MapReduce: Two step join

- Calculate the average income in a city for the year 2007

- Use data from multiple tables

Average Income in a City

Table 1: (SSN, {Personal Information})

123456:(John Smith;Sunnyvale, CA)

123457:(Jane Brown;Mountain View, CA)

123458:(Tom Little;Mountain View, CA)

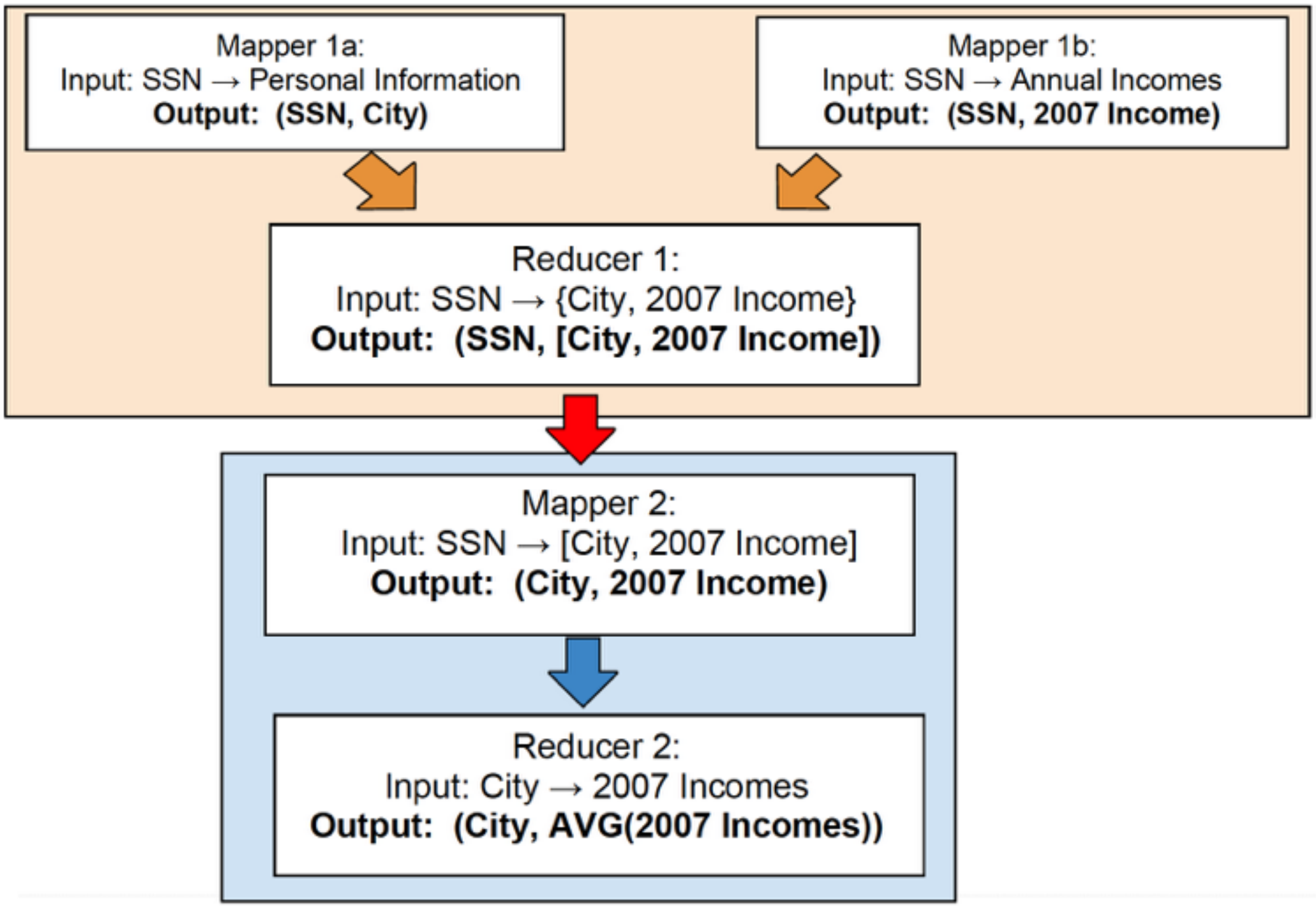
Table 2: (SSN, {year, income})

