**Question 1:**

Write a map() function to handle the given input.

Input for your map() will be a a pair of (key, value)  
where key is the recored number of input and value  
is a single record of input; for example the first  
record will be passed to a map() function as  
(1, "20,Male,Ames,IA,USA,25000"), second record as:  
(2, "24,Male,Ames,IA,USA,29000"), and so on...

**Answer:**

The input to the all the mapper would be (key,value) pairs where the key is the record number in the data and the value is a single record of input i.e. in this case, (1, “20,Male,Ames,IA,USA,25000”) would be the input.

The mapper function will handle the data only after the following criteria is met:

1. format of the data - <age><,><gender><,><city><,><state><,><country><,><salary> and
2. where the age > 15

The mapper function can thus be written as:

map (k, v) {

// Split the words if a comma is encountered  
tokens = v.split (“,”)

// checking if the format is correct and the age > 15

//Assuming isnumeric function returns TRUE for numeric values

//Assuming ischarater function returns TRUE for character values  
if ( len(tokens) == 6 AND  
isnumeric(token[0]) AND   
token[1] in ('Male', 'Female') AND  
ischaracter(token[2]) AND  
len(token[3]) == 2 AND  
ischarater(token[3]) AND

ischaracter(token[4]) AND  
isnumeric(token[5]) AND   
token[0] > 15) //Check for age filter

{

// if the above conditions are satisfied then output the following  
             emit (token[0], token[5])  
             emit ("unique-cities", token[2])  
                        if (token[1] == "Male") {  
                              emit ("number-of-males", 1) }  
                        else {  
                                emit ("number-of-females", 1)}  
                        }  
}

The mapper function from above can be explained as below:

1.  After a key and value is provided to the mapper, the value (here a record) is split into separate words wherever a comma is encountered.

2. The next step is to verify if the input record follows the format and if the age > 15. This is done as specified above.

3. Only after all the conditions are satisfied in the if statement, then the record will emit the contents of the if statement.

4. This process will continue until all the records/ partitions are handled by the mapper.

**Question 2:** Show output of all mappers: show your work as detail as possible

**Answer:**

After the input data is partitioned and stored in different nodes, mapper function is applied at each node.

**If the input to the mapper is :**

(1, "20,Male,Ames,IA,USA,25000")  
(2, "24,Male,Ames,IA,USA,29000")  
(3, "24,Female,Sunnyvale,CA,USA,57000") and so on

**Output for the first record:**

(20, 25000)  
("unique-cities", Ames)  
("number-of-males", 1)

**Output for the second record:**

(24, 29000)  
("unique-cities", Ames)  
("number-of-males", 1)

**Output for the third record:**

(24, 57000)  
("unique-cities", Sunnyvale)  
("number-of-females", 1) and so on

**This output can be explained as follows:**

1. After the record is received as an input to the mapper, the input is first checked for its format. For checking the format, we first need to ensure that there are 6 values in the input record after removing the commas i.e. [20 Male Ames IA USA 25000].  Then we check whether each of these 6 values match the criteria for our input data and for that we check if the first value is a number, the second value is either Male or Female, the third value and the fifth values are characters, the fourth value has a length of 2 and the last value is a number. But we also want to filter out data with age < 15 and hence we check if the first value has a value greater than 15.

2.  Only after the above conditions are met, the next part is to get the desired output.  The conditions under the if statement will now be executed. Say if we have [20 Male Ames IA USA 25000]. Then for each such record the following will be emitted

(age, salary) i.e. (20, 25000)  
("unique-cities",  city) i.e. ("unique-cities", Ames)  
("number-of-males", 1) (or it can emit ("number-of-females",1) if the person is a female)

3. For getting the above output, inside the if statement, we ask the mapper to emit (age, salary) then emit ("unique-cities", city), then we check if the second value is a Male, if yes emit ("number-of-males", 1) and if not i.e. if the second value is Female then emit ("number-of-females", 1)

4. This is thus the output from the mapper.

**Question 3:** Show all of the input to "Sort & Shuffle" phase.

**Answer:**

The output from the mapper then becomes the input for the “Sort and Shuffle” phase. Hence the input for the sort and shuffle will thus be:

**Output for the first record:**

(20, 25000)  
("unique-cities", Ames)  
("number-of-males", 1)

**Output for the second record:**

(24, 29000)  
("unique-cities", Ames)  
("number-of-males", 1)

**Output for the third record:**

(24, 57000)  
("unique-cities", Sunnyvale)  
("number-of-females", 1) and so on

**Question 4:** Show output of the "Sort & Shuffle" phase.

**Answer:**

The sort and shuffle is a process of merging and sorting the mappers’ output and sending it to the reducers.

The key and all its associated values will be the output from sort and shuffle. For example:

**For the age as the key:**

(20, [25000, 24000, 26000])  
(24, [29000, 57000, 34000])   
(40, [43000, 44000, 48000])

**Similarly for "unique-cities" as the key, output will be:**  
("unique-cities", [Ames, Ames, Sunnyvale, Ames, ......, Boone])

**For "number-of-males" as the key:**  
("number-of-males", [1,1,1,1,1,1,1])

**For "number-of-females" as the key:**  
("number-of-females", [1,1,1])

The output from the sort and shuffle then goes to the reducer.

**Question 5:** Write a reduce () function to generate the desired output.

**Answer:**

At the reducer nodes, reducer function will be applied to get the desired output:

reduce (k, v) {

// Check to see if the key is age i.e. a number  
                if (isnumeric(k)) {  
                    count = 0, sum = 0  
                     for i in v {  
                     count = count + 1  
                     sum = sum + i  
                      }   
                   avg = sum/count  
                    emit (k, avg)  
                 }

// Else check if the key contains "unique-cities"  
                 else if (k == "unique-cities") {  
                    list = []

//Assuming unique function will provide all the unique values in an iterable  
                    list = list.append(unique(v))   
                     emit (k, list)  
                  }

// Else check if the key contains "number-of-males"  
                  else if (k == "number-of-males") {  
                    count = 0  
                    for i in v {  
                    count = count + 1   
                       }  
                  emit (k, count)  
                  }

// Else emit the output for "number-of-females"  
              else {   
                  count = 0  
                  for i in v {  
                  count = count + 1   
                     }  
                emit (k, count)  
              }  
}

The reducer function checks:

1. If the key is a number, if it is, then it would calculate the average salary for that age (by taking the sum/count) and emit the output as (age, average salary)

2. If the key is "unique-cities" then it takes out all the unique city values and appends it in an empty list created at the start and outputs ("unique-cities", list-of-unique-cities)

3. If the key is "number-of-males" then it sums all the values and outputs ("number-of-males", count)

4. If the key is "number-of-females" then it sums all the values and outputs ("number-of-females", count)

**Question 6:** Show output of all reducers:

**Answer:**

If the input to the reducers is :  
(20, [25000, 24000, 26000])  
(24, [29000, 57000, 34000])   
(40, [43000, 44000, 48000])  
("unique-cities", [Ames, Ames, Sunnyvale, Ames,......, Boone])  
("number-of-males", [1,1,1,1,1,1,1])  
("number-of-females", [1,1,1])

The output from all the reducers would be:  
(20, 25000)  
(24, 40000)  
(40, 45000)  
("unique-cities", [Ames, Sunnyvale, Boone, Cupertino])  
("number-of-males", 7)  
("number-of-females", 3)

The output from each of the reducer nodes will then be collated and printed/stored in a file.