# **TASK 1**

## **TASK 1.1: IMPLEMENT A PYTHON PROGRAM (EXTRACTION)**

A computer screen shot of a program code

Description automatically generated

**# Making database connection**

**# Open a txt file for sorting year and company data**

**# Loop through each movie document in the cur**

For each I in the range from 0 to min(3, the length of movie[‘companies’]):

Extract the last four characters (year) of movie[‘date’]

Extract the company name from the ‘companies’ field

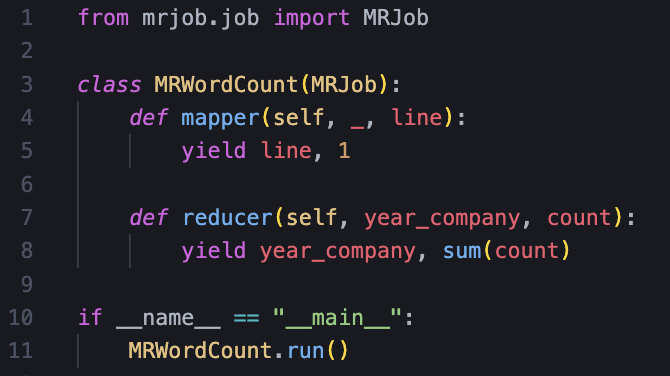
Construct a string containing the year and company name using a delimiter “,”

Split the line into year and company

**# Write the year\_company file**

**# Close the txt file**

## **TASK 1.2: IMPLEMENT THE MAPREDUCE PROGRAM (COUNT)**



Input the text file

**# Mapper function**

Input: Each line contains a record in the format "year\_company, count"

For each line in the input:

Split the line into year\_company and count using a delimiter ","

Emit key-value pair (line, 1)

**# Reducer function**

Input: List of key-value pairs

// key: year and company

//value: a list of counts

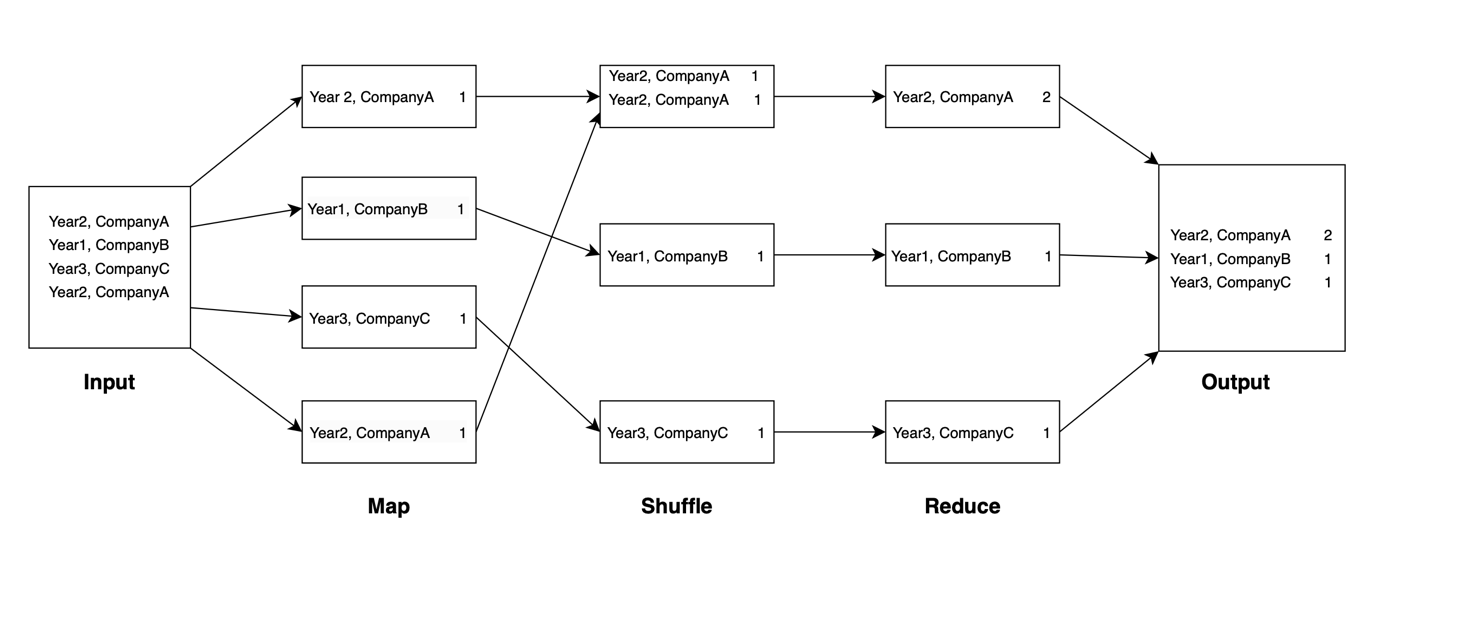
For each key-value pair in the input:

Initialize a variable sum(count) to 0

For each count in the list of counts:

Emit key-value pair (year\_company, sum(count))

### **FLOWCHART FOR TASK 1**

****

# **TASK 2**

## **TASK 2.1: MERGE SORT**



Input the txt file   
**# Steps function**

**# Mapper function**

Input: Each line contains a record in the format "year\_company, count"

