|  |  |  |
| --- | --- | --- |
|  |  | Challenge:  Data consists of taxi trips including starting point, drop-off point, corresponding timestamps, and information related to the payment. Data are reported at the end of the trip, i.e., upon arrive in the order of the drop-off timestamps. |
| KiNGSTON  Title: Big Data Challenge **CONTACT** PHONE:  65- 8286 9296  EMAIL:  kingstonmsc@gamil.com |  |

## Attributes: problem DESCRIPTION

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **medallion** | an md5sum of the identifier of the taxi - vehicle bound | | **hack\_license** | an md5sum of the identifier for the taxi license | | **pickup\_datetime** | time when the passenger(s) were picked up | | **dropoff\_datetime** | time when the passenger(s) were dropped off | | **trip\_time\_in\_secs** | duration of the trip | | **trip\_distance** | trip distance in miles | | **pickup\_longitude** | longitude coordinate of the pickup location | | **pickup\_latitude** | latitude coordinate of the pickup location | | **dropoff\_longitude** | longitude coordinate of the drop-off location | | **dropoff\_latitude** | latitude coordinate of the drop-off location | | **payment\_type** | the payment method - credit card or cash | | **fare\_amount** | fare amount in dollars | | **surcharge** | surcharge in dollars | | **mta\_tax** | tax in dollars | | **tip\_amount** | tip in dollars | | **tolls\_amount** | bridge and tunnel tolls in dollars | | **total\_amount** | total paid amount in dollars | | The data is sorted chronologically according to the dropoff\_datetime. Events with the same dropoff\_datetime are in random order. Please note that the quality of the data is not perfect. Some events might miss information such as drop off and pickup coordinates or fare information. Moreover, some information, such as, e.g., the fare price might have been entered incorrectly by the taxi drivers thus introducing additional skew. Handle accordingly.  From the data, please answer the following query (assume the data is streaming):  Find the top 10 most frequent routes during the last 30 minutes. A route is represented by a starting grid cell and an ending grid cell. All routes completed within the last 30 minutes are considered for the query. The output results must be updated whenever any of the 10 most frequent routes changes. The output format for the result stream is:  ***pickup\_datetime, dropoff\_datetime, start\_cell\_id\_1, end\_cell\_id\_1, ... , start\_cell\_id\_10, end\_cell\_id\_10, delay***  where pickup\_*datetime*, dropoff\_*datetime* are the timestamps of the trip report that resulted in an update of the result stream, start\_cell\_id\_X the starting cell of the Xth-most frequent route, end\_cell\_id\_X the ending cell of the Xth-most frequent route. If less than 10 routes can be identified within the last 30 min, then NULL is to be output for all routes that lack data.  The attribute ***delay*** captures the time delay between reading the input event that triggered the output and the time when the output is produced. Participants must determine the delay using the current system time right after reading the input and right before writing the output. This attribute will be used in the evaluation of the submission.  The cells for this query are squares of 500 m X 500 m. The cell grid starts with cell 1.1, located at 41.474937, -74.913585 (in Barryville). The coordinate 41.474937, -74.913585 marks the center of the first cell. Cell numbers increase towards the east and south, with the shift to east being the first and the shift to south the second component of the cell, i.e., cell 3.7 is 2 cells east and 6 cells south of cell 1.1. The overall grid expands 150km south and 150km east from cell 1.1 with the cell 300.300 being the last cell in the grid. All trips starting or ending outside this area are treated as outliers and must not be considered in the result computation. |

## SAMPLE DATA

|  |  |
| --- | --- |
| Solution EValuation  STEP 1:  Taxi booking apps used to capture Peer to peer ride information via GPS to track the customer travel location and information stored in server to analysis and monitoring of ride data. | High Level Contextual Diagram  DB – Booking  Info  Booking App  GPS |
| STEP 2  Two way of approach to solving this problem  2.1 Data Management  2.2 Clustering Technics ( Most accurate and recommended Solution) | High-level Structure    Streaming Model –K means  Model of Computation  Streaming Output  Spark Job  Clean  HDFS  Data Streaming (Kafka)  Data Management  Database Design & Model |
| STEP: 3.1 LOGICAL iMPLEMENTATION  point 2.1 data Management | Job schedule on every 30 minutes  Read the last 30 mins data from table  Select the require attributes in Dataset  Round the distance and aggregate the trip by location & limit the top 10 and send to Strem  Calculate the trip distance and location  Clean the missing Data like drop off and pickup coordinates or fare information |

Read the last 30 mins data from table

STEP: 3.1 LOGICAL iMPLEMENTATION

Point 2.2 Clustering Technics

Grouping & visualizing no of clusters

K- Means Algorithm

Score Model

Train Model

Clean the missing Data like drop off and pickup coordinates or fare information

Select the require attributes in Dataset

|  |
| --- |
| Execute the main script will perform the below action  ./ dbs\_challenge\_top10\_batch.sh   1. DB and table creation    1. sample data staging table    2. Error record validation by standard validation like missing trip start time ,trip end time ,longitude and latitude attributes    3. Extracting error record into exception error table record for cleaning data using route lookup table    4. Extracting good record to get top 10 routes good\_sorted\_data    5. Final data loaded into top\_10\_routes\_final\_output    6. Moved the good process data into backup table clean\_sorted\_data |

Script & Jar



Clustering Technics

|  |  |
| --- | --- |
|  | Reading csv file  DataFmgood=good\_sorted\_data  #Saving data into table  Import seaborn as sb  Sb.factorplot(data= DataFmgood,x=1’cluster’,kind=’count’  Size=10,aspect=2) |