MODULE 10: Assignment

1) **Write a map only algorithm which will read the original dataset as input and filter out all the records which have event_epoch_time, user_id, device_id, user_agent as NULL.**

/* As this is filtering process and data aggregation is not needed so the best suitable process is MAP only algorithm as it does filtering and data transformation activities */

/* In this pseudo code we are filtering the data fields (event_epoch_time, user_id, device_id and user_agent) which are not null */

Algorithm

```
MAP(Key, Value)                 /* here the algorithm which takes Key value pair as input,
                                   We can ignore the Key as it is record identifier as is of no
                                   use and the value is a tab separated record */
LS = split(Value, '\t')         /* split function takes the entire record and a variable
                                   Delimiter as input and returns an array of strings with all
                                   The record values in it preserving the same original order
                                   as in the data set and stores in LS */
event_epoch_time = LS[1]        /* Fetch the first element from array LS */

user_id = LS[2]                 /* Fetch the second element from array LS */
device_id = LS[3]               /* Fetch the third element from array LS */
user_agent = LS[4]              /* Fetch the forth element from array LS */
if (event_epoch_time != NULL && user_id !=NULL &&   /* Here we check the condition
                                                       whether the values in the fields are
        device_id != NULL  &&  user_agent != NULL)  Not Null if its NULL we filter out
                                                       those fields */
        write(key, Value)   /* we write these fields as Key value pair */
```

2) **An algorithm to read the user agent and extract OS Version and Platform from it.**

/* As this is data transformation activity the best suitable process is MAP only algorithm as data aggregation is not needed here */

/* In this pseudo code we are splitting the filed (user_agent) into two parts separated by colon and we are taking those two fields in two separate variables and we are displaying it */

Algorithm

```
MAP(Key, Value)                 /* here the algorithm which takes Key value pair as input,
                                   We can ignore the Key as it is record identifier as is of no
                                   use and the value is a tab separated record */

LS = Split(Value, '\t')         /* split function takes the entire record and a variable
                                   Delimiter as input and returns an array of strings with all
                                   The record values in it preserving the same original order
                                   as in the data set and stores in LS */
A = Split(LS[4], ':')           /* here we split the 4th element of the array by colon and
                                   take it into a separate array field 'A' */
Platform = A[1]                 /* we take the first element of the array field A[1] */
OS_version = A[2]               /* we take the second element of the array field A[2] */
Write(user_agent, [platform, OS_version] ) /* we write the fields as key value pairs */
```

3) /* As there is global count variable there is no need of Reduce algorithm so we can use MAP only algorithm here */

a) **Find out the no of veg and non veg pizzas sold**

/* As this is filtering activity the best suitable process is MAP only algorithm as data aggregation is not needed here */

Algorithm

```
MAP(Key, value)                 /* here the algorithm which takes Key value pair as input,
                                   We can ignore the Key as it is record identifier as is of no
                                   use and the value is a tab separated record */

LS = Split(value, '\t')         /* split function takes the entire record and a variable
                                   Delimiter as input and returns an array of strings with all
```

The record values in it preserving the same original order as in the data set and stores in LS */

isVeg = LS[12] /* Fetch the 12th element from array LS */

if (isVeg == Y) /* check the condition whether its veg pizza or not */

getCounter("VEG"). IncrementBy(1) /* If yes then we display it VEG and increment By 1

elseif (isVeg == N) /* check the condition whether its non veg pizza or not */

getCounter("NON-VEG").incrementBy(1) /* if yes then we display Nonveg and increment the counter by 1

End if

b) Find out the size wise distribution of pizzas sold

Algorithm

MAP(Key, value) /* here the algorithm which takes Key value pair as input,

We can ignore the Key as it is record identifier as is of no use and the value is a tab separated record */

LS = split(value, '\t') /* split function takes the entire record and a variable

Delimiter as input and returns an array of strings with all

The record values in it preserving the same original order as in the data set and stores in LS */