123456:(2007,$70000),(2006,$65000),(2005,$6000),…

123457:(2007,$72000),(2006,$70000),(2005,$6000),…

123458:(2007,$80000),(2006,$85000),(2005,$7500),…

Average Income in a City



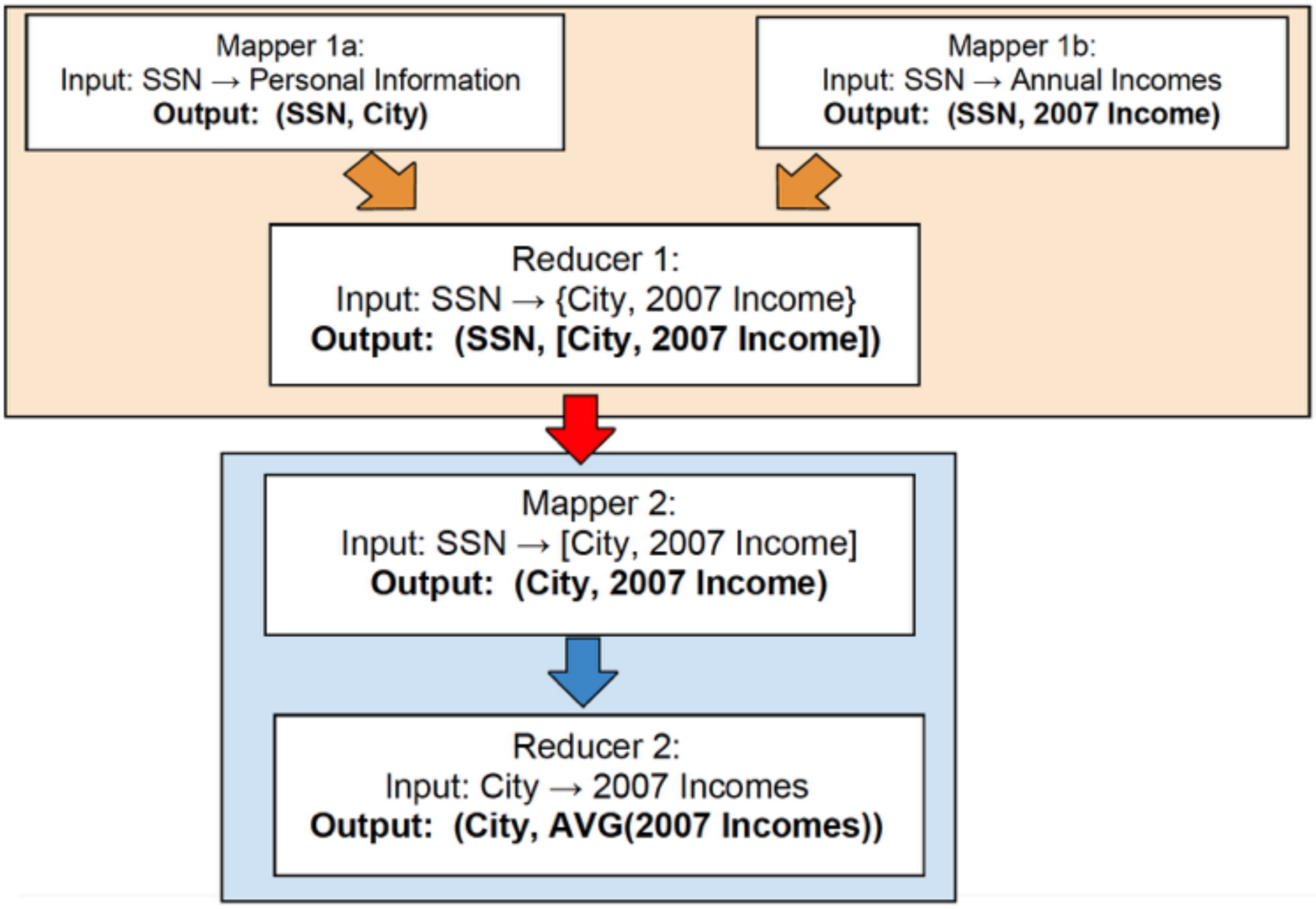
Average income: Mapper 1a

```
private static class PersonalInformationMapper extends Mapper<LongWritable, Text, Text, Text> {

    private Text ssn = new Text();
    private Text city = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] line = value.toString().split(":");
        ssn.set(line[0]);
        city.set(line[1].split(";")[1]);
        context.write(ssn, city);
    }
}
```

