20
        |AND userAge <= 35 group by userAge,UID
        |order by countTravel desc """.stripMargin)
      .show()
  }
}
```

Output of Dataset Used

1. Holidays.txt

TravelSQL

```
+---+-------+-----------+-------------+--------+----+
|UID|Arrival|Destination|Transport_Mode|Distance|Year|
+---+-------+-----------+-------------+--------+----+
|  1|    CHN|        IND|      airplane|     200|1990|
|  2|    IND|        CHN|      airplane|     200|1991|
|  3|    IND|        CHN|      airplane|     200|1992|
|  4|    RUS|        IND|      airplane|     200|1990|
|  5|    CHN|        RUS|      airplane|     200|1992|
|  6|    AUS|        PAK|      airplane|     200|1991|
|  7|    RUS|        AUS|      airplane|     200|1990|
|  8|    IND|        RUS|      airplane|     200|1991|
|  9|    CHN|        RUS|      airplane|     200|1992|
| 10|    AUS|        CHN|      airplane|     200|1993|
|  1|    AUS|        CHN|      airplane|     200|1993|
|  2|    CHN|        IND|      airplane|     200|1993|
|  3|    CHN|        IND|      airplane|     200|1993|
|  4|    IND|        AUS|      airplane|     200|1991|
|  5|    AUS|        IND|      airplane|     200|1992|
|  6|    RUS|        CHN|      airplane|     200|1993|
|  7|    CHN|        RUS|      airplane|     200|1990|
|  8|    AUS|        CHN|      airplane|     200|1990|
|  9|    IND|        AUS|      airplane|     200|1991|
| 10|    RUS|        CHN|      airplane|     200|1992|
+---+-------+-----------+-------------+--------+----+
only showing top 20 rows

Holidays dataframe displayed
```

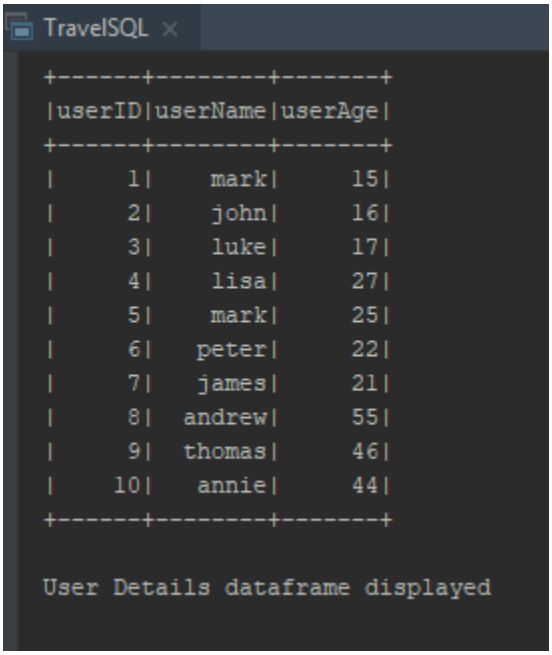
2. TransportMode.txt

TravelSQL

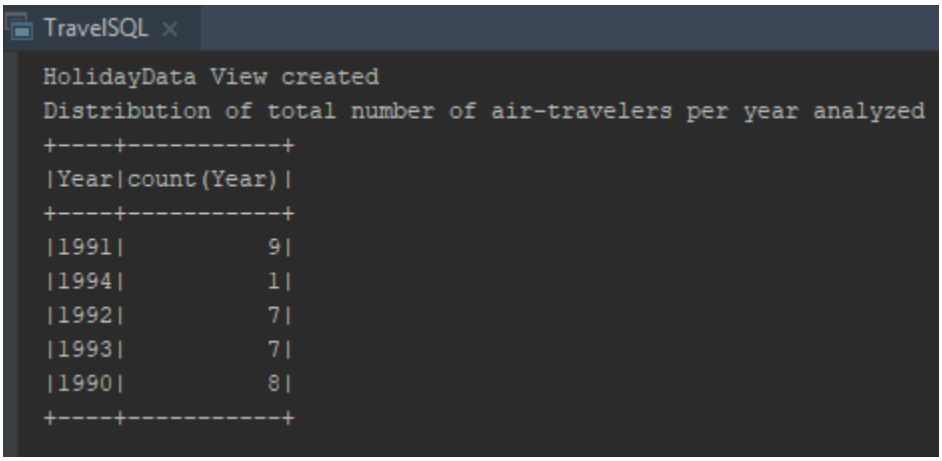
```
+----------+-------------+
|Trans_Mode|Trans_Expense|
+----------+-------------+
|  airplane|          170|
|       car|          140|
|     train|          120|
