Answer 1.

Map (key, value)

1. LS = split(value, \t) // reads the original data set
2. event_epoch_time = LS[1]
3. user_id = LS[2]
4. device_id = LS[3]
5. user_agent = LS[4]
6. if (event_epoch_time == NULL || user_id == NULL || device_id == NULL || user_agent == NULL)
7. write (LS) //junk records
8. stop //end of map function

Answer 2.

Map (key, value)

1. LS = split(value, '\t') //where '\t' denotes tab
2. user_id = LS [2]
3. user_agent = LS[4]
4. if (user_id != null)
    a. LS1 = split (user_agent , ':')
        i. Platform = LS1[1]
        ii. osVersion = LS[2]
        iii. application_used = LS1[3]
5. write (user_id, (osVersion , platform)
6. stop

Answer 3.

Map (key, Value)

//Reads and cleans the input dataset

1. LS = split(value , '\t')
2. event_epoch_time = List[1]
3. user_id = List[2]
4. device_id = List[3]
5. user_agent = List[4]
6. if (event_epoch_time != NULL || user_id != NULL || device_id != NULL || user_agent != NULL)
7. Write(LS)
8. stop

//Now the output of the above map function will be takes as the input of the next map function to count the number of veg and non veg pizzas sold, size wise distribution of pizzas, number of cheese burst pizzas, and also the number of cheese burst pizzas below the price of 500.

Answer 3.a

Map (key, LS)

1. is_veg = LS[12]
2. getCounter(“veg”)
3. getCounter(“nonVeg”)
4. if(is_veg ==”Y”)
    a. getCounter(“veg”).incrementBy(1)
5. else
    a. getCounter(“nonVeg”).incrementBy(1)
6. stop map function
7. print veg
8. print nonVeg

//The number of veg and non veg pizzas will be displayed in the output.

Answer 3.b

Map (key, LS)

1. size = LS[7]
2. getCounter(“regular”)
3. getCounter(“medium”)
4. getCounter(“large”)
5. if(size == “R”)
    a. getCounter(“regular”).incrementBy(1)
6. if(size == “M”)
    a. getCounter(“medium”).incrementBy(1)
7. if(size == “L”)
    a. getCounter(“large”).incrementBy(1)
8. stop map function
9. print regular
10. print medium
11. print large

//Prints the number of regular, medium and large pizzas in the output.

Answer 3.c

Map (key, LS)

1. cheeseBurst = LS[6]
2. getCounter(“isCheeseBurstCount”)
3. if(cheeseBurst ==”Y”)
    a. getCounter(“isCheeseBurstCount”). incrementBy(1)
4. stop map function
5. print isCheeseBurstCount

//Prints the number of cheese burst pizzas.

Answer 3.d

Map (key, LS)

1. cheeseBurst = LS[6]

2. pizzaSize = LS[7]
3. getCounter(“isCheeseBurstRegularCount”)
4. if(cheeseBurst ==”Y” and pizzaSize == “R”)
    a. getCounter(“isCheeseBurstRegularCount”). incrementBy(1)
5. stop map function
6. print isCheeseBurstRegularCount

//Prints the number of cheese burst pizzas with regular size. Although in the given sample set there are no such pizzas.

Answer 3.e

Map (key, LS)

1. cheeseBurst = LS[6]
2. price = LS[9]
3. getCounter(“cheeseBurstPriceCount”)
4. if(cheeseBurst==”Y” and price <500)
    a. getCounter(“cheeseBurstPriceCount”). incrementBy(1)
5. stop map function
6. print cheeseBurstPriceCount

//Prints the number of cheese burst pizzas with price lower than 500.

Answer 4.

Map (key, Value)

//Reads the input dataset and skips the junk records

1. LS = split(value , ‘\t’)
2. event_epoch_time = List[1]
3. user_id = List[2]
4. device_id = List[3]
5. user_agent = List[4]
6. if (event_epoch_time != NULL || user_id != NULL || device_id != NULL || user_agent != NULL)
7. write(LS)
8. stop

Answer 4.1

Map (key, LS)

1. isVeg = LS[12]
2. if(isVeg == “Y”)
    a. pizzaType = “veg”
3. if(isVeg == “N”)
    a. pizzaType = “non-veg”
4. write (pizzaType , 1)
5. stop

Reduce (key, valueList)

1. pizzaTypeCount = 0
2. for i = 1 to valueList.length
    a. pizzaTypeCount = pizzaTypeCount + 1
3. write (key, pizzaTypeCount)
4. stop

//Prints the count of the types of pizzas sold i.e. veg or non veg.

Answer 4.2

Map (key, LS)

1. size = LS[7]
2. if(size == “R”)
    a. pizzaSize = “Regular”
3. if(size == “M”)
    a. pizzaSize = “Medium”
4. if(size == “L”)
    a. pizzaSize = “Large”
5. write (pizzaSize , 1)
6. stop

Reduce (key, valueList)

1. pizzaSizeCount = 0
2. for i = 0 to valueList.length
    a. pizzaSizeCount = pizzaSizeCount + 1
3. write (key, pizzaSizeCount)
4. stop

//Prints the count of size wise distribution of pizzas sold.

Answer 4.3

Map (key , LS)

1. isCheeseBurst = LS[6]
2. if (isCheeseBurst == “Y”)
    a. pizzaIsCheeseBurst = “yes”
3. else
    a. pizzaIsCheeseBurst = “no”
4. write (key , pizzaIsCheeseBurst)
5. stop

Reduce (key , valueList)

1. cheeseBurstPizzaCount = 0
2. for i = 0 to valueList.length
    a. cheeseBurstPizzaCount = cheeseBurstPizzaCount + 1
3. if (key == “yes”)

    a. write (key, cheeseBurstPizzaCount)
4. stop

//Prints the number of cheese burst pizzas sold.

Answer 4.4

Map (key , LS)

1. isCheeseBurst = LS[6]
2. size = LS[7]
3. if (isCheeseBurst == “Y” and size ==”R”)
    a. cheeseBurstRegular = “yes”
4. else
    a. cheeseBurstRegular = “no”
5. write (key , cheeseBurstRegular)
6. stop

Reduce (key , valueList)

1. cheeseBurstRegularCount = 0
2. for i = 0 to valueList.length
    b. cheeseBurstRegularCount = cheeseBurstRegularCount + 1
3. if (key == “yes”)
    a. write (key, cheeseBurstRegularCount)
4. stop

// The count should be 0 as in the given data set sample, by visualisation there are no regular cheese burst pizzas.

Answer 4.5

Map (key , LS)

1. cheeseBurst = LS[6]
2. price = LS[9]
3. if (cheeseBurst == “Y” and price <500)
    a. pizzaPriceCheeseBurst = “yes”
4. else
    a. pizzaPriceCheeseBurst = “no”
5. write (key, pizzaPriceCheeseBurst)
6. stop

Reduce (key, valueList)

1. pizzaPriceCount = 0
2. for i = 0 to valueList.length
    a. pizzaPriceCount = pizzaPriceCount + 1
3. if (key == “yes”)
    a. write (key, pizzaPriceCount)
4. stop //Prints number of cheese burst pizzas with price below 500.