Average Income in a City



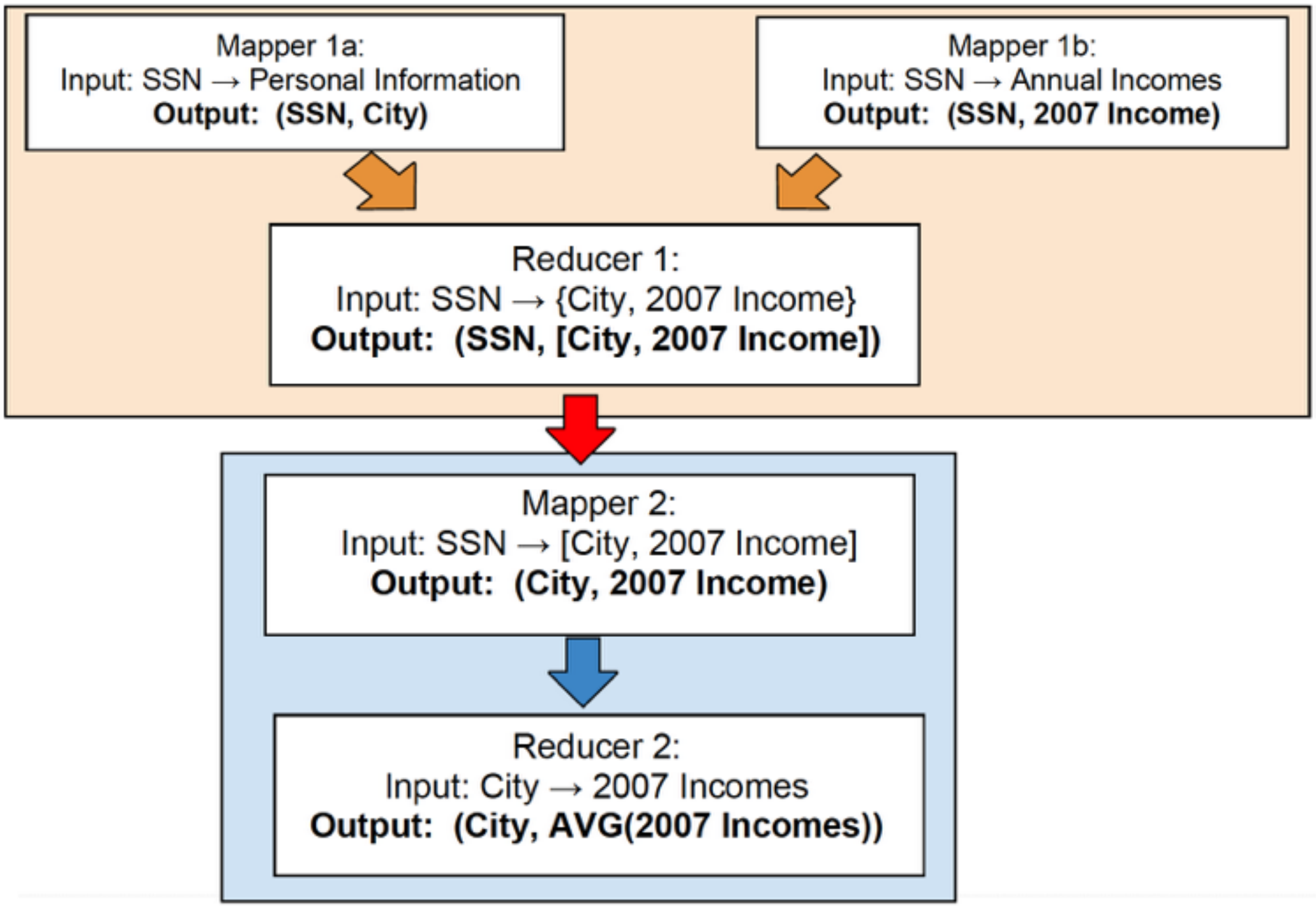
Average income: Mapper 1b

```
private static class IncomeMapper extends Mapper<LongWritable, Text, Text, Text> {

    private Text ssn = new Text();
    private Text income = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] line = value.toString().split(":");
        ssn.set(line[0]);
        String[] years = line[1].split(",");
        for (String year: years) {
            String[] record = year.substring(1, year.length() - 1).split(";");
            if (record[0].equals("2007")) {
                income.set(record[1]);
                context.write(ssn, income);
            }
        }
    }
}
```

