**ASSIGNMENT 6.6**

**Problem Statement:**

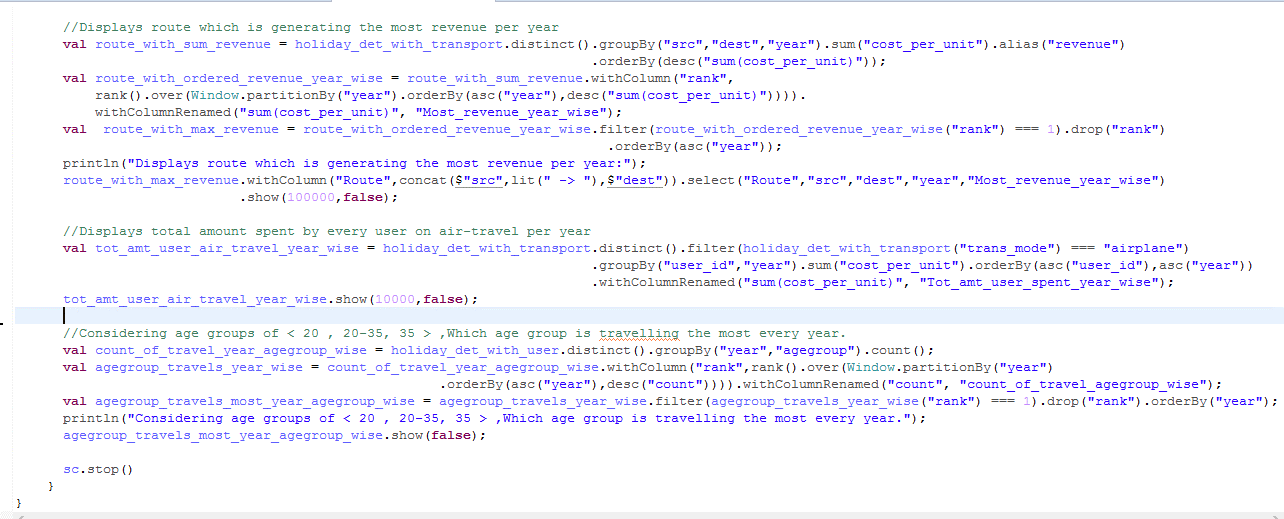
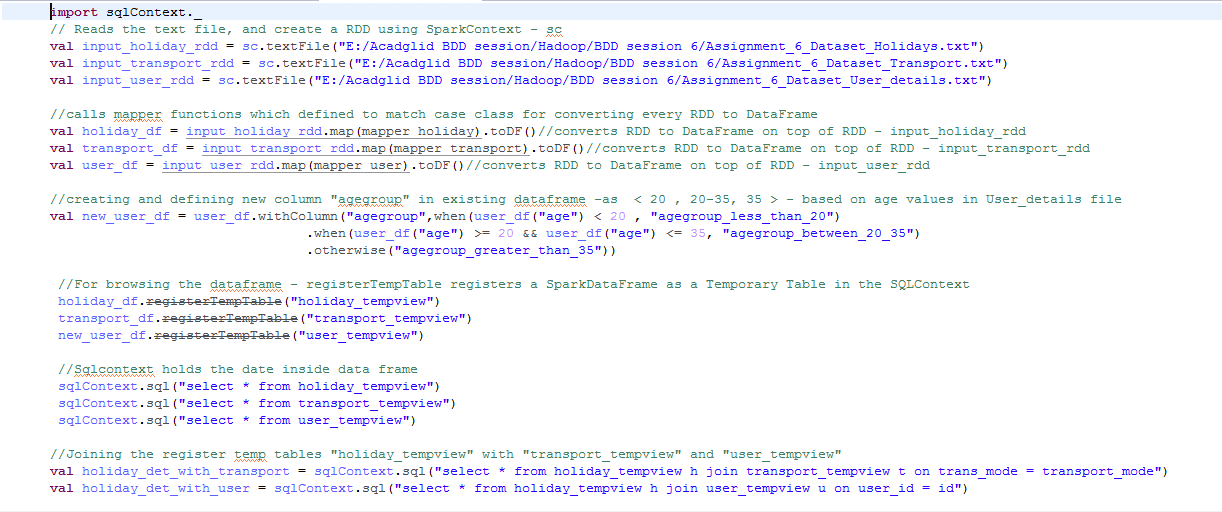
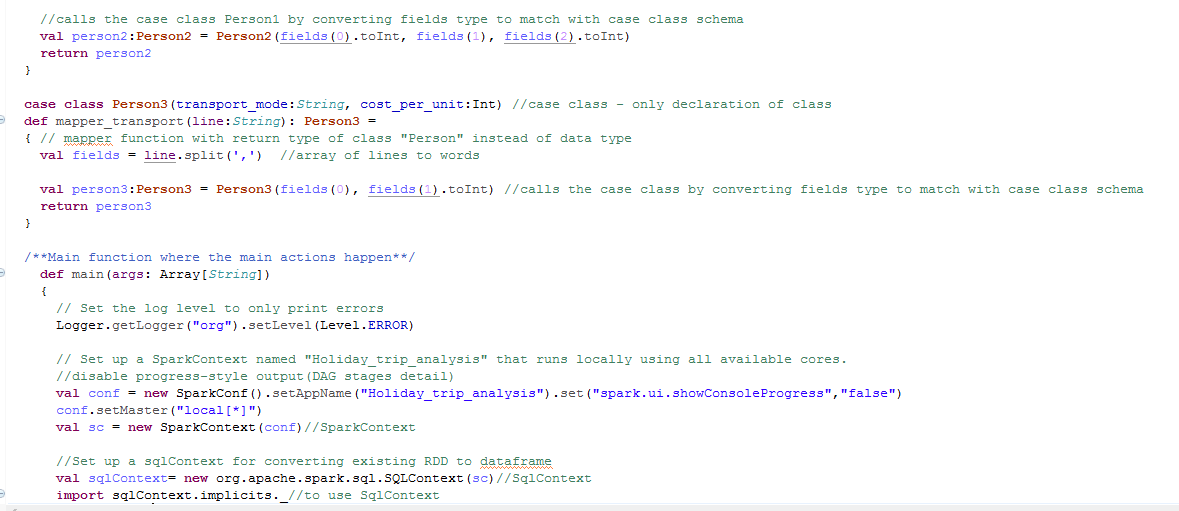
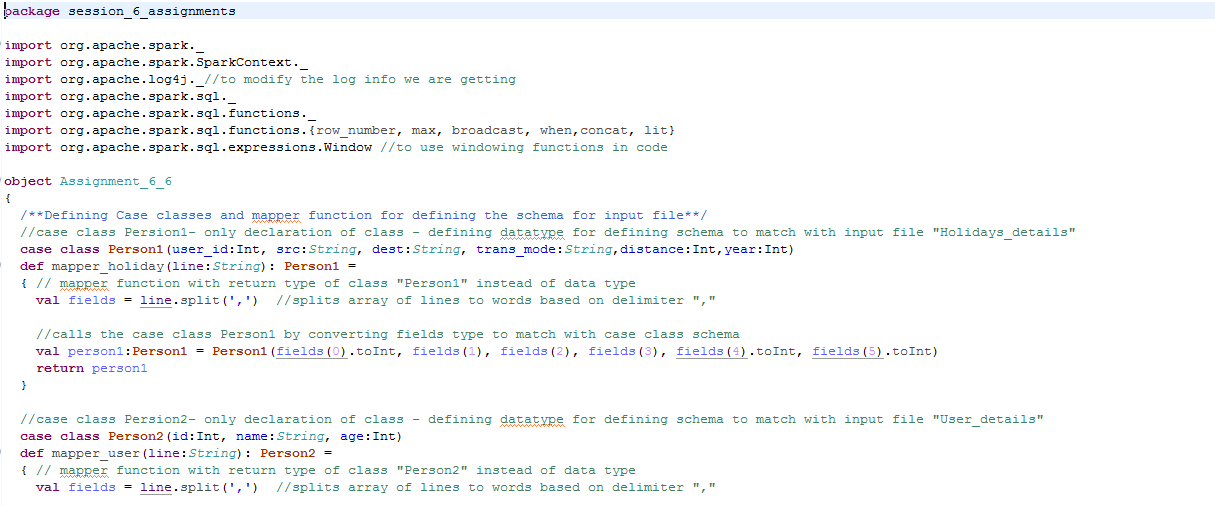
1) Which route is generating the most revenue per year

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

3) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most

every year.

