Assignment20(SparkSql1):

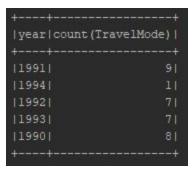
4. Associated Data Files

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

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

```
//To get the distribution of the total number of air-travelers per year
val TotAirTravellers = spark.sql("select year, count(TravelMode) from HolidayTrip
where TravelMode = 'airplane' group by year ")
TotAirTravellers.show()
```

Output:



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 userld and year

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

Output:

UserId year sum(Dista 	6001
1 1993 10 1990 10 1992 10 1993 2 1991 2 1993 3 1991	6001
10 1990 10 1992 10 1993 2 1991 2 1993 3 1991	
10 1990 10 1992 10 1993 2 1991 2 1993 3 1991	
10 1992 10 1993 2 1991 2 1993 3 1991	
10 1993 2 1991 2 1993 3 1991	200
2 1991 2 1993 3 1991	200
2 1993 3 1991	2001
3 1991	4001
	200
1 3119921	200
	2001
3 1993	2001
4 1990	400
4 1991	200
5 1991	2001
5 1992	4001
1 5 1994	200
[6]1991]	4001
6 1993	2001
1 7119901	6001
8 1990	2001
8 1991	2001
1 8119921	
+	
only showing top 20 rd	2001

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 PrefferedDest = 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
${PrefferedDest.foreach(println)} ")
```

Output:

```
[9,IND]
Gives the number of times and Destination visted by the user ()
```

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:

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:

+	+	+
Us	erId year sum(CAST(e	xpense AS DOUBLE))!
+	10110001	170.01
	10 1990 4 1990	170.0 340.0
	7 1990	510.01
	8 1990	170.01
	1 1990	170.01
	2 1991	340.01
	3 1991	170.01
	4 1991	170.01
	5 1991	170.01
	6 1991	340.01
	8 1991	170.01
	9 1991	170.01
	10 1992	170.01
	3 1992	170.01
	5 1992	340.01
	8 1992	170.01
	9 1992	340.01
	1 1993	510.01
	10 1993	170.01
	2 1993	170.0
+	++	+
onl	y showing top 20 row	3

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, Maximium number of times travelled
${Agegroup.foreach(println)} ")
```

Output:

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