Introduction

You were introduced to Spark with the claim that it’s 100x faster than MapReduce because it does in-memory processing. In this assignment, you will be expected do a Proof of Concept(POC) on this.

As per Techopedia, a proof of concept (POC) is a demonstration, the purpose of which is to verify that specific concepts or theories have the potential for real-world application. POC is, therefore, a prototype that is designed to determine feasibility but does not represent deliverables. In IT industry POCs are done for various purposes, one of them could be to compare the performance of two different tools designed to perform the same task. The overall objective of POC is to find solutions to technical problems. POCs are purely experimental. Hence, the results achieved during a POC may not be the final one. You might get different results when the solution is implemented on a separate dataset or an entirely different scenario. So, it is encouraged to test and experiment with recurring use cases or scenes and take a decision after analysing all the results.

So for this POC, you will be doing your analysis using the below-mentioned use cases:

- lookup of a single row from the entire data set
- Filtering multiple rows
- Group by and then order by

The above three use cases will be implemented using Pig and Spark RDD. Which means you are expected to develop 3*2 i.e. 6 programs, execute each of them in your Amazon EC2 instance/Cloudera Quickstart VM and generate the output in an HDFS location. Let's say for the use case "lookup of a single row from the entire data set", you will develop and execute a Pig

script and Spark code using RDDs. You have to ensure that, the output generated by both the methods are consistent.

Dataset Description

The dataset which will be used for this assignment is the TLC trip record data for Yellow taxis. The fields present in the data and their meaning is given below for your reference:

Field Name	Description
VendorID	A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.
tpep_pickup_datetime	The date and time when the meter was engaged
tpep_dropoff_datetime	The date and time when the meter was disengaged
passenger_count	The number of passengers in the vehicle. This is a driver-entered value
trip_distance	The elapsed trip distance in miles reported by the taximeter
RatecodeID	The final rate code in effect at the end of the trip. 1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride

store_and_fwd_flag	This flag indicates whether the trip record was held in-vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. Y= store and forward trip N= not a store and forward trip
PULocationID	The ID of the location from where the passenger was picked.
DOLocationID	The ID of the location where the passenger was dropped.
payment_type	A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
fare_amount	The time-and-distance fare calculated by the meter.
extra	Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges
mta_tax	$0.50 MTA tax that is automatically triggered based on the metered rate in use.
tip_amount	This field is automatically populated for credit card tips. Cash tips are not included.
tolls_amount	The total amount of all tolls paid on the trip.

improvement_surcharge	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
total_amount	The total amount charged to passengers. Does not include cash tips.

Please note that the input data is for the months July 2017 to Dec 2017 and total size is ~7GB.