|      ship|          200|
+----------+-------------+

Transport mode dataframe displayed
```

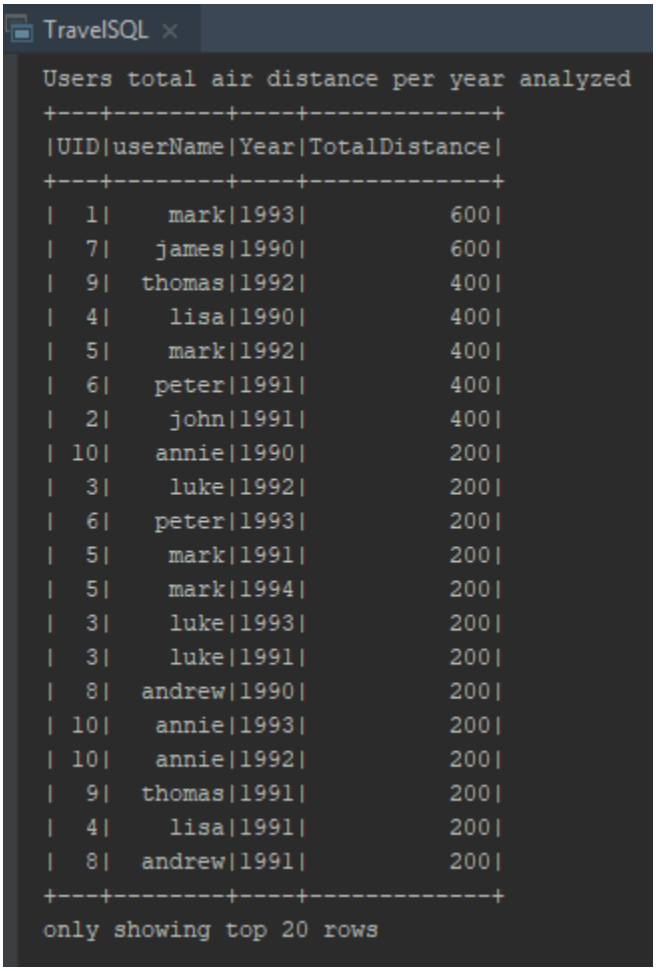
3. UserDetails.txt



Output of Task performed for Problem Statement 1



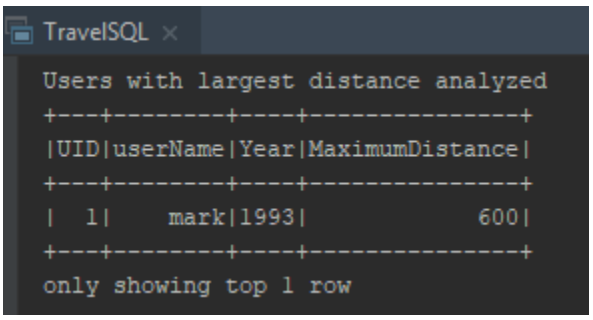
Output of Task performed for Problem Statement 2



TravelSQL

```
Users total air distance per year analyzed
+---+--------+----+-------------+
|UID|userName|Year|TotalDistance|
+---+--------+----+-------------+
|  1|    mark|1993|          600|
|  7|   james|1990|          600|
|  9|  thomas|1992|          400|
|  4|    lisa|1990|          400|
|  5|    mark|1992|          400|
|  6|   peter|1991|          400|
|  2|    john|1991|          400|
| 10|   annie|1990|          200|
|  3|    luke|1992|          200|
|  6|   peter|1993|          200|
|  5|    mark|1991|          200|
|  5|    mark|1994|          200|
|  3|    luke|1993|          200|
|  3|    luke|1991|          200|