Size = LS[7] /* Fetch the 7th element from array LS */

If(size == R) /* check the condition whether the condition is euqla to 'R' or not */

getCounter("Regular").Incrementby(1) /* if yes the display Regular and increment by 1 for every true condition

elseif (size == M) /* check the condition whether the condition is euqla to 'M' or not */

getCounter("Medium").IncrementBy(1) /* if yes the display Mediumand ncrement by 1 for every true condition

elseif(size == L) /* Check the condition whether it is equal to 'L' or not

getCounter("LARGE").IncrementBy(1) /* if yes the display Large and increment by 1 for every true condition

End if

c) Find out how many cheese burst pizzas were sold

Algorithm

MAP(Key, value) /* here the algorithm which takes Key value pair as input,

We can ignore the Key as it is record identifier as is of no use and the value is a tab separated record */

LS = split(value, '\t') /* split function takes the entire record and a variable

Delimiter as input and returns an array of strings with all

The record values in it preserving the same original order as in the data set and stores in LS */

isCheeseBurst = LS[6] /* Fetch the 6^{th} element from array LS */

if (isCheeseBurst == Y) /* check the condition whether isCheeBurst is equal to 'y' or not

getCounter("CheeseBurst").IncrementBy(1) /* if yes the display CheeseBurst and increment by 1 for every true condition

End if

d) find out how many small cheese burst pizzas were sold. Ideally the count should be 0 because the cheese burst pizzas is available for medium and large.

Algorithm

MAP(Key, value) /* here the algorithm which takes Key value pair as input,

We can ignore the Key as it is record identifier as is of no use and the value is a tab separated record */

LS = split(value, '\t') /* split function takes the entire record and a variable

Delimiter as input and returns an array of strings with all

The record values in it preserving the same original order as in the data set and stores in LS */

isCheeseBurst = LS[6] /* Fetch the 6th element from array LS */

size = LS[7] /* Fetch the 7th element from array LS */

if (size == S && isCheeseBurst == Y) /* check the condition whether Size equal to 'S and isCheeseBurst is equal to y */

getCounter("Small CheeseBurst").IncrementBy(1) /* If yes then display Small cheeseBurst and increment by 1 for every true record */

End if.

e) Find out the number of CheeseBurst pizzas whose cost is below 500

Algorithm

MAP(Key, value) /* here the algorithm which takes Key value pair as input,

We can ignore the Key as it is record identifier as is of no use and the value is a tab separated record */

LS = split(value, '\t') /* split function takes the entire record and a variable

Delimiter as input and returns an array of strings with all

The record values in it preserving the same original order as in the data set and stores in LS */

isCheeseBurst = LS[6] /* Fetch the 6th element from array LS */

Price = LS[9] /* Fetch the 9th element from array LS */

If (Price < 500 && isCheeseBurst == Y) /* check the condition and display */

getCounter("CheeseBurst").IncrementBy(1)

End If

4) **Assume that the predefined method getCounter does not exists write the updated algorithms for the tasks in point 3**

   a. **Find out the no of veg and non veg pizzas sold**

/* As this code doesn't have any global variable for count as it consists of data aggregation the best suited algorithm is Map reduce algorithm, the output of the map only algorithm is given as input to the Reduce algorithm and data aggregation part is done in this */

Algorithm

```
MAP(Key, value)          /* here the algorithm which takes Key value pair as input,
                            We can ignore the Key as it is record identifier as is of no
                            use and the value is a tab separated record */

LS = split(value, '\t')  /* split function takes the entire record and a variable
                            Delimiter as input and returns an array of strings with all
                            The record values in it preserving the same original order
                            as in the data set and stores in LS */

isVeg = LS[12]           /* Fetch the 12th element from array LS */

write (isVeg, 1)
```

Reduce algorithm

```
REDUCE(Key, ValueList)   /* the values from map only algorithm are sent to reduce
                            algorithm for data aggregation /*
count = 0                /* Initialize the cunt to zero */
For I = 1 to valueList.length   /* Loop for every key value pairs in the map only construct */
        count = count + 1
Write (key, count)       /* Display the final count */
```

   b. **Find out the size wise distribution of pizas sold**

Algorithm

```
MAP(Key, value )         /* here the algorithm which takes Key value pair as input,
                            We can ignore the Key as it is record identifier as is of no
                            use and the value is a tab separated record */

LS = split(value, '\t')  /* split function takes the entire record and a variable
```

