Assignment 20.1
Spark SQL 1

Task 1 :

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.

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

In below program, we have created case classes for **holidays, transport** and **users** data file and created Spark object. Then we have loaded data from text files and converted them to DataFrames and then to tables. Then by using count() function and group by clause on year column in **sql** transformation, we have printed the total number of air-travelers per year.

Scala Code :

```
object Assignment_20 {

  case class holidays(id: Int, source: String, destination: String, transport_mode: String, distance: Int, year: Int)

  case class transport(transport_mode : String, cost_per_unit : Int)


  case class users(id : Int, name : String, age : Int)


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


    val spark = SparkSession
      .builder()
      .master("local")
      .appName("Spark SQL Assignment ")
      .config("spark.some.config.option", "some-value")
      .getOrCreate()

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

    println("Spark Session Object created")


  val holidayFile = spark.sparkContext.textFile("C:\\AcadGild Hadoop\\Assignments\\S20_Dataset_Holidays.csv");
```

Assignment 20.1
Spark SQL 1

```
    import spark.implicits._


 val  holidayDF =holidayFile
    .map( x => x.split(",")).map(x =>
holidays(x(0).trim.toInt,x(1),x(2),x(3),x(4).trim.toInt,x(5).trim.toInt)).toDF()



    holidayDF.registerTempTable("holidays")

    println("holidays table is created ")



    val TransportFile = spark.sparkContext.textFile("C:\\AcadGild
Hadoop\\Assignments\\S20_Dataset_Transport.csv");

    val TransportDF = TransportFile.map(x => x.split(",")).map(x =>
transport(x(0),x(1).trim.toInt)).toDF

    TransportDF.registerTempTable("transport")

    println("transport table is created ")



    val usersFile = spark.sparkContext.textFile("C:\\AcadGild
Hadoop\\Assignments\\S20_Dataset_User_details.csv")

    val usersDF = usersFile.map(x => x.split(",")).map(x =>
users(x(0).trim.toInt,x(1),x(2).trim.toInt)).toDF()

    usersDF.registerTempTable("users")

    println("users table is created ")



    println("Below is the distribution of the total number of air-travelers per
year")

    val holidays_sql = spark.sql("select year,count(*) no_of_air_travelers from
holidays group by year order by year")

    holidays_sql.show()
```

Output:

```
Spark Session Object created
holidays table is created
transport table is created
users table is created
Below is the distribution of the total number of air-travelers per year
+----+-------------------+
|year|no_of_air_travelers|
+----+-------------------+
|1990|                  8|
|1991|                  9|
|1992|                  7|
|1993|                  7|
|1994|                  1|
+----+-------------------+
```

Assignment 20.1
Spark SQL 1

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

We have used **sql** transformation to use sql query. In sql query, we have used join clause to join **holidays** and **users** tables and then used group by clause on id, name and year columns. Then we have printed the total air distance covered by each user per year.

Scala Code :

```
    println("Below is the total air distance covered by each user per year")

    val users_sql = spark.sql("select u.id,u.name,h.year,sum(h.distance)
total_distance from holidays h join users u on h.id = u.id group by
u.id,u.name,h.year order by u.id,u.name,h.year")

    users_sql.show()
```

Output:

```
Below is the total air distance covered by each user per year
+---+------+----+--------------+
| id|  name|year|total_distance|
+---+------+----+--------------+
|  1|  mark|1990|           200|
|  1|  mark|1993|           600|
|  2|  john|1991|           400|
|  2|  john|1993|           200|
|  3|  luke|1991|           200|
|  3|  luke|1992|           200|
|  3|  luke|1993|           200|
|  4|  lisa|1990|           400|
|  4|  lisa|1991|           200|
|  5|  mark|1991|           200|
|  5|  mark|1992|           400|
|  5|  mark|1994|           200|
|  6| peter|1991|           400|
|  6| peter|1993|           200|
|  7| james|1990|           600|
|  8|andrew|1990|           200|
|  8|andrew|1991|           200|
|  8|andrew|1992|           200|
|  9|thomas|1991|           200|
|  9|thomas|1992|           400|
+---+------+----+--------------+
only showing top 20 rows
```

Assignment 20.1
Spark SQL 1

3) Which user has travelled the largest distance till date

We have used **sql** transformation to use sql query. In sql query, we have used join clause to join **holidays** and **users** tables and then used group by clause on id and name columns to take sum of distance and sorted them in descending order and picked first record from top by using limit clause. Then we have printed that user who has travelled largest distance till date.