**Scala code:**



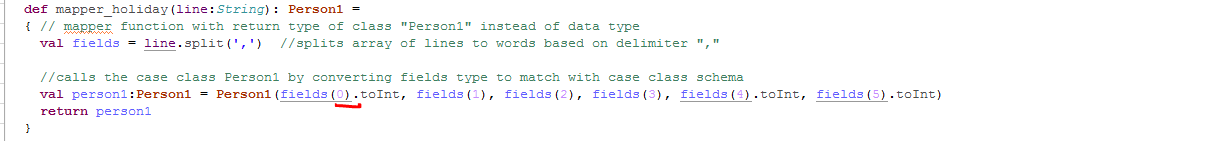
**Code explanation:**

* Created 3 **case classes** called “Person1”,”Person2”,Person3” for defining the schema and name of the fields in input file

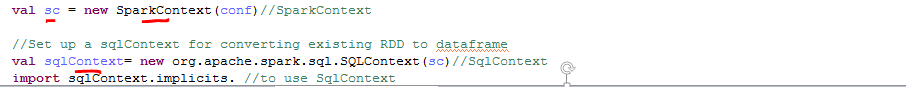
Person1(Holidays file),Person2(Transport file),Person3(User\_details file)



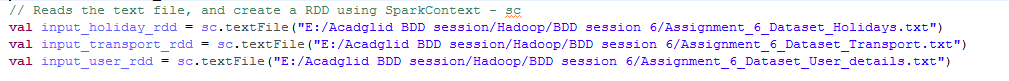
* Created 3 **mapper functions** for all 3 input files – which will convert the datatype of the fields to match with corresponding Person class schema and returns “Person” type



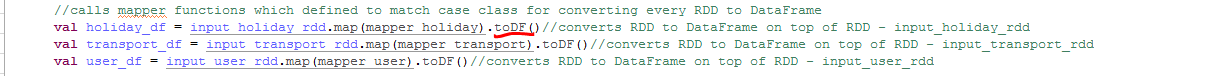
* Set the **log level to “Error”** to display only error messages - Logger.getLogger("org").setLevel(Level.ERROR) **and disabled progress-style** output(DAG stage details) - set("spark.ui.showConsoleProgress","false")
* Set up a **SparkContext** named – sc – for application "Holiday\_trip\_analysis" that runs locally using all available cores.
* Set up a **sqlContext** for converting existing RDD to **dataframe**



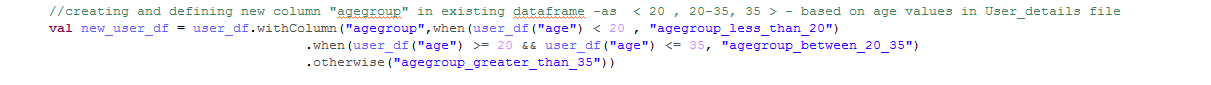
* Reads the text file, and create a RDD using SparkContext – sc



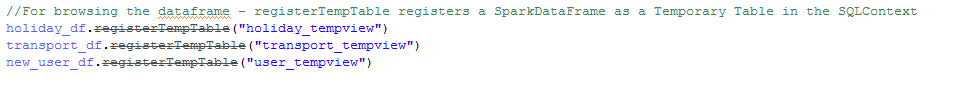
* Calls mapper functions which defined to match case class for converting every RDD to DataFrame
* Calls mapper functions by passing mapper function with RDD and retunrs Dataframe as toDF() is used



* For the problem statement, **created and defined a new column "agegroup" in existing dataframe** -as < 20 , 20-35, 35 > - based on age values in User\_details file using withColumn – which is to create or add columns dynamically to the new data frame from existing DF



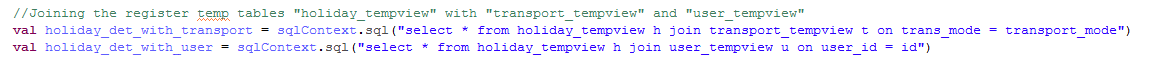
* For browsing the dataframe **- registerTempTable registers a SparkDataFrame** as a Temporary Table in the SQLContext and Sqlcontext holds the date of data frame



* Now as the data from RDD is converted to Dataframe,it can be queried with “SparkContext”.

