Pig Script 1 :

```
############################   The following is the Pig script for
Problem 1: Record lookup
###################################################################
############################   This script is to be stored as Pig1.pig
#########################################################################
###############
############################   The script picks up input data from
directory - spark assignment created under HDFS on which root user has
Owner access #######
############################   The script is run from Pig grunt , part of
Cloudera setup on AWS VM - CDH install
##############################################
################## Since,the script is being run from a directory -
Pig(created under root) , we could avoid giving absolute input path i.e.
user/root/.... ###
############################   The output is being written to a directory
- pigoutput in root directory
##############################################
############################   Document followed - How to run Pig on AWS
EC2 from Module 6 Session1
##############################################




data = LOAD 'spark_assignment/input_dataset/yellow_tripdata_*' USING
PigStorage(',') AS (f1:int, f2:chararray, f3:chararray, f4:int,
f5:double, f6:int, f7:bytearray,
f8:int, f9:int, f10:int, f11:double, f12:double, f13:double, f14:double,
f15:double, f16:double, f17:double);

filtered = FILTER data BY f1 == 2 AND f2 == '2017-10-01 00:15:30' AND f3
== '2017-10-01 00:25:11' AND f4 == 1 AND f5 == 2.17 ;

STORE filtered INTO 'pigoutput/PigOutput1.out';

##############################################         End of Pig1.pig
###############################################################
```

Pig Script 2 :

```
############################   The following is the Pig script for
Problem 2: Filter Records
###############################################################
############################   This script is to be stored as Pig2.pig
#########################################################################
###############
############################   The script picks up input data from
directory - spark assignment created under HDFS on which root user has
Owner access #######
############################   The script is run from Pig grunt , part of
Cloudera setup on AWS VM - CDH install
##############################################
################## Since,the script is being run from a directory -
Pig(created under root) , we could avoid giving absolute input path i.e.
user/root/.... ###
```

```
############################   The output is being written to a directory
- pigoutput in root directory
##############################################
############################   Document followed - How to run Pig on AWS
EC2 from Module 6 Session1
##############################################



data = LOAD 'spark_assignment/input_dataset/yellow_tripdata_*' USING
PigStorage(',') AS (f1:int, f2:chararray, f3:chararray, f4:int,
f5:double, f6:int, f7:bytearray,
f8:int, f9:int, f10:int, f11:double, f12:double, f13:double, f14:double,
f15:double, f16:double, f17:double);

filtered = FILTER data BY f6 == 4;

STORE filtered INTO 'pigoutput/PigOutput2.out';

##########################################             End of Pig2.pig
###############################################################


Pig Script 3 :

############################   The following is the Pig script for
Problem 3: Group By , Count and Sort
####################################################
############################   This script is to be stored as Pig3.pig
##########################################################################
###############
############################   The script picks up input data from
directory - spark assignment created under HDFS on which root user has
Owner access #######
############################   The script is run from Pig grunt , part of
Cloudera setup on AWS VM - CDH install
##############################################
################## Since,the script is being run from a directory -
Pig(created under root) , we could avoid giving absolute input path i.e.
user/root/.... ###
############################   The output is being written to a directory
- pigoutput in root directory
##############################################
############################   Document followed - How to run Pig on AWS
EC2 from Module 6 Session1
##############################################

data = LOAD 'spark_assignment/input_dataset/yellow_tripdata_*' USING
PigStorage(',') AS (VendorID,
tpep_pickup_datetime,
tpep_dropoff_datetime,
passenger_count,
trip_distance,
RatecodeID,
store_and_fwd_flag,
PULocationID,
DOLocationID,
payment_type,
fare_amount,
extra,
mta_tax,
tip_amount,
tolls_amount,
improvement_surcharge,
total_amount
);

data_not_null = FILTER data BY payment_type != '';
grouped_data = GROUP data_not_null by payment_type;
group_count = FOREACH grouped_data GENERATE group AS PAYMENT_TYPE,
COUNT_STAR(data_not_null) AS cnt;
sorted_count = ORDER group_count BY cnt;
result_without_header = FILTER sorted_count BY $0 != 'payment_type';
STORE result_without_header INTO 'pigoutput/PigOutput3.out';


##Line no 76 filters out all the non-null records
##Line no 77 groups by payment_type on not null dataset
##Line no 78 applies COUNT_STAR operation on non-null dataset so that
count of records corresponding to each payment_type could be found
##Line no 79 applies ORDER By on count obtained, in ascending order
##Line no 80 gets the records by removing the header row

##########################################             End of Pig3.pig
###############################################
```

Spark Java Code for problem 1

```
############################   The following is the Java code for Problem
1: Record lookup
###################################################################
############################   This code is to be stored as
AssignmentProblem1.java
#########################################################################
##
############################   The program picks up input data from
directory - spark assignment created under HDFS on which root user has
Owner access #######
############################   The script is run from Shell corresponding
to AWS VM - CDH install
##############################################
############################   The script is being run from root, the
executable jar is placed in /home/ec2-user directory