|  8|  andrew|1990|          200|
| 10|   annie|1993|          200|
| 10|   annie|1992|          200|
|  9|  thomas|1991|          200|
|  4|    lisa|1991|          200|
|  8|  andrew|1991|          200|
+---+--------+----+-------------+
only showing top 20 rows
```

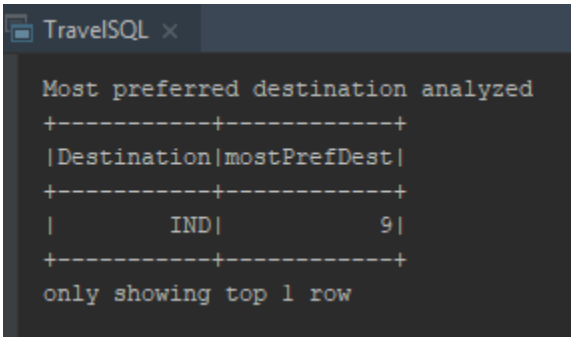
Output of Task performed for Problem Statement 3



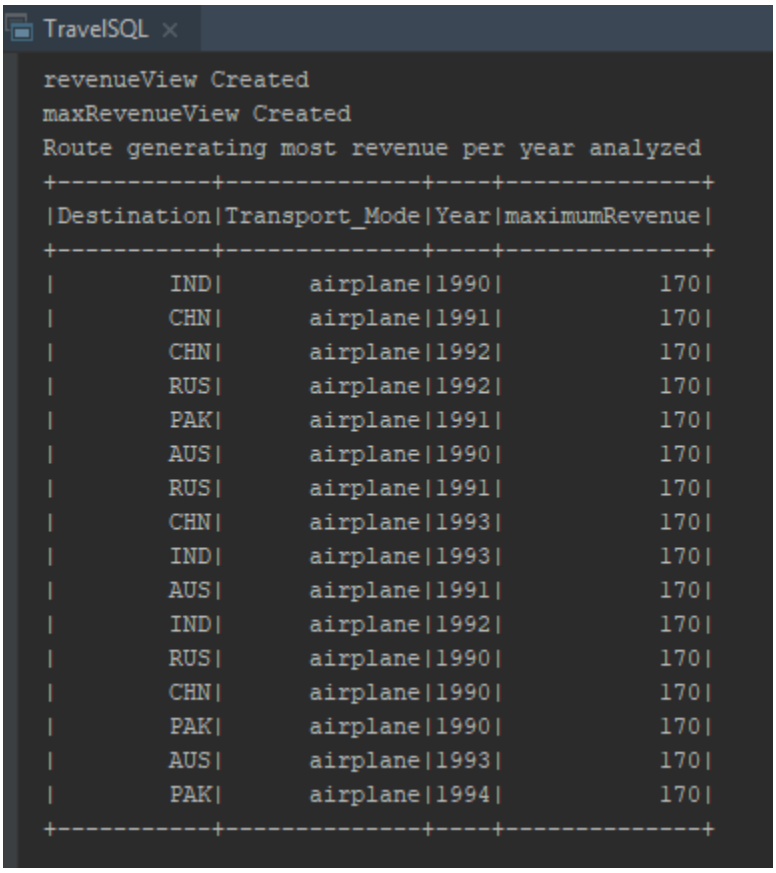
TravelSQL

```
Users with largest distance analyzed
+---+--------+----+---------------+
|UID|userName|Year|MaximumDistance|
+---+--------+----+---------------+
|  1|    mark|1993|            600|
+---+--------+----+---------------+
only showing top 1 row
```

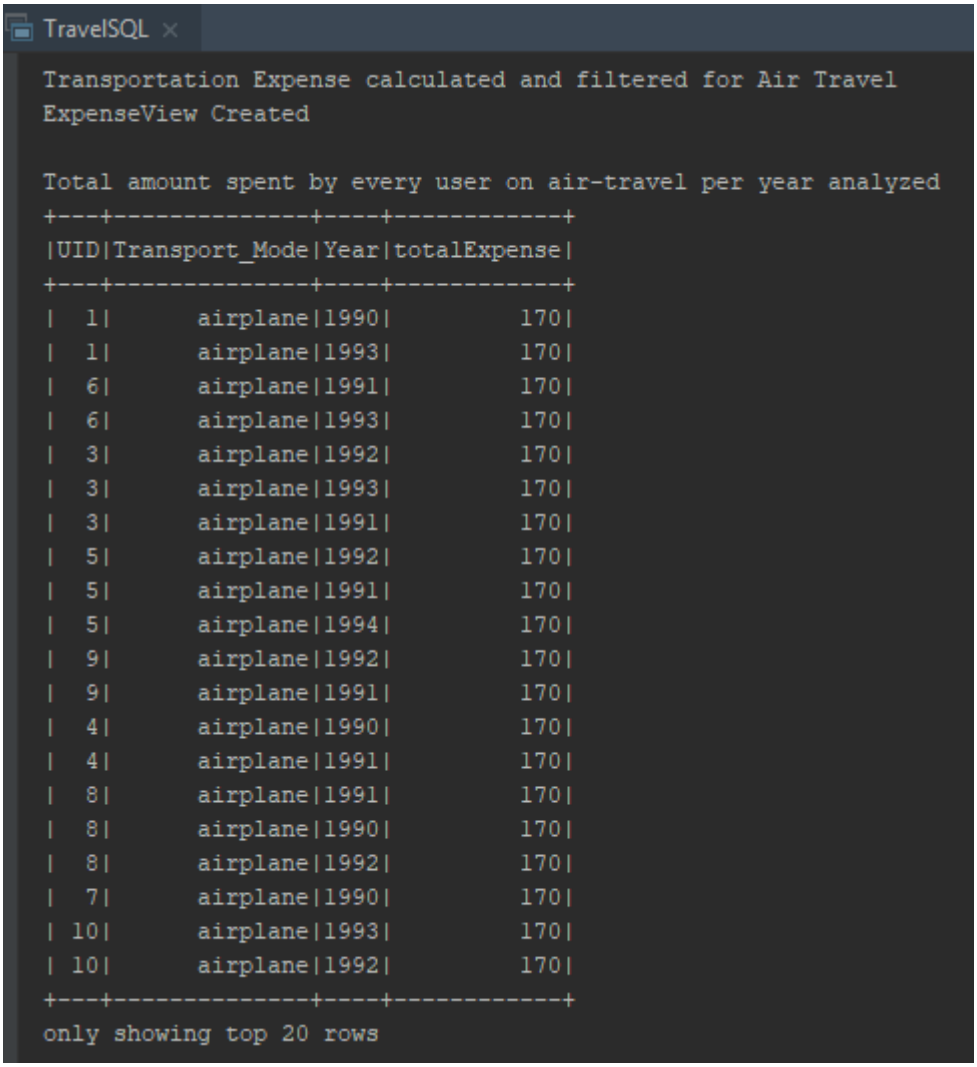
Output of Task performed for Problem Statement 4



Output of Task performed for Problem Statement 5



Output of Task performed for Problem Statement 6



Output of Task performed for Problem Statement 7