**Joins:**

* Joined the registered temp table “"holiday\_tempview" with "transport\_tempview" based on transport\_mode to get the value of “cost\_per\_unit” from transport file with SqlContext
* Joined the registered temp table “"holiday\_tempview" with "user\_tempview" based on id to get the value of “age” from user file.

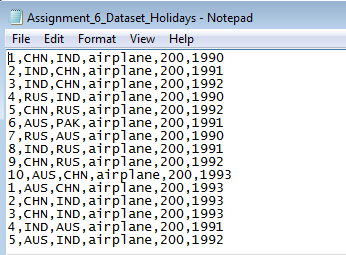


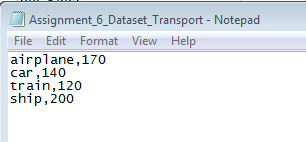
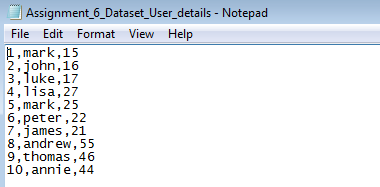
**Snapshot of input file:**

User: id, name, age – Assignment\_6\_Dataset\_User\_details

Transport: transport\_mode, cost\_per\_unit - Assignment\_6\_Dataset\_Transport

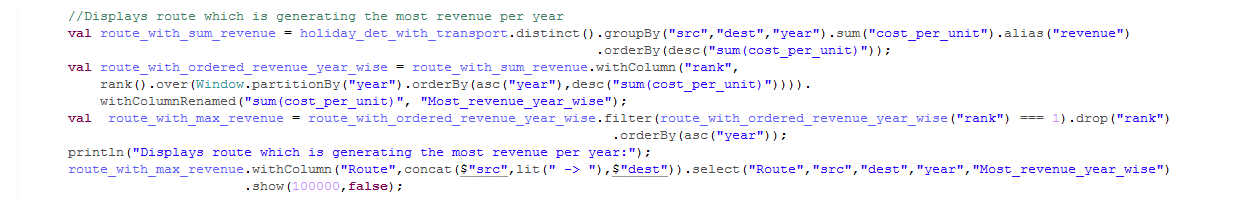
Holidays: id, src, dest, transport\_mode, distance, year - Assignment\_6\_Dataset\_Holidays



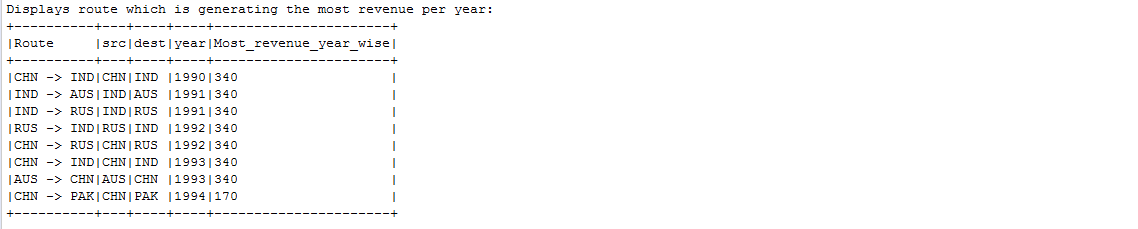
**1) Which route is generating the most revenue per year**

* Grouped by **src,dest,year** and get the sum of cost\_per\_unit - will give the **sum(cost\_per\_unit)** – total revenue in year wise according to src,desc
* Using windowing function **partition by year** order by **desc(sum) and get the first row** – will get one row holding max(amt) in each and every year along with route
* Displayed the route by concatenating src & dest



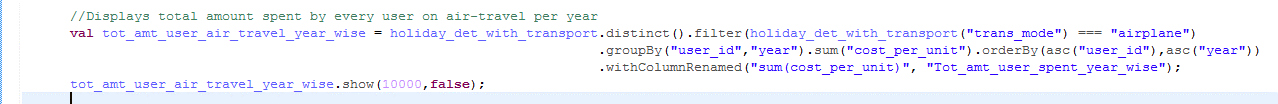
**Output:**

Displays Route(concatenation of src &desc) and highest sum(cost\_per\_unit) year wise. **If two records are having same “amount” for a year both records would be displayed.(as in year 1991)**