##################################
############################   Hence, for input path , the absolute path
for input dataset is specified
##################################
############################   The output is being written to a directory
created under HDFS on which root user has Owner access
###########################
############################   References - Guidelines for Running Spark
Codes in EC2 Unix Box in project2
#########################################


package spark.assignment1;

//Importing all the required Spark packages

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;

public class AssignmentProblem1 {

           public static void main(String[] args) {

                 //Instantiating the spark conf instance

                 SparkConf conf = new
SparkConf().setAppName("MyFirstSparkProgram").setMaster("local[*]");
                 System.out.println("Conf has been set");

                 //Declaring a JavaRDD and initializing it to null

                 JavaRDD<String> stringRDD = null;

                 //Need a try catch block here so that any problem in
creating Spark Context
                 // Or in reading the input file be handled gracefully

                 try {

                       JavaSparkContext sc = new JavaSparkContext(conf);
                       System.out.println("Context has been
initialized");

                       //Reading the input data set as a text file in
JavaRDD created
                       stringRDD =
sc.textFile("/user/root/spark_assignment/input_dataset/yellow_tripdata*")
;
                       System.out.println("File has been read
successfully");
                 }catch(Exception e) {

                       //returning the stack trace of Exception
encountered
                       e.printStackTrace();
                 }

                 //Implementing filter transformation on JavaRDD which
has input data set

                 JavaRDD<String> lookupRDD = stringRDD.filter(new
Function<String,Boolean>(){
                       public Boolean call(String row) throws Exception {

                             //Reading each record and splitting it by:
","
                             //Adding the split record to an array

                             String[] rowValues = row.split(",");

                             //Checking if the record under processing is
not null
                             //and has length at least = 17
                             if(rowValues!=null && rowValues.length>=17) {
```

```
                                        //Below is the filter condition
as per Problem no 1
                                   if("2".equals(rowValues[0]) && "2017-
10-01 00:15:30".equals(rowValues[1]) && "2017-10-01
00:25:11".equals(rowValues[2]) && "1".equals(rowValues[3]) &&
"2.17".equals(rowValues[4])) {
                                        return true;
                                   } //Only records are returned in
lookupRDD which satisfy the filter condition
                              }
                              return false;
                         }
                    }
                    );  //lambda function ends here

                    System.out.println("Writing the output now");

                    //Writing the output to a text file which will be saved
at location as given in command line - args[1]
                    lookupRDD.saveAsTextFile(args[1]);

                    System.out.println("File has been written with output");
          }



}


##########################              End of AssignmentProblem1.java
####################################################
Spark Java Code for problem 2


############################   The following is the Java code for Problem
2: Filter Records
###################################################################
############################   This code is to be stored as
AssignmentProblem2.java
#########################################################################
##
############################   The program picks up input data from
directory - spark assignment created under HDFS on which root user has
Owner access #######
############################   The script is run from Shell corresponding
to AWS VM - CDH install
#############################################
############################   The script is being run from root, the
executable jar is placed in /home/ec2-user directory
##################################
############################   Hence, for input path , the absolute path
for input dataset is specified
##################################
############################   The output is being written to a directory
created under HDFS on which root user has Owner access
###########################
############################   References - Guidelines for Running Spark
Codes in EC2 Unix Box in project2
#########################################
```

```
package spark.assignment2;

//Importing all the required Spark packages

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;

public class AssignmentProblem2 {

      public static void main(String[] args) {

            //Instantiating the spark conf instance

            SparkConf conf = new
SparkConf().setAppName("MySecondSparkProgram").setMaster("local[*]");
            System.out.println("Conf has been set");

            //Declaring a JavaRDD and initializing it to null

            JavaRDD<String> stringRDD = null;

            //Need a try catch block here so that any problem in creating
Spark Context
            // Or in reading the input file be handled gracefully

            try {

                  JavaSparkContext sc = new JavaSparkContext(conf);
                  System.out.println("Context has been initialized");

                  //Reading the input data set as a text file in JavaRDD
created

                  stringRDD =
sc.textFile("/user/root/spark_assignment/input_dataset/yellow_tripdata*")
;
                  System.out.println("File has been read successfully");
            }catch(Exception e) {

                  //returning the stack trace of Exception encountered
                  e.printStackTrace();
            }

            //Implementing filter transformation on JavaRDD which has
input data set

            JavaRDD<String> filterRDD = stringRDD.filter(new
Function<String,Boolean>(){

                  @Override
                  public Boolean call(String row) throws Exception {