Scala Code :

```
  println("Below is the user who has travelled the largest distance till date")

    val max_distance_sql = spark.sql("select h.id,u.name,sum(distance)
total_distance from holidays h join users u on u.id = h.id group by h.id,u.name
order by total_distance desc limit 1")

  max_distance_sql.show()
```

Output:

```
Below is the user who has travelled the largest distance till date
+---+----+--------------+
| id|name|total_distance|
+---+----+--------------+
|  1|mark|           800|
+---+----+--------------+
```

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

We have used **sql** transformation to use sql query. In sql query, we have used count function in holidays table and used group by clause on destination column and sorted them in descending order and picked first record from top by using limit clause. Then we have printed that record. i.e. most preferred destination for all users.

Scala Code :

```
    println("Below is the most preferred destination for all users")

    val destination_sql = spark.sql("select destination,count(*) max_count from
holidays group by destination order by max_count desc limit 1")

   destination_sql.show()
```

Output :

```
Below is the most preferred destination for all users
+-----------+---------+
|destination|max_count|
+-----------+---------+
|        IND|        9|
+-----------+---------+
```

Assignment 20.1
Spark SQL 1

5) Which route is generating the most revenue per year

We have used **sql** transformation to use sql query. In sql query, we have used join clause to join **holidays** and **transport** tables and created **revenue** table from this dataframe.

Then we have used group by clause on year and route to take sum of multiplication of distance and cost per unit and created **total_revenue** table from that dataframe.

Then we have taken maximum of revenue by using group by clause on year on **total_revenue** table and created **max_revenue_per_dest** table from this dataframe.

After this we have used join clause to join **max_revenue_per_dest** and **total_revenue** tables and then printed the route which is generating the most revenue per year.

Scala Code :

```
println("Below is the route which is generating the most revenue per year")

val max_revenue_sqlDF = spark.sql("select
id,source,destination,h.transport_mode,distance,year,cost_per_unit from transport t
join holidays h on h.transport_mode = t.transport_mode").toDF()

max_revenue_sqlDF.createOrReplaceTempView("revenue")


val revenue_sql = spark.sql("select sum(distance*cost_per_unit) totalRevenue,
year,concat(source,destination) route from revenue group by
year,concat(source,destination) order by year").toDF()

revenue_sql.createOrReplaceTempView("total_revenue")

val revenue_sql2 = spark.sql("select * from total_revenue")



val max_revenue_sql = spark.sql("select max(totalRevenue) max_revenuePerDest, year
from total_revenue group by year order by year").toDF()

max_revenue_sql.createOrReplaceTempView("max_revenue_per_dest")

val revenue_sql3 = spark.sql("select * from max_revenue_per_dest")



val max_revenue_sql2 = spark.sql("select a.route, a.year,a.totalRevenue from
total_revenue a join max_revenue_per_dest b on a.totalRevenue=b.max_revenuePerDest
and a.year = b.year order by a.year asc, a.totalRevenue desc").toDF()

max_revenue_sql2.createOrReplaceTempView("max_revenue")

val revenue_sql4 = spark.sql("select * from max_revenue")

revenue_sql4.show()
```

Assignment 20.1
Spark SQL 1

Output :

```
Below is the route which is generating the most revenue per year
+------+----+------------+
| route|year|totalRevenue|
+------+----+------------+
|CHNIND|1990|       68000|
|INDRUS|1991|       68000|
|INDAUS|1991|       68000|
|CHNRUS|1992|       68000|
|RUSIND|1992|       68000|
|AUSCHN|1993|       68000|
|CHNIND|1993|       68000|
|CHNPAK|1994|       34000|
+------+----+------------+
```

Assignment 20.1
Spark SQL 1

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

We have used **sql** transformation to use sql query. In sql query, we have used join clause to join **holidays, transport** and **users** tables and then used group by clause on id , name and year columns to take total sum of multiplication of distance and cost per unit. Then we have printed the total amount spent by every user on air-travel per year

Scala Code :

```
println("Below is the total amount spent by every user on air-travel per year")

val total_amount_sql = spark.sql("select
h.id,u.name,year,sum(distance*t.cost_per_unit) total_amount from holidays h join
transport t join users u on h.transport_mode = t.transport_mode and u.id = h.id
where h.transport_mode = 'airplane'  group by h.id,u.name,year order by
h.id,u.name,year")

total_amount_sql.show()
```

