MapReduce Assignment

Question 1

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2.Check if any of event_epoch_time or user_id or device_id or user_agent is NULL
3. If any of the field is NULL return/filter out the particular record

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records. Check was made if any of the any of event_epoch_time or user_id or device_id or user_agent is NULL. Then if NULL then the values of corresponding records are given to the fields.For the Output the key is taken as a default value whereas the Values here represent the list of value of fields in each record.

Pseudocode:

```
Map(key,value)
LS = split(value, ‘\t’)
event_epoch_time = LS[1]
user_id = LS[2]
device_id = LS[3]
user_agent = LS[4]
if (event_epoch_time == NULL || user_id == NULL || device_id == NULL ||
user_agent == NULL) {
```

```
event_epoch_time = LS[1]
user_id  = LS[2]
device_id = LS[3]
user_agent = LS[4]
pizza_name = LS[5]
isCheeseBurst = LS[6]
Size = LS[7]
AddedToppings = LS[8]
Price = LS[9]
CouponCode = LS[10]
Order_Event = LS[11]
isVeg = LS[12]

for (i=0 to value.length)
 {
   write (Record, LS[i])
   i++;
 }
}
end Map
```

Output:

Sample Example:

(Record , { 1515507854000, abc@gmail.com, DeviceId001, NULL , Veg Extravaganza,Y,M,NULL,700,NULL,Delivered,Y})

********** End Question 1 **********

Question 2:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2.Read the User agent by giving corresponding position in array to the field
3. Use the split function in the User_Agent to split Platform and OS version and filter the same as output

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to extract OS Version and Platform from the user_agent we denote 4^{th} position of the array to the User_agent. Then we further split the Value present in the user_agent with the Split function and correspondingly filter them out as Platform and OS version separately.

Pseudocode:

```
Map(key,value)
LS = split(value , '\t')
user_agent = LS[4]
if (user_agent != NULL)
  {
   UA = split(LS[4], ':')
   platform = UA[1]
   os_version = UA[2]
   write (user_agent , platform)
   write (user_agent , os_version)
end Map
```

OutputSample: (user_agent , iPhone7Plus) , (user_agent ,IOS 10.3.3)

********** End Question 2**********

Question 3.1:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isVeg is “Y” , if “Y” then pass the key as the String Name to the getCounter.incrementBy method to get the unique global counter variable and increments it with the check.
3. Filter out the same using the write function for getting the count of Veg Pizza’s
4.Else it will give the count of Non_Veg pizza’s
5.Filter out the above using the write function for getting the count of Non_Veg Pizza’s

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check if the Pizza is Veg or not so we check with the 12^{th} field in array and map it to isVeg field. Check for isVeg is “Y” gives the Veg pizza’s and we use the getCounter.IncrementBy method to get the global counter variable with a count as output . We use that count to write the number of Veg Pizza’s and through the Else part we calculate the count of Nonveg pizza when the isVeg != “Y” to get the count and write the number of Non_Veg pizza’s.

Pseudocode:

```
Map(key,value)
LS = split(value , '\t')
isVeg = LS[12]
String vegPizza
String nonvegPizza
if (isVeg == "Y")
  {
  isVeg = Key
  vegPizza = isVeg
  veg = getCounter.("vegPizza").incrementBy(1)
 write (vegPizza,veg) // output will be (vegPizza , {1,1,1})
 }
else
  {
  isVeg = Key
  nonvegPizza = isVeg
  non_veg = getCounter.("nonvegPizza").incrementBy(1)
  write (nonvegPizza,non_veg) // output will be (nonvegPizza , 1)
 }
end Map
```

**********End Question 3.1**********

Question 3.2:

Algorithm:

1. Read Dataset as Input through Map Function with "Key" as Record identifier and the tab separated records as "Values"
2. Check if the field size is "M" , if "M" then pass the key as the String Name to the getCounter.incrementBy method to get the unique global counter variable and increments it with the check.
3. Filter out the same using the write function for getting the count of medium

pizza's.
4.Elseif the size = 'L' then the counter will give the count of Large size pizza's
5.Filter out the above using the write function for getting the count of Large Pizza's
6. Else the count of small pizza's can be obtained through the count method and those can be filtered out using the write function for getting the count of small pizza's

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the "Split function" with values as "Key" and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check the Size in the 7^{th} field in array and map it to size field. Check if size is "M" gives the pizza's in Medium and we use the getCounter.IncrementBy method to get the global counter variable with a count as output . We use that count to write the count of Medium pizza's and through the Elseif part we calculate the count of size of Large Pizza's when the size = "L" to get the count and write the number of large pizza's. Else we calculate the count of size of Regular Pizza's when the size != "M" or "L" to get the count and write the number of small pizza's

