PSTR - Project 1

André Neves 46264, Daniel Henriques 46144, Ricardo Loureiro 45351

May 2019

1 Taxi trips

Figure 1: Hey taxi!

2 Grid system

For the creation of the 300x300 grid system we created a stream *RoutesStream300* shown in listing 1, which takes the input stream data and transfers pick-up and drop-off coordinates into cells.

We used the flat earth assumption as recommended. If cell (1,1) is at 41.474937, -74.913585, and a 500m displacement equals to 0.004491556 degrees if south or 0.005986 degrees if east, then we simply calculate the distance between two coordinates in degrees, and divide it by the correspondent degree change. Which gives us distance between cells in 500m intervals, then we simply add 1 to the result and get a coordinate system that has (1,1) as 41.474937, -74.913585.

After the conversion there is a filter which filter data that falls outside the 300x300 grid.

For the 600x600 grid we used the same code but divided the displacement degrees,0.004491556 and 0.005986 degrees, by two. Giving us 250m displacement, which corresponds to a a grid with half the cell size of the original.

There is then a filter which filter data that falls outside the 600x600 grid.

Listing 1: 300x300 Grid conversion stream

```
@info(name='route_query_300')
from TaxiRidesProductionStream
select medallion as carId,
convert( -1*((-74.913585 - pickup_longitude) / 0.005986 ) + 1 , 'int') as
     start_cell_x,
convert(((41.474937 - pickup_latitude)/ 0.004491556) + 1 , 'int') as
    start_cell_y,
convert( -1*((-74.913585 - dropoff_longitude) / 0.005986 ) + 1 , 'int')
    as ending_cell_x,
convert(((41.474937 - dropoff_latitude)/ 0.004491556) + 1 , 'int') as
    ending_cell_y
having (start_cell_x >= 1 and start_cell_x <= 300) and
(start_cell_y >= 1 and start_cell_y <= 300) and
(ending_cell_y >= 1 and ending_cell_y <= 300) and
(ending_cell_x >= 1 and ending_cell_x <= 300)
insert into RoutesStream300;
```

3 Queries

3.1 Q1: Find the top 10 most frequent routes during the last 30 minutes

For Q1, we take the data from *RoutesStream300* and aggregate it by ending_cell_x, ending_cell_y, start_cell_x, start_cell_ym , this means that we have a route for every existent two cells.

We count the number of cars in each cell combo and create a *FrequentRoutesStream* stream, which contains for every combination of cells, which represent a route, the number of trips made. A time window of 30 minutes is also added (*#window.time(30 min)*).

Finally we simply take the data from that stream and apply a item sorted window (*#window.sort(10, nTrips, 'desc')* to get the final results, shown in listing 2.

Listing 2: Q1 output stream events

```
...
[2019-05-10_15-32-28_555] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502348555, data=[156, 155, 157,
    153, 1], isExpired=false}
```

```
[2019-05-10_15-32-28_555] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502348555, data=[164, 160, 164,
    160, 4], isExpired=false}
[2019-05-10_15-32-28_556] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502348555, data=[184, 179, 190,
    185, 1], isExpired=false}
[2019-05-10_15-32-28_556] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502348556, data=[156, 165, 160,
    159, 1], isExpired=false}
[2019-05-10_15-32-29_547] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349547, data=[157, 161, 154,
    168, 1], isExpired=false}
[2019-05-10_15-32-29_587] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349587, data=[159, 169, 160,
    169, 1], isExpired=false}
[2019-05-10_15-32-29_587] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349587, data=[186, 172, 186,
    172, 2], isExpired=false}
[2019-05-10_15-32-29_588] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349588, data=[160, 159, 175,
    157, 1], isExpired=false}
[2019-05-10_15-32-29_660] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349660, data=[151, 171, 157,
    161, 1], isExpired=false}
[2019-05-10_15-32-29_660] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349660, data=[153, 160, 154,
    168, 1], isExpired=false}
[2019-05-10_15-32-29_676] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349676, data=[153, 166, 155,
    164, 1], isExpired=false}
[2019-05-10_15-32-29_677] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q1 : Event{timestamp=1557502349676, data=[163, 167, 160,
    171, 1], isExpired=false}
...
```

3.2 Q2: Identify areas that are currently most profitable for taxi drivers

For this query we use a 600x600 grid.

There are are two ways we made the *FinalRoutesStream600*, first, assuming the events arrive in order and with simulated timestamp, we create a *FinalRoutesStream600*, which takes the *RoutesStream600* data and checks for taxis within a 15 minute window that do not have a new pickup event. Meaning the taxi did not pick anyone up. The fare and tip data are then stored for future processing.

The second way was to calculate the difference between new event pickup time and the dropoff time of the original event, and then filtering those whose difference is bigger than 15 minutes. The first stream is commented in the code and the latter is the one used to produce the output.

Secondly, we create a *AreaProfitStream*, which takes the *RoutesStream600* data and aggregates the profit by starting cell, averaging the fate + tip for each ending within the last 15 minutes.