Output :

```
Below is the total amount spent by every user on air-travel per year
+---+------+----+------------+
| id|  name|year|total_amount|
+---+------+----+------------+
|  1|  mark|1990|       34000|
|  1|  mark|1993|      102000|
|  2|  john|1991|       68000|
|  2|  john|1993|       34000|
|  3|  luke|1991|       34000|
|  3|  luke|1992|       34000|
|  3|  luke|1993|       34000|
|  4|  lisa|1990|       68000|
|  4|  lisa|1991|       34000|
|  5|  mark|1991|       34000|
|  5|  mark|1992|       68000|
|  5|  mark|1994|       34000|
|  6| peter|1991|       68000|
|  6| peter|1993|       34000|
|  7| james|1990|      102000|
|  8|andrew|1990|       34000|
|  8|andrew|1991|       34000|
|  8|andrew|1992|       34000|
|  9|thomas|1991|       34000|
|  9|thomas|1992|       68000|
+---+------+----+------------+
only showing top 20 rows
```

Assignment 20.1
Spark SQL 1

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

We have used **sql** transformation to use sql query. In sql query, we have used join clause to join **holidays** and **users** tables and then used group by clause on year and age for each category to take sum of distance and then used union all clause to combine these three categories. Then created **users_age_wise** table from that dataframe.

We have taken sum of distance by using group by clause on year and age columns on **users_age_wise** table and created **users_age_new** table from this dataframe.

Then we have taken maximum of revenue by using group by clause on year on **users_age_new** table and created **maximum_distance** table from this dataframe.

After this we have used join clause to join **maximum_distance** and **users_age_new** tables and then printed the age group which is travelling the most every year.

Scala Code :

```
println("Below is the age group who is travelling the most every year.")

val age_sql_new = spark.sql("select h.year,'age < 20' age, sum(h.distance)
total_distance from holidays h join users u on h.id = u.id where u.age < 20 group
by h.year,age union all select h.year,'age between 20 & 35' age,sum(h.distance)
total_distance from holidays h join users u on h.id = u.id where u.age between 20
and 35 group by h.year,age union all select h.year,'age > 35' age,sum(h.distance)
total_distance from holidays h join users u on h.id = u.id where u.age > 35 group
by h.year,age ").toDF()

age_sql_new.createOrReplaceTempView("users_age_wise")

val age_sql_new2 = spark.sql("select * from users_age_wise")



val age_wise_group_by_sql = spark.sql("select sum(total_distance)
max_distance,age,year from users_age_wise group by age,year").toDF()

age_wise_group_by_sql.createOrReplaceTempView("users_age_new")

val age_sql_new3 = spark.sql("select * from users_age_new")



val age_wise_group_by_sql2 = spark.sql("select max(max_distance)
max_total_distance,year from users_age_new group by year").toDF()

age_wise_group_by_sql2.createOrReplaceTempView("maximum_distance")

val age_wise_group_by_sql7 =  spark.sql("select * from maximum_distance")


val age_wise_group_by = spark.sql("select a.age,a.year, a.max_distance from
users_age_new a join maximum_distance b on b.max_total_distance  =a.max_distance
and a.year = b.year order by a.year")

age_wise_group_by.show()
```

Assignment 20.1
Spark SQL 1

Output :

```
Below is the age group who is travelling the most every year.
+------------------+----+------------+
|               age|year|max_distance|
+------------------+----+------------+
|age between 20 & 35|1990|        1000|
|age between 20 & 35|1991|         800|
|           age > 35|1992|         800|
|           age < 20|1993|        1000|
|age between 20 & 35|1994|         200|
+------------------+----+------------+
```

