Project 3
Problem 1 description
There were issues to create a maven project and use scala module in it, thats why immediately created a new spark project in intelij and then designed the EventDataGenrator file, which is responsible for creating the CSV fils required by the problem description. As this is the first time I have written Scala code, I used charGPT to help for synta,x but I wrote the codes myself, and I understand each single line of the code. I changed the main method to main1 so I can run the next steps. The generated files are big as said in the exercise description, the sick people are about 1% of them. The generated csv files had long names, we just manually renamed them to make it easy to work with them.

For query 1

We created a Q1.scala object file, there initialize the spark session, loaded the mega-event.csv file, and applied a simple filter on it $"test" === "sick" to get only people who have test == sick, then we used the coalesce method to save the result into a new file, we also calculated the total time of the job. 4.2 seconds.

For query 2

The Q2.scala file was created, two datasets were loaded (Mega-Event-No-Disclosure.csv and ReportedIllnesses.csv) and inner join was performed on it on the id column, time 4.7 seconds. The name of the main method we changes into main2

Query 3

In the Q3.scala we designed map reduce solution,

Map phase - we use the map transformation to convert each line of the CSV into a tuple: (table, (id, test)).

groupByKey(): This groups the data by the table field. The result is an RDD of type (table, Iterable[(id, test)]), meaning we have all people at each table in a list.

Reduce step - for each table, we filter out the sick people, If the table has at least one sick person, we emit the healthy people's IDs. Save the results in the file. The results are written in a txt file this time. The execution time is 4.4 secund which is a very good result.
The name of the method “main” we changed into “main3”

Query 4

We load the csv file as rdd omitting the header,
Map
Extract table and health status, map to (1, isHealthy)
Reduce
Aggregate by table to count people and check if all are healthy.
Format results into (table, count, flag).
At the end we changed the main method into main4
Execution time 1.8 sec

Query5

Each record from the mega event file is mapped to a key-value pair where the key is the table and the value is a tuple (id, name, isSick), with isSick set by checking the broadcasted sick IDs.For each table, we use reduceByKey to combine the Boolean flags (using logical OR) to determine if the table had any sick person.
We then collect the list of tables with at least one sick individual and broadcast it.
We filter the mega event records to select only those entries where:The table is among those with sick individuals And The person’s ID is not in the sick set (indicating the person is healthy).\
Execution time - 2.43