                        //Reading each record and splitting it by: ","
                        //Adding the split record to an array
                        String[] rowValues = row.split(",");

                        //Checking if the record under processing is not
null
```

```
                        //and has length at least = 17

                        if(rowValues!=null && rowValues.length>=17) {

                              //Below is the filter condition as per
Problem no 2
                              if("4".equals(rowValues[5])) {
                                    return true;
                              }
                        }
                        return false;
                  }
            }
                        );//lambda function ends here
            System.out.println("Writing the output now");

            //Writing the output to a text file which will be saved at
location as given in command line - args[1]
            filterRDD.saveAsTextFile(args[1]);

            System.out.println("File has been written with output");

      }

}
```

```
##########################              End of AssignmentProblem2.java
####################################################
```

Spark Java Code for problem 3

```
##############################   The following is the Java code for Problem
3: Group By , Count and Sort
#####################################################
##############################   This code is to be stored as
AssignmentProblem2.java
##########################################################################
##
##############################   The program picks up input data from
directory - spark assignment created under HDFS on which root user has
Owner access #######
##############################   The script is run from Shell corresponding
to AWS VM - CDH install
##############################################
##############################   The script is being run from root, the
executable jar is placed in /home/ec2-user directory
##################################
##############################   Hence, for input path , the absolute path
for input dataset is specified
##################################
##############################   The output is being written to a directory
created under HDFS on which root user has Owner access
###########################
##############################   References - Guidelines for Running Spark
Codes in EC2 Unix Box in project2
#########################################
```

```
package spark.assignment3;

//Importing all the required Spark packages

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;

import scala.Tuple2;

public class AssignmentProblem3 {

      public static void main(String[] args) {

            //Instantiating the spark conf instance

            SparkConf conf = new
SparkConf().setAppName("MyThirdSparkProgram").setMaster("local[*]");
            System.out.println("Spark Conf has been set successfully");

            //Declaring a JavaRDD and initializing it to null

            JavaRDD<String> stringRDD = null;

            //Need a try catch block here so that any problem in creating
Spark Context
            // Or in reading the input file be handled gracefully

            try {

                  JavaSparkContext sc = new JavaSparkContext(conf);
                  System.out.println("Context has been initialized");

                  //Reading the input data set as a text file in JavaRDD
created

                  stringRDD =
sc.textFile("/user/root/spark_assignment/input_dataset/yellow_tripdata*")
;
                  System.out.println("File has been read successfully");

            }catch(Exception e) {

                  //returning the stack trace of Exception encountered
                  e.printStackTrace();
            }

            //Implementing filter transformation on JavaRDD which has
input data set

            JavaRDD<String> resultantRDD = stringRDD.filter(new
Function<String,Boolean>(){
                  @Override
                  public Boolean call(String row) throws Exception {

                        //Reading each record and splitting it by: ","
                        //Adding the split record to an array
                        String[] rowValues = row.split(",");

                        //Checking if the record under processing is not
null
                        //and has length at least = 17

                        if(rowValues!=null && rowValues.length>=17) {

                              //Filtering out the non-null rows and then
filtering out the header row
                              if( (!rowValues[9].isEmpty())) {

      if(!("payment_type".equalsIgnoreCase(rowValues[9])))
                                    return true;
                              }
                        }
                        return false;
                  }
            }
            ); //lambda function ends here

            //Using map to Pair transformation to get key value pairs
            JavaPairRDD<String , Integer> pairRDD =
resultantRDD.mapToPair(

                        //This is going to split the record and get value
of payment type column : key-> Payment_type , value -> 1
                        x -> new Tuple2<String , Integer>(x.split(",")[9]
, 1)

                        );

            //The reduce by key transformation is going to give the sum of
all records of a payment type
            JavaPairRDD<String , Integer> groupRDD = pairRDD.reduceByKey(
                        (x,y) -> x+y
                        );

            //The result will be sorted by key by default
            //We first swap the key and value so that result can be sorted
by value
            //We again swap back the result so that we have payment type
as key and sum of records of that payment type as value

            groupRDD = groupRDD.mapToPair(x ->
x.swap()).sortByKey().mapToPair(x -> x.swap());

            System.out.println("Writing the output now");

            //Writing the output to a text file which will be saved at
location as given in command line - args[1]
            groupRDD.saveAsTextFile(args[1]);
            System.out.println("File has been written with output");

            //groupRDD.foreach(x -> System.out.println(x._1 + ":" +
x._2));

      }

}
```

```
##########################              End of AssignmentProblem3.java
####################################################
```