Assignment 20.1
Spark SQL 1

Complete Scala code :

```
import org.apache.spark.sql.SparkSession

  object Assignment_20_Spark_SQL_1  {

    case class holidays(id: Int, source: String, destination: String,
transport_mode: String, distance: Int, year: Int)

    case class transport(transport_mode : String, cost_per_unit : Int)

    case class users(id : Int, name : String, age : Int)


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


      val spark = SparkSession
        .builder()
        .master("local")
        .appName("Spark SQL Assignment ")
        .config("spark.some.config.option", "some-value")
        .getOrCreate()

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

      println("Spark Session Object created")



      val holidayFile = spark.sparkContext.textFile("C:\\AcadGild
Hadoop\\Assignments\\S20_Dataset_Holidays.csv");

      import spark.implicits._

      val  holidayDF =holidayFile
        .map( x => x.split(",")).map(x =>
holidays(x(0).trim.toInt,x(1),x(2),x(3),x(4).trim.toInt,x(5).trim.toInt)).toDF()

      //holidayDF.show()

      holidayDF.registerTempTable("holidays")

      println("holidays table is created ")

      val TransportFile = spark.sparkContext.textFile("C:\\AcadGild
Hadoop\\Assignments\\S20_Dataset_Transport.csv");

      val TransportDF = TransportFile.map(x => x.split(",")).map(x =>
transport(x(0),x(1).trim.toInt)).toDF

      TransportDF.registerTempTable("transport")

      println("transport table is created ")

      val usersFile = spark.sparkContext.textFile("C:\\AcadGild
Hadoop\\Assignments\\S20_Dataset_User_details.csv")

      val usersDF = usersFile.map(x => x.split(",")).map(x =>
users(x(0).trim.toInt,x(1),x(2).trim.toInt)).toDF()

      usersDF.registerTempTable("users")

      println("users table is created ")
```

Assignment 20.1
Spark SQL 1

```
println("Below is the distribution of the total number of air-travelers per year")

val holidays_sql = spark.sql("select year,count(*) no_of_air_travelers from holidays group by year order by year")

holidays_sql.show()



println("Below is the total air distance covered by each user per year")

val users_sql = spark.sql("select u.id,u.name,h.year,sum(h.distance) total_distance from holidays h join users u on h.id = u.id group by u.id,u.name,h.year order by u.id,u.name,h.year")

users_sql.show()



println("Below is the user who has travelled the largest distance till date")

val max_distance_sql = spark.sql("select h.id,u.name,sum(distance) total_distance from holidays h join users u on u.id = h.id group by h.id,u.name order by total_distance desc limit 1")

max_distance_sql.show()



println("Below is the most preferred destination for all users")

val destination_sql = spark.sql("select destination,count(*) max_count from holidays group by destination order by max_count desc limit 1")

destination_sql.show()



println("Below is the total amount spent by every user on air-travel per year")

val transport_sql = spark.sql("select * from transport")

val total_amount_sql = spark.sql("select h.id,u.name,year,sum(distance*t.cost_per_unit) total_amount from holidays h join transport t join users u on h.transport_mode = t.transport_mode and u.id = h.id where h.transport_mode = 'airplane'  group by h.id,u.name,year order by h.id,u.name,year")

total_amount_sql.show()


println("Below is the route which is generating the most revenue per year")

val max_revenue_sqlDF = spark.sql("select id,source,destination,h.transport_mode,distance,year,cost_per_unit from transport t join holidays h on h.transport_mode = t.transport_mode").toDF()

max_revenue_sqlDF.createOrReplaceTempView("revenue")


val revenue_sql = spark.sql("select sum(distance*cost_per_unit) totalRevenue, year,concat(source,destination) route from revenue group by year,concat(source,destination) order by year").toDF()

revenue_sql.createOrReplaceTempView("total_revenue")
```

```
      val revenue_sql2 = spark.sql("select * from total_revenue")


      val max_revenue_sql = spark.sql("select max(totalRevenue) max_revenuePerDest,
year from total_revenue group by year order by year").toDF()

      max_revenue_sql.createOrReplaceTempView("max_revenue_per_dest")

      val revenue_sql3 = spark.sql("select * from max_revenue_per_dest")


      val max_revenue_sql2 = spark.sql("select a.route, a.year,a.totalRevenue from
total_revenue a join max_revenue_per_dest b on a.totalRevenue=b.max_revenuePerDest
and a.year = b.year order by a.year asc, a.totalRevenue desc").toDF()

      max_revenue_sql2.createOrReplaceTempView("max_revenue")

      val revenue_sql4 = spark.sql("select * from max_revenue")

      revenue_sql4.show()


      println("Below is the age group who is travelling the most every year.")

      val age_sql_new = spark.sql("select h.year,'age < 20' age, sum(h.distance)
total_distance from holidays h join users u on h.id = u.id where u.age < 20 group
by h.year,age union all select h.year,'age between 20 & 35' age,sum(h.distance)
total_distance from holidays h join users u on h.id = u.id where u.age between 20
and 35 group by h.year,age union all select h.year,'age > 35' age,sum(h.distance)
total_distance from holidays h join users u on h.id = u.id where u.age > 35 group
by h.year,age ").toDF()

      age_sql_new.createOrReplaceTempView("users_age_wise")

      val age_sql_new2 = spark.sql("select * from users_age_wise")


      val age_wise_group_by_sql = spark.sql("select sum(total_distance)
max_distance,age,year from users_age_wise group by age,year").toDF()

      age_wise_group_by_sql.createOrReplaceTempView("users_age_new")

      val age_sql_new3 = spark.sql("select * from users_age_new")


      val age_wise_group_by_sql2 = spark.sql("select max(max_distance)
max_total_distance,year from users_age_new group by year").toDF()

      age_wise_group_by_sql2.createOrReplaceTempView("maximum_distance")

      val age_wise_group_by_sql7 =  spark.sql("select * from maximum_distance")


      val age_wise_group_by = spark.sql("select a.age,a.year, a.max_distance from
users_age_new a join maximum_distance b on b.max_total_distance  =a.max_distance
and a.year = b.year order by a.year")

      age_wise_group_by.show()

    }

  }
```