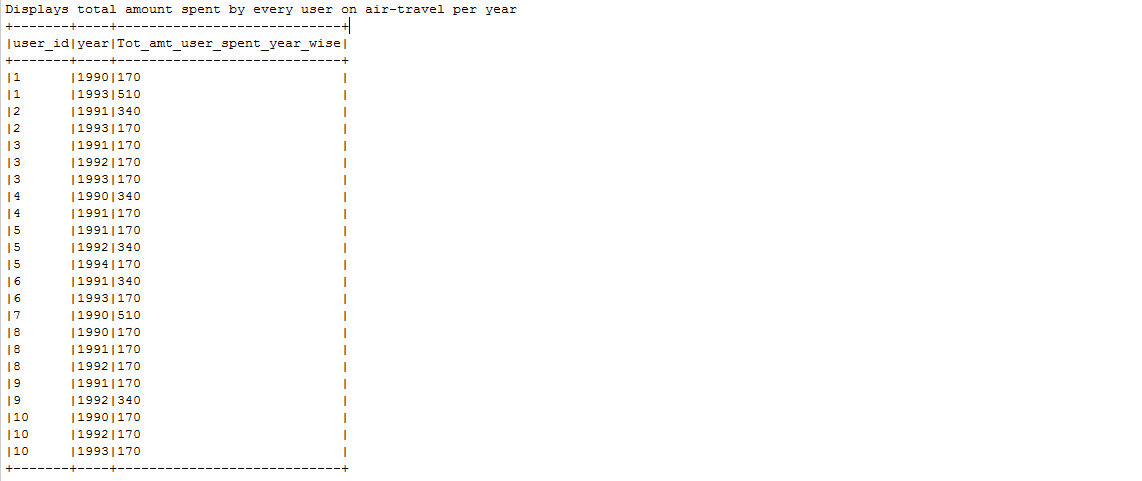
**2) What is the total amount spent by every user on air-travel per year**

* Filtered the records which has **transport mode as “airplane”** as it is supposed to include only air travel
* **Grouped by year and user\_id,year and taken sum(cost\_per\_unit)** – will hold sum(amount spent) per year for every user
* Show(false) – will show values without truncating in the console



**Output:**

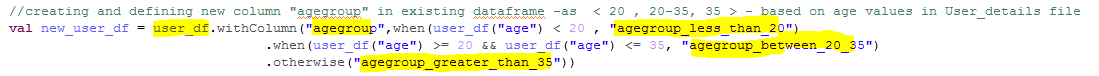
Displays the user\_id and total amount spent for air\_travel every year. Result is ordered in user\_id and year.



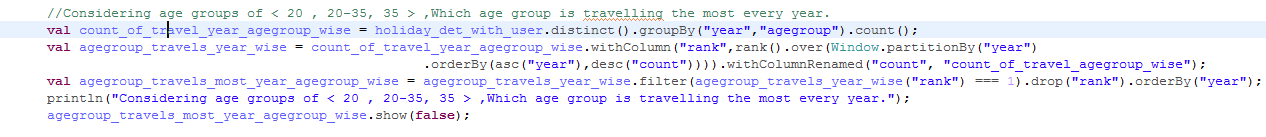
**3) Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most**

**every year.**

Created a **new column age group “agegroup\_less\_than\_20”,” agegroup\_between\_20-35”,” agegroup\_greater\_than\_35>”**already based on age present in user\_details file



* Grouped data based **on agegroup,year and take count of rows in every group** – will hold the **count of travel in every year according to each agegroup.**
* Using Windowing function **generated rank column** wherein data is **ordered based on count of travel in every year** - will display the all year, agegroup and count of travels according to agegroup in descending order in each year
* **And taken first row from every group of year and agegroup** – to get the **age group which is travelling most every year** – will hold the year, agegroup name which had highest travel in every year.



**Output:**

Displays the year, agegroup name which is travelling the most every year along with the count of travel

