PROJECT MILESTONE-5- HOTSPOT ANALYSIS

Team#19 Christopher Selby Haishan Luo, Haritha Gurram , Jeffrey Rothenberg,Jessica Harris

Hot zone analysis:

Two input data sets were provided.

1. Point_hotzone.csv, which provided all the pick-up points from NYC taxi dataset.
2. Zone_hotzone.csv, provides rectangular zone points.

Created a function a UDF *ST_Contains* in *HotzoneUtils* which takes 2 inputs a string of rectangle co-ordinates and a string with point co-ordinates are given as input, returns true if a given point is present in the given rectangular co-ordinates.

In *HotzoneAnalysis*, following steps are performed to get the desired output.

1. Loaded pickup points data into a data frame called *pointDf* from Point_hotzone.csv, and created a temp table '*point'*.
2. Loaded rectangle points data into a data frame called *rectangleDf* from Zone_hotzone.csv, and created a temp talble '*rectangle'*
3. A range join query is performed on point and rectangle tables with *ST_Contains* in filter condition, the resultant data frame '*joinDf* 'contained rectangle points and points within it.
4. Final step is to aggregate on number of points that are in a rectangle zone.

Hot Cell Analysis:

1. Required data the pickup point, date and time are loaded from input file into the data frame '*pickupInfo'* and a temp table called '*nyctaxitrips'* is created.
2. A co-ordinated off set of 0.01 is considered as part of problem statement, used UDF's to create a space-time cube with (x,y,z) co-ordinates. (x,y) denotes the pickup point and z denotes the time, here the given day of the pickup(1 to 31 days).
3. To calculate the hotcell Z-score, a boundary area was given with boundary values for (x,y,z). The whole boundary needs to be considered to calculate the neighbor's and spatial weight.
4. The $x_j$represents the number of trips that took place on a particular day at a given pickup point, this is calculated and stored in *tripsCountData* dataframe that contains the aggregate of count of trips and a temp table called *totaltripsCountData* is created.
5. Standard deviation and Mean are calculated based on the Total trips count and number of cells in the given boundary.
6. Next step is to calculate $\sum_{j=1}^{n} w_{i,j} x_j$ the sum of trips of all the neighbors cell(x-1,y-1,z-1) and (x+1,y+1,z+1) and the cell itself since $w_{i,j}$ is 1, which is calculated by cross joining the table *totaltripsCountData* with predicates to check if a cell is a neighbor or not. The resultant data with neighbours trip data is stored in *neighboursData.*
7. A UDF *CalculateWeight* is used to calculate spatial weight $\sum_{j=1}^{n} w_{i,j}$, $\sum_{j=1}^{n} w_{i,j}^2$ and $(\sum_{j=1}^{n} w_{i,j})^2$ is the number of neighbour's for a given cell based on its location. Corner, Edge, Face, Inner cells have 8, 12,18 and 27 neighbors' respectively. (considering a given cell is a neighbor for it-self).
8. Finally, UDF *CaluclateGScore* is implemented to calculate the scores with Mean, Standard deviation, neighboursData and number of neighbours.