Assignment 20.1
Spark SQL 1

Complete Output :

```
Spark Session Object created
holidays table is created
transport table is created
users table is created


Below is the distribution of the total number of air-travelers per year
+----+-------------------+
|year|no_of_air_travelers|
+----+-------------------+
|1990|                  8|
|1991|                  9|
|1992|                  7|
|1993|                  7|
|1994|                  1|
+----+-------------------+



Below is the total air distance covered by each user per year
+---+------+----+--------------+
| id|  name|year|total_distance|
+---+------+----+--------------+
|  1|  mark|1990|           200|
|  1|  mark|1993|           600|
|  2|  john|1991|           400|
|  2|  john|1993|           200|
|  3|  luke|1991|           200|
|  3|  luke|1992|           200|
|  3|  luke|1993|           200|
|  4|  lisa|1990|           400|
|  4|  lisa|1991|           200|
|  5|  mark|1991|           200|
|  5|  mark|1992|           400|
|  5|  mark|1994|           200|
|  6| peter|1991|           400|
|  6| peter|1993|           200|
|  7| james|1990|           600|
|  8|andrew|1990|           200|
|  8|andrew|1991|           200|
|  8|andrew|1992|           200|
|  9|thomas|1991|           200|
|  9|thomas|1992|           400|
+---+------+----+--------------+
only showing top 20 rows


Below is the user who has travelled the largest distance till date
+---+----+--------------+
| id|name|total_distance|
+---+----+--------------+
|  1|mark|           800|
+---+----+--------------+



Below is the most preferred destination for all users
+-----------+---------+
|destination|max_count|
+-----------+---------+
|        IND|        9|
+-----------+---------+
```

Assignment 20.1
Spark SQL 1

Below is the total amount spent by every user on air-travel per year

```
+---+------+----+------------+
| id|  name|year|total_amount|
+---+------+----+------------+
|  1|  mark|1990|       34000|
|  1|  mark|1993|      102000|
|  2|  john|1991|       68000|
|  2|  john|1993|       34000|
|  3|  luke|1991|       34000|
|  3|  luke|1992|       34000|
|  3|  luke|1993|       34000|
|  4|  lisa|1990|       68000|
|  4|  lisa|1991|       34000|
|  5|  mark|1991|       34000|
|  5|  mark|1992|       68000|
|  5|  mark|1994|       34000|
|  6| peter|1991|       68000|
|  6| peter|1993|       34000|
|  7| james|1990|      102000|
|  8|andrew|1990|       34000|
|  8|andrew|1991|       34000|
|  8|andrew|1992|       34000|
|  9|thomas|1991|       34000|
|  9|thomas|1992|       68000|
+---+------+----+------------+
only showing top 20 rows
```

Below is the route which is generating the most revenue per year

```
+------+----+------------+
| route|year|totalRevenue|
+------+----+------------+
|CHNIND|1990|       68000|
|INDRUS|1991|       68000|
|INDAUS|1991|       68000|
|CHNRUS|1992|       68000|
|RUSIND|1992|       68000|
|AUSCHN|1993|       68000|
|CHNIND|1993|       68000|
|CHNPAK|1994|       34000|
+------+----+------------+
```

Below is the age group who is travelling the most every year.

```
+------------------+----+------------+
|               age|year|max_distance|
+------------------+----+------------+
|age between 20 & 35|1990|        1000|
|age between 20 & 35|1991|         800|
|          age > 35|1992|         800|
|          age < 20|1993|        1000|
|age between 20 & 35|1994|         200|
+------------------+----+------------+
```