Pseudocode:

```
Map(key,value)
LS = split (value, '\t')
size = LS[7]
String medium
String large
String regular
if (size == "M") {
```

```
  size = key
  medium = size
  medium_pizza = getCounter.(“medium”).incrementBy(1)
  write (medium , medium_pizza) // output will be (medium , 2)
 }
elseif (size == “L”) {
  size = key
  large = size
  large_pizza = getCounter.(“large”).incrementBy(1)
  write (large , large_pizza) // output will be (large , 1)
}
else {
  size = key
  regular = size
  small_pizza = getCounter.(“regular”).incrementBy(1)
  write (regular , small_pizza) // output will be (regular ,1)
 }
end Map
```

********** End of Question 3.2**********

Question 3.3:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isCheeseBurst is “Y” , if “Y” then pass the key as the String Name to the getCounter.incrementBy method to get the unique global counter variable and increments it with the check.
3. Filter out the same using the write function for getting the count of CheeseBurst Pizza’s

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the "Split function" with values as "Key" and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check the pizza if it is CheesBurst or not so in the 6^{th} field in array and map it to isCheeseBurse field. Further we check if the value in isCheeseBurst is "Y" if yes we use the getCounter.IncrementBy method to get the global counter variable with a count as output and we use that count to write the number of CheeseBurst pizza's.

Pseudocode:

```
Map(Key,value)
LS = split (value , '\t')
isCheeseBurst = LS[6]
String cheeseBurst
if (isCheeseBurst == "Y")
  {
  isCheeseBurst = Key
  cheeseBurst = isCheeseBurst
  count_Cheese_Burst = getCounter.("cheeseBurst").incermentBy(1)
 write (cheeseBurst , count_Cheese_Burst) // output will be (cheeseBurst , 3)
  }
end Map
```

**********End of question 3.3**********

Question3.4:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isCheeseBurst is “Y” and if the size is “R”(small), then pass the key as the String Name to the getCounter.incrementBy method to get the unique global counter variable and increments it with the check.
3. Filter out the same using the write function for getting the count of small CheeseBurst Pizza’s

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check the pizza of small size it is CheeseBurst or not so in the 6^{th} field in array and map it to isCheeseBurst field and the Size to the 7^{th} field. Further we check if the value in isCheeseBurst is “Y” and the size to be equal to “R” if yes we use the getCounter.IncrementBy method to get the global counter variable with a count as output and we use that count to write the number of small CheeseBurst pizza’s.

Pseudocode:

```
Map (key , value)
LS = split (value , ‘\t’)
isCheeseBurst = LS[6]
Size = LS[7]
String SmallCb
if (Size == “R” && isCheeseBurst == “Y”)
{
 SmallCb = Key
 count_scb = getCounter.(“SmallCb”).incrementBy(1)
 write (SmallCb , count_scb) // output will be (SmallCb , 0)
```

}
end Map
**********End of question 3.4**********

Question 3.5:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isCheeseBurst is “Y” and if the prize is less than 500, then pass the key as the String Name to the getCounter.incrementBy method to get the unique global counter variable and increments it with the check.
3. Filter out the same using the write function for getting the count of CheeseBurst Pizza’s with prize less than 500.

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check the pizza of small size it is CheeseBurst or not so in the 6^{th} field in array and map it to isCheeseBurst field and the Price to the 9^{th} field. Further we check if the value in isCheeseBurst is “Y” and the Price is less than 500 if yes we use the getCounter.IncrementBy method to get the global counter variable with a count as output and we use that count to write the number of small CheeseBurst pizza’s.

Pseudocode:

```
Map (Key , Value)
LS = split (value , '\t')
isCheeseBurst = LS[6]
Price = LS[9]
String CheeseBurst
if (isCheeseBurst == "Y" &&  Price < 500)
{
 CheeseBurst = Key
count_ccb = getCounter("CheeseBurst").incrementBy(1)
write (CheeseBurst , count_ccb) // output will be (CheeseBurst , 1)
}
end Map
```
********** End of Question 3.5**********

Question 4.1

Algorithm:

1. Read Dataset as Input through Map Function with "Key" as Record identifier and the tab separated records as "Values"
2. Check if the field isVeg is "Y" , if "Y" then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of Veg Pizza's
3. Else the isVeg will have the Non_veg pizza's so the output of the Map will be Non_veg pizza's and the list of values which is given as input to the Reduce
4.Reducer increments the count filters out the key and the total count of the Veg/Non veg pizza's respectively

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the "Split function" with values

as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check if the Pizza is Veg or not so we check with the 12^{th} field in array and map it to isVeg field. Check for isVeg is “Y” gives the Veg pizza’s and we use the output of Veg pizza as the input to the REDUCE function else gives the NonVeg pizzas where we use the output of NonVeg pizza as the input to REDUCE function , here the REDUCE function filters out the Key and the number of count for each key as a summation of value.

