

Assignment20(SparkSql1):

4. Associated Data Files

https://drive.google.com/open?id=1oWb_lxlzb5PkFgf6P6lwbHXubAMI-HvK

5. Problem Statement Task 1

1) What is the distribution of the total number of air-travelers per year

Query used:

```
val data1 = spark.sql("select year, count(TravelMode) from HolidayTrip group by year")
data1.show()
```

Output:

```
+---+-----+
|year|count(TravelMode)|
+---+-----+
|1991|          9|
|1994|          1|
|1992|          7|
|1993|          7|
|1990|          8|
+---+-----+
```

2) What is the total air distance covered by each user per year

Query used:

Added extra query to order the output based on the userId and year

```
//the total air distance covered by each user per year
val TotAirDistCovered = spark.sql("select UserId, year, sum(Distance) from HolidayTrip group by year, UserId order by UserId, year ")
TotAirDistCovered.show()
```

Output:

```
-----  
+-----+-----+-----+  
|UserId|year|sum(Distance)|  
+-----+-----+-----+  
|      1|1993|          600|  
|      10|1990|          200|  
|      10|1992|          200|  
|      10|1993|          200|  
|       2|1991|          400|  
|       2|1993|          200|  
|       3|1991|          200|  
|       3|1992|          200|  
|       3|1993|          200|  
|       4|1990|          400|  
|       4|1991|          200|  
|       5|1991|          200|  
|       5|1992|          400|  
|       5|1994|          200|  
|       6|1991|          400|  
|       6|1993|          200|  
|       7|1990|          600|  
|       8|1990|          200|  
|       8|1991|          200|  
|       8|1992|          200|  
+-----+-----+-----+  
only showing top 20 rows
```

3) Which user has travelled the largest distance till date

Query used:

```
-----  
//Which user has travelled the largest distance till date  
val LargestDistanceUser = spark.sql("select sum(Distance) as MaxDistance, UserId from  
HolidayTrip group by UserId order by MaxDistance desc").take(1)  
println(s"Gives Largest distance covered by the user and user id  
${LargestDistanceUser.foreach(println)} ")
```

Output:

```
-----  
[800,5]  
Gives Largest distance covered by the user and user id ()
```

4) What is the most preferred destination for all users.

Query used:

```
-----  
//Most preferred destination for all users.  
val PreferredDest = spark.sql("select count(*) as distribution, Destination from  
HolidayTrip group by Destination order by distribution desc").take(1)  
println(s"Gives the number of times and Destination visted by the user  
${PreferredDest.foreach(println)} ")
```

Output:

```
-----  
[9, IND]  
Gives the number of times and Destination visted by the user  ()  
18:45 (20: 15:46:56) INFO: SparkSqlProgram: Reader completed: TotalRows=
```

5) Which route is generating the most revenue per year

Query used:

```
-----  
//Which route is generating the most revenue per year  
val MoreRevenue = spark.sql("select year, source, Destination, sum(expense) as  
MaxExpense from TotAmountPerUser group by year, source, Destination order by  
MaxExpense desc")  
MoreRevenue.show()
```

Output:

```
-----  
+----+-----+-----+-----+  
|year|source|Destination|MaxExpense|  
+----+-----+-----+-----+  
|1991|  IND|      RUS|    340.0|  
|1991|  IND|      AUS|    340.0|  
|1993|  AUS|      CHN|    340.0|  
|1992|  RUS|      IND|    340.0|  
|1990|  CHN|      IND|    340.0|  
|1993|  CHN|      IND|    340.0|  
|1992|  CHN|      RUS|    340.0|  
|1991|  PAK|      RUS|    170.0|  
|1992|  AUS|      IND|    170.0|  
|1991|  CHN|      PAK|    170.0|  
+----+-----+-----+-----+  
only showing top 10 rows
```

6) What is the total amount spent by every user on air-travel per year

Query used:

```
-----  
// What is the total amount spent by every user on air-travel per year  
val newColumn = when(col("TravelMode").equalTo("airplane"),  
"170").when(col("TravelMode").equalTo("ship"),  
"200").when(col("TravelMode").equalTo("car"),  
"140").when(col("TravelMode").equalTo("train"), "120")  
val newData = Holiday_data.withColumn("Expense",newColumn) //Adding new column  
containing the expenses  
newData.createOrReplaceTempView("TotAmountPerUser")  
val TotExpenseperuser = spark.sql("select UserId, year, sum(expense) from  
TotAmountPerUser where TravelMode == 'airplane' group by Year, UserId order by Year,  
UserId")  
TotExpenseperuser.show()
```

Output:

```
-----  
+-----+-----+-----+  
|UserId|year|sum(CAST(expense AS DOUBLE))|  
+-----+-----+-----+  
| 10|1990|170.0|  
| 4|1990|340.0|  
| 7|1990|510.0|  
| 8|1990|170.0|  
| 1|1990|170.0|  
| 2|1991|340.0|  
| 3|1991|170.0|  
| 4|1991|170.0|  
| 5|1991|170.0|  
| 6|1991|340.0|  
| 8|1991|170.0|  
| 9|1991|170.0|  
| 10|1992|170.0|  
| 3|1992|170.0|  
| 5|1992|340.0|  
| 8|1992|170.0|  
| 9|1992|340.0|  
| 1|1993|510.0|  
| 10|1993|170.0|  
| 2|1993|170.0|  
+-----+-----+-----+  
only showing top 20 rows
```

7) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year.

Query used:

```
//Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year

val newUserColumn = Holiday_data.join(UserDetails,Holiday_data("UserId") <=>
UserDetails("UserId") )
newUserColumn.show()

newUserColumn.createOrReplaceTempView("YearwiseDetails")

val Agegroup = spark.sql("select Year, age ,count(Age) as Agecount from
YearwiseDetails group by Year,age order by Agecount desc").take(1)

println(s"Gives the year, agegroup, Maximum number of times travelled
${Agegroup.foreach(println)} ")
```

Output:

```
[1990,21,3]
Gives the year, agegroup, Maximum number of times travelled ()
```