Delimiter as input and returns an array of strings with all

The record values in it preserving the same original order as in the data set and stores in LS */

Size = LS[7] /* Fetch the 7th element from array LS */

Write(size, 1)

Redue algorithm

REDUCE(Key, ValueList) /* the values from map only algorithm are sent to reduce algorithm for data aggregation /*

count = 0 /* Initialize the cunt to zero */

For I = 1 to valueList.length /* Loop for every key value pairs in the map only construct */

count = count + 1

Write (key, count) /* Display the final count */

c. Find out how many cheese burst pizzas sold

Algorithm

MAP(Key, value) /* here the algorithm which takes Key value pair as input,

We can ignore the Key as it is record identifier as is of no use and the value is a tab separated record */

LS = split(value, '\t') /* split function takes the entire record and a variable

Delimiter as input and returns an array of strings with all

The record values in it preserving the same original order as in the data set and stores in LS */

isCheeseBurst = LS[6] /* Fetch the 6th element from array LS */

if (isCheeseBurst == Y) /* Check the condition */

write(isCheeseBurst, 1)

Reduce algorithm

```
REDUCE(Key, ValueList)          /* the values from map only algorithm are sent to reduce
                                   algorithm for data aggregation /*
count = 0                       /* Initialize the cunt to zero */
For I = 1 to valueList.length   /* Loop for every key value pairs in the map only construct */
        count = count + 1
Write (key, count)              /* Display the final count */
```

d. Find out how many small cheese burst pizzas were sold. Ideally, the count should be 0 because cheese burst is available for medium and large

Algorithm

```
MAP(Key, value)                 /* here the algorithm which takes Key value  pair as input,
                                   We can ignore the Key as it is record identifier as is of no
                                   use and the value is a tab separated record */

LS = split(value  '\t')         /*  split function takes the entire record and a variable
                                   Delimiter as input and returns an array of strings with all
                                   The record values in it preserving the same original order
                                   as in the data set and stores in LS  */

Size = LS[7]                    /* Fetch the 7th  element from array LS */
isCheeseBurst = LS[6]           /* Fetch the 6th  element from array LS */
If (size == S && isCheeseBurst == y)  /* check the condition */
        Write(size, 1)
```

Reduce algorithm

```
REDUCE(Key, ValueList)          /* the values from map only algorithm are sent to reduce
                                   algorithm for data aggregation /*
count = 0                       /* Initialize the cunt to zero */
For I = 1 to valueList.length   /* Loop for every key value pairs in the map only construct */
        count = count + 1
Write (key, count)              /* Display the final count */
```

e. Find out number of cheese Burst Pizzas whose cost is below 500

Algorithm

```
MAP(Key, value)                 /* here the algorithm which takes Key value     pair as input,
                                   We can ignore the Key as it is record identifier as is of no
                                   use and the value is a tab separated record */

LS = split(value,  '\t')        /*  split function takes the entire record and a variable
                                   Delimiter as input and returns an array of strings with all
                                   The record values in it preserving the same original order
                                   as in the data set and stores in LS    */

isCheeseBurst = LS[6]           /* Fetch the 6th   element from array LS */
Price = LS[9]                   /* Fetch the 9th   element from array LS */
If (isCheeseBurst == Y && Price < 500)  /* check the condition */
Write (isCheeseBurst, 1)
```

Reduce algorithm

```
REDUCE(Key, ValueList)          /* the values from map only algorithm are sent to reduce
                                   algorithm for data aggregation /*
count = 0                       /* Initialize the cunt to zero */
For I = 1 to valueList.length   /* Loop for every key value pairs in the map only construct */
        count = count + 1
Write (key, count)              /* Display the final count */
```