Pseudocode:

```
Map(Key , value)
LS = split (value , ‘\t’)
isVeg = LS[12]
String VegPizza
String NonVegPizza
if (isVeg == “T”)
  {
  isVeg = Key
  VegPizza = isVeg
  write (VegPizza , 1) // output will be (VegPizza , {1,1,1})
  }
else
   {
  isVeg = Key
  NonVegPizza = isVeg
  write (NonVegPizza ,1) // output will be (NonVegPizza ,1)
}
end Map

REDUCE (key , valuelist)
 pcount = 0
 for (i=0 to valuelist.length)
```

pcount = pcount + 1;
write (key , pcount)
end REDUCE

**********End of Question 4.1**********

Question4.2:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field Size is “M” , if “M” then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of Medium Pizza’s
3. Check if the field Size is “L” , if “L” then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of Large Pizza’s
4. Else then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of Small Pizza’s
5. Reducer increments the count filters out the key and the total count of the different sizes of pizza respectively

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check for different Pizza Size so we check with the 7^{th} field in array and map it to Size field. Further for each field we get the output from the mapper to give it as input to the reducer and filter out number of counts of pizza’s on each size.

Pseudocode:

```
Map (Key, value)
LS = Split (value , '\t')
Size = LS[7]
String Medium
String Large
String Regular
if (Size == "M")
  {
  Size = Key
  Medium = Size
  write (Medium , 1)
  }
elseif (Size == "L")
  {
  Size = Key
  Large = Size
  write (Large,1)
  }
else
  {
   Size = Key
   Regular = Size
   write (Regular , 1)
   }
end Map

REDUCE (key , valuelist)
 scount = 0
 for (i=0 to valuelist.length)
scount = scount + 1;
write (key , scount )
```

end REDUCE

**********End of Q4.2**********

Question 4.3:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isCheeseBurst is “Y” , if “Y” then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of CheeseBurst Pizza’s
3. Reducer increments the count filters out the key and the total count of the Cheeseburst pizza’s respectively

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check for number of CheeseBurst Pizza’s Size so we check with the 6^{th} field in array and map it to isCheeseBurst field. Further for each field we get the output from the mapper to give it as input to the reducer and filter out number of counts of Cheese Burst pizza’s.

Pseudocode:

```
Map (Key,Value)
LS = Split (value , ‘\t’)
isCheeseBurst = LS[6]
String CheeseBurst
```

```
if (isCheeseBurst == “Y”)
{
  isCheeseBurst = Key
  CheeseBurst = isCheeseBurst
  write (CheeseBurst , 1)
}
end Map

REDUCE (key , valuelist)
 cCount = 0
 for (i=0 to valuelist.length)
cCount = cCount + 1;
write (key , cCount )
end REDUCE
```

**********End Question 4.3**********

Question 4.4:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isCheeseBurst is “Y” , if “Y” and if the Size is “R” then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of small CheeseBurst Pizza’s
3. Reducer increments the count filters out the key and the total count of the small Cheeseburst pizza’s respectively

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values

as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check for number of small CheeseBurst Pizza’s Size so we check with the 6th field in array for Cheeseburst pizza’s and 7th field for the size of pizza’s and map it to Size field. Further for each field we get the output from the mapper to give it as input to the reducer and filter out number of counts of small Cheese Burst pizza’s.

Pseudocode:

```
Map (Key,Value)
LS = Split (value , '\t')
isCheeseBurst = LS[6]
Size = LS[7]
String SmallCheeseBurst
if (isCheeseBurst == "Y" && Size == "R")
{
  SmallCheeseBurst = Key
  write (SmallCheeseBurst , 1)
}
end Map

REDUCE (key , valuelist)
 scCount = 0
 for (i=0 to valuelist.length)
scCount = scCount + 1;
write (key , scCcount )
end REDUCE
```

**********End of Question 4.4**********

Question 4.5:

Algorithm:

1. Read Dataset as Input through Map Function with “Key” as Record identifier and the tab separated records as “Values”
2. Check if the field isCheeseBurst is “Y” , if “Y” and if the Price is less than 500 then the Input of the Reduce will be Output of the Map which will take the key and the valuelist of small CheeseBurst Pizza’s
3. Reducer increments the count filters out the key and the total count of the CheeseBurst pizza less than price of 500 respectively

Explanation:

The Inputs from the given dataset can be read through a Map Function with Key as the record identifier and the tab separated records as Values, then the values which are separated by tab are further split using the “Split function” with values as “Key” and given to a Array variable LS. Further the corresponding position in array are given to the fields in the records in this case since we need to check for number of CheeseBurst Pizza’s less than 500 price Size so we check with the 6^{th} field in array for Cheeseburst pizza’s and 9^{th} field for the price of pizza’s and map it to Price field. Further for each field we get the output from the mapper to give it as input to the reducer and filter out number of counts of small Cheese Burst pizza’s.

Pseudocode:

```
Map (Key,Value)
LS = Split (value , ‘\t’)
isCheeseBurst = LS[6]
Price = LS[9]
String CheeseBurst
if (isCheeseBurst == “Y” && Price < 500)
{
  CheeseBurst = Key
  write (CheeseBurst , 1)
}
```

```
end Map

REDUCE (key , valuelist)
 scCount = 0
 for (i=0 to valuelist.length)
scCount = scCount + 1;
write (key , scCcount )
end REDUCE
```

**********End of Question 4.5**********