For each line in the input

Split the line into year\_company and count using a delimiter “\t”

Combine year\_company and count by using “key”

Emit key-value pair (“key” (year\_company, count))

yield "key", (year\_company[1:-1], count)

**# Merge\_sort function**

If the length of array >1:

Calculate the middle index mid

Split the array into left\_half and right\_half

Call merge\_sort on left\_half and right\_half

Initialize variables i, j, and k to 0

While i < length of left\_half and j < length of right\_half:

Compare elements at indices i and j based on the second element of each tuple

Update array at index k with element from left\_half

Increment i

Else:

Update array at index k with the element from right\_half

Increment j

Increment k

Copy any remaining elements from left\_half

Copy any remaining elements from right\_half

**# Return the sorted array**  
**# Reducer function**

Input: List of key-value pairs

// key: year\_company

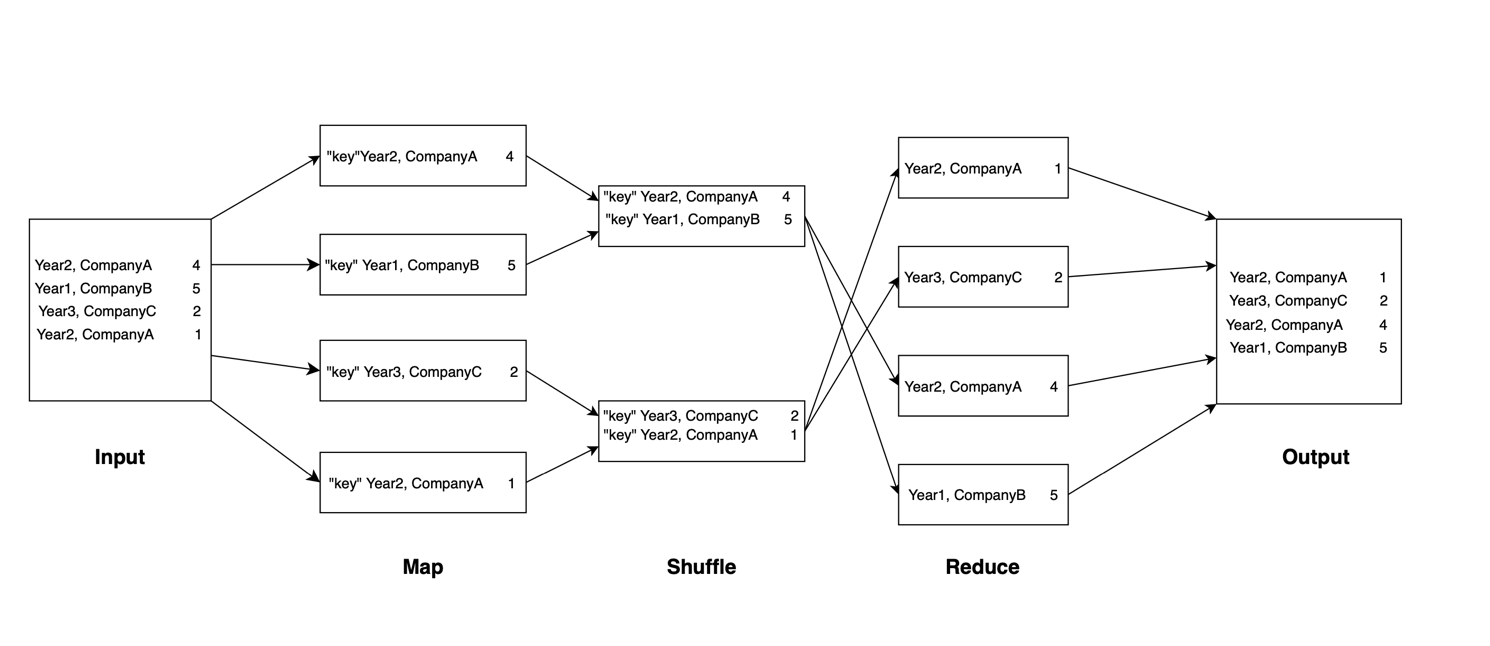
// value: count

Sort the list of tuples using merge-sort

For each tuple in the sorted list:

Emit key-value pair with the first element of the tuple (year\_company) and the second element (count)

### **FLOWCHART FOR MERGE SORT**

****

## **TASK 2.2: BUCKET SORT**

A screen shot of a computer program

Description automatically generated

Input the txt file

**# Configure command-line arguments function**

**# Steps function**

**# Mapper function**

Split the input line into year\_company and count using a delimiter “\t”

Convert count to an integer

Calculate the bucket\_id based on count and bucket size

Emit key-value pair

// key: bucket\_id

// value: count

**# Reducer function for records (count) sorting**

Sort records in descending order based on the second element of each tuple (records)

Emit sorted key-value pairs

Combine bucket\_id and record by using “key”

// key: “key”

// tuple: bucket\_id and record

**# Reducer function for bucket\_id sorting**

Sort records in descending order based on the first element of each tuple (bucket\_id)

Emit sorted sorted values from bucketid\_records

**FLOWCHART FOR BUCKET SORT A diagram of a company

Description automatically generated**