A01 – EX4 REPORT

Full Day Tour Plan:

We are doing a full day tour from 10AM to 5PM on a given date

Lunch break is for 1 hour probably between 1PM - 2PM

Major tourist spots in the city are at stops AA, BB, CC, DD (example 1921, 395, 1445, 7459) and we are planning to spend 1 hour each in these 4 spots

We are at a stop XX (Current stop: 1935), so the plan is to hop in the first bus arriving at XX, and then find the list of busses and their timings to get to spot AA, we spend 1 hour at spot AA and then decide to hop on the very next available bus/multiple busses to go to spot BB, likewise for spot CC and DD.

The dataset we have in our hands certainly allows us to infer this information

Formal Definition:

Current time: 2013-01-10 9:59:59

Current stop: 1935

Major tourist spots: 1921, 395, 1445, 7459

Lunch start: 2013-01-10 12:59:59

Our task:

To find out the optimal busses to board to start and finish the day tour on time considering all the time requirements at each tourist spot.

The solution should be as follows:

1. Current stop:
    a. [Time1, Bus_Stop_1, Vehicle_ID_1]
    b. [Time2, Bus_Stop_2, Vehicle_ID_2]
    c. [Time3, Tourist_Spot_1, Vehicle_ID_3]
    d. <1 hour delay to roam around Tourist_Spot_1>
2. Tourist_Spot_1:
    a. [Time4, Bus_Stop_4, Vehicle_ID_4]
    b. [Time4, Bus_Stop_5, Vehicle_ID_5]
    c. [Time6, Tourist_Spot_2, Vehicle_ID_6]
    d. <1 hour delay to roam around Tourist_Spot_2>

This pattern continues for all the major tourist spots (note: the Vehicle ID could be the same or different depending on the BusID and BusLine)

The main steps to implement this operation are:

- To find the first bus arriving at the current_stop
- To find if the same busID would take you to the tourist_spot1
- If not, find out which stop to get down and which next available bus is the best to board into inorder to go to trousist_spot_1, we might have to find a bus which might be different busID and might not take us to tourist_spot_1, but somewhere closer to it from where we have to get another bus to get to the actual spot
- Give delay of 1 hour at the spot
- Search for the next arriving bus at the tourist_spot_1 and repeat the same steps above to find best possible bus/busses to get to spot_2
- The lunch break delay of 1 hour should be taken care of while analysing the bus data

If I was to implement this exercise, I will go for Spark SQL because from my experience working on other exercises in this assignment, Spark SQL was almost 3 times quicker than Spark Core, for example the exercise 1 took around 5 mins on Core and 2 mins on SQL.

The exercise above is very difficult compared to the other exercises (1, 2 and 3) since there is a lot of analysis to be done on the data to find the optimal busses and stops to get down and get into the bus at the right time. I might have missed many steps in my implementation above as there is too much to consider when designing this exercise, the whole point of the exercise is for someone who wants to go to 4 different spots in four different locations in a day, and would like to have a list of busses which can be boarded at convenient times (I got this idea from the Cork City Tour Bus which is owned by Dermot Cronin Coaches, they have these city tour red coloured open roof bus which can be boarded at Grand Parade, then it takes you to different locations around the city where you can get down and walk around (the bus will carry on their travel with other passengers) once you feel like you have seen enough and are planning on getting back to the bus, you can hop in the very next red coloured city tour bus using the same ticket you got earlier, one time pay at Grand Parade.