We create an aditional *EmptyTaxisStream*, which simply takes data from *FinalRoutesStream600* and counts the number of empty taxis on each cell in the last 30 minutes.

Finally, we join both *AreaProfitStream*-¿A and *EmptyTaxisStream*-¿E with 15 minutes window and calculate the profitability as (A.profit / E.emptyTaxis).

Results are present on listing 3.

Listing 3: Q2 output stream events

```
[2019-05-10_16-40-33_218] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : Event{timestamp=1557506433218, data=[304, 334,
    8.085714285714285], isExpired=false}
[2019-05-10_16-40-33_237] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : Event{timestamp=1557506433237, data=[313, 322,
    10.13653846153846], isExpired=false}
[2019-05-10_16-40-34_138] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : Event{timestamp=1557506434137, data=[313, 322,
    10.075925925925924], isExpired=false}
[2019-05-10_16-40-34_241] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : Event{timestamp=1557506434240, data=[311, 324,
    7.814705882352941], isExpired=false}
[2019-05-10_16-40-34_270] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : Event{timestamp=1557506434270, data=[311, 324,
    8.012777777777778], isExpired=false}
[2019-05-10_16-40-34_284] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : [Event{timestamp=1557506434282, data=[310, 323,
    4.0], isExpired=false}, Event{timestamp=1557506434282, data=[310,
    323, 5.65], isExpired=false}, Event{timestamp=1557506434282, data
    =[310, 323, 5.1000000000000005], isExpired=false}, Event{timestamp
    =1557506434282, data=[310, 323, 5.45], isExpired=false}, Event{
    timestamp=1557506434282, data=[310, 323, 5.16], isExpired=false},
    Event{timestamp=1557506434282, data=[310, 323, 5.55], isExpired=
    false}, Event{timestamp=1557506434282, data=[310, 323,
    5.8999999999999995], isExpired=false}, Event{timestamp
    =1557506434282, data=[310, 323, 6.2875], isExpired=false}, Event{
    timestamp=1557506434282, data=[310, 323, 6.866666666666666],
    isExpired=false}, Event{timestamp=1557506434282, data=[310, 323,
    6.7299999999999995], isExpired=false}, Event{timestamp
    =1557506434282, data=[310, 323, 7.163636363636363], isExpired=false
    }, Event{timestamp=1557506434282, data=[310, 323,
    7.6499999999999995], isExpired=false}, Event{timestamp
    =1557506434282, data=[310, 323, 7.869230769230769], isExpired=false
    }, Event{timestamp=1557506434282, data=[310, 323,
    7.914285714285714], isExpired=false}, Event{timestamp=1557506434282,
     data=[310, 323, 8.22], isExpired=false}]
[2019-05-10_16-40-34_301] INFO {org.wso2.siddhi.core.stream.output.sink.
```

```
    LogSink} - Q2 : Event{timestamp=1557506434299, data=[311, 324,
    7.722631578947368], isExpired=false}
[2019-05-10_16-40-34_367] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q2 : [Event{timestamp=1557506434362, data=[317, 315,
    6.5], isExpired=false}, Event{timestamp=1557506434362, data=[317,
    315, 7.6899999999999995], isExpired=false}, Event{timestamp
    =1557506434362, data=[317, 315, 7.46], isExpired=false}, Event{
    timestamp=1557506434362, data=[317, 315, 8.02], isExpired=false}]
```

3.3 Q3: Alert whenever the average idle time of taxis is greater than a given amount of time (say 10 minutes)

For this query we use the same system as Q1, in which we have a *TimeBetweenDropsAndPicksStream*, calculated between pickup and dropoff times of two events.

Then we filter this stream and create a *AlertIdleTimeExceededStream* to only have taxis which had a trip in the last hour, and average all the rows idle time with a filter to only alert when this idle time is bigger than 10 minutes.

Results are present in listing 4, with a mean idle time threshold of 30 seconds for simplicity only.

Listing 4: Q3 output stream events

```
[2019-05-10_17-12-13_522] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508333522, data=[60000.0],
    isExpired=false}
[2019-05-10_17-12-15_821] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508335821, data=[80000.0],
    isExpired=false}
[2019-05-10_17-12-16_821] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508336821, data=[75000.0],
    isExpired=false}
[2019-05-10_17-12-16_822] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508336822, data=[72000.0],
    isExpired=false}
[2019-05-10_17-12-17_901] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508337901, data=[70000.0],
    isExpired=false}
[2019-05-10_17-12-17_919] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508337919, data
    =[94285.71428571429], isExpired=false}
[2019-05-10_17-12-17_964] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508337964, data=[97500.0],
    isExpired=false}
[2019-05-10_17-12-17_978] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508337978, data
    =[126666.66666666667], isExpired=false}
[2019-05-10_17-12-19_039] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508339039, data
    =[125454.54545454546], isExpired=false}
[2019-05-10_17-12-19_061] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q3 : Event{timestamp=1557508339061, data=[120000.0],
    isExpired=false}
```

3.4 Q4: Detect congested areas

For this query we use a 300x300 grid.

For this query we simply create a *CongestedAreasStream*, which takes the 300x300 grid data and waits for an increasing continuous trip duration of the same car, by car.

The pattern used is as follows, all the events have the same carId: E1 - E2(duration increases) - E3(duration increases) - E4(duration increases)

The results are represented on listing 5.

Listing 5: Q4 output stream events

```
[2019-05-10_17-20-43_390] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q4 : Event{timestamp=1557508843390, data=[156, 156],
    isExpired=false}
[2019-05-10_17-20-45_449] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q4 : Event{timestamp=1557508845449, data=[156, 167],
    isExpired=false}
[2019-05-10_17-20-47_757] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q4 : Event{timestamp=1557508847757, data=[152, 167],
    isExpired=false}
[2019-05-10_17-20-48_724] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Q4 : Event{timestamp=1557508848724, data=[160, 169],
    isExpired=false}
```

3.5 Q5: Select the most pleasant taxi drivers

For query 5, we create a *DailyTipsStream*, which sums the total amount of daily tips for each driver, we assume each driver only drives one car, and one car is only driver by one driver, represented by carId. The stream has a timebatch tumbling window for the 24h spacing of results.

We then create a *DailyTipsStream* which simple creates a sorted windows with 1 element, this gives us the most pleesent driver by the day.

Results are shown on listing 6, a time batch windows of 1 minute is used for the production of the data:

Listing 6: Q5 3output stream events

```
Put results here.
```

4 Extra

4.1 Q6 - Tips hall of fame!

It may be interesting to know the biggest tip per cell, creating a client hall of fame of tips!

For this we use a 300x300 grid, we create a *BiggestTipsStream*, which keeps the maximum tip value of each cell for the duration of the query by using the maxForever aggregation.

Results are present on listing 7:

Listing 7: Q6 output stream events

```
[2019-05-11_12-48-13_943] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578893943, data=[160, 155, 10.4], isExpired=false}
[2019-05-11_12-48-13_943] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578893943, data=[162, 156, 6.9], isExpired=false}
[2019-05-11_12-48-13_943] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578893943, data=[156, 167, 10.4], isExpired=false}
[2019-05-11_12-48-13_943] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578893943, data=[152, 164, 25.7], isExpired=false}
[2019-05-11_12-48-13_943] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578893943, data=[164, 162, 5.1], isExpired=false}
[2019-05-11_12-48-13_943] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578893943, data=[160, 159, 7.5], isExpired=false}
[2019-05-11_12-48-14_944] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578894944, data=[152, 172, 6.1], isExpired=false}
[2019-05-11_12-48-14_944] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578894944, data=[153, 175, 1.0], isExpired=false}
[2019-05-11_12-48-14_944] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578894944, data=[155, 165, 16.0], isExpired=false}
[2019-05-11_12-48-14_944] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578894944, data=[156, 164, 10.0], isExpired=false}
[2019-05-11_12-48-14_944] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578894944, data=[157, 156, 10.0], isExpired=false}
[2019-05-11_12-48-14_944] INFO {org.wso2.siddhi.core.stream.output.sink.LogSink} - req : Event{timestamp=1557578894944, data=[157, 163, 10.0], isExpired=false}
```

4.2 Q7 - Tip forecast of the last 30 minutes

It may be interesting to get a tip amount forecast

We use as tip_amount the independent variable and trip_time_in_secs, trip_distance, fare_amount as the dependant variable.

Using *#timeseries:regress* with a size of 15 and a confidence interval of 0.95, we obtain the beta coefficients $\beta0, \beta1, \beta2, \beta3$ and the error ϵ.

Then we build the equation with a window of 30 minutes:

$$Y = \beta0 + \beta1 * x1 + \beta2 * x2 + \beta3 * x3 + \epsilon$$

Results are presented on listing 8

Listing 8: Q7 output stream events

```
[2019-05-11_12-16-57_747] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577017746, data=[B
    83044D63E9421B76011917CE280C137, 1.921992156758505], isExpired=false
    }
[2019-05-11_12-16-57_752] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577017750, data=[
    CBDAFC4B9364E4DFE6712AA1AB188C08, 2.022971360539633], isExpired=
    false}
[2019-05-11_12-16-57_754] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577017753, data=[DB7
    FA1855AAF51425A2A1FAAFCD83D36, 1.9907940918815326], isExpired=false}
[2019-05-11_12-16-57_756] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577017755, data=[F151
    C67B2D912D2F29931711B28204F4, 2.0456190504035923], isExpired=false}
[2019-05-11_12-17-05_990] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577025984, data=[1390
    FB380189DF6BBFDA4DC847CAD14F, 1.663971140230514], isExpired=false}
[2019-05-11_12-17-08_069] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577028063, data=[655E
    773C92FA446353D5C8B7416BE818, 1.8347500790288158], isExpired=false}
[2019-05-11_12-17-10_121] INFO {org.wso2.siddhi.core.stream.output.sink.
    LogSink} - Tip forecast: : Event{timestamp=1557577030119, data
    =[7166914A326D849158A5161E811ADDEA, 1.779950556742482], isExpired=
    false}
```