Average Income in a City



Average income: Reducer 1

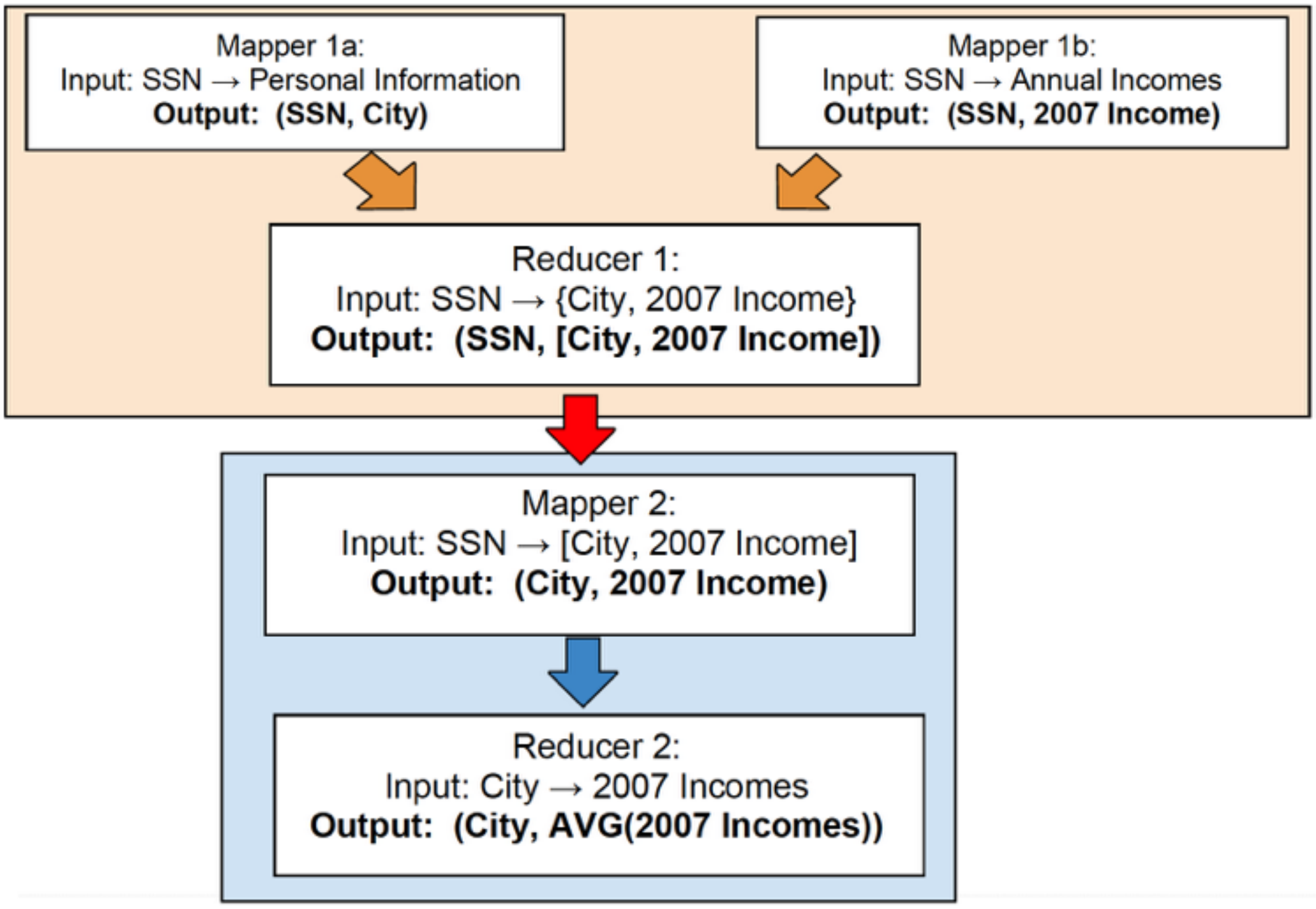
```
private static class CityIncomeReducer extends Reducer<Text, Text, Text, Text> {

    Text result = new Text();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        String income = "";
        String city = "";
        Iterator<Text> iterator = values.iterator();
        while (iterator.hasNext()) {
            String next = iterator.next().toString();
            if (next.matches("\\d+")) {
                income = next;
            } else {
                city = next;
            }
        }

        result.set(city + "," + income);
        context.write(key, result);
    }
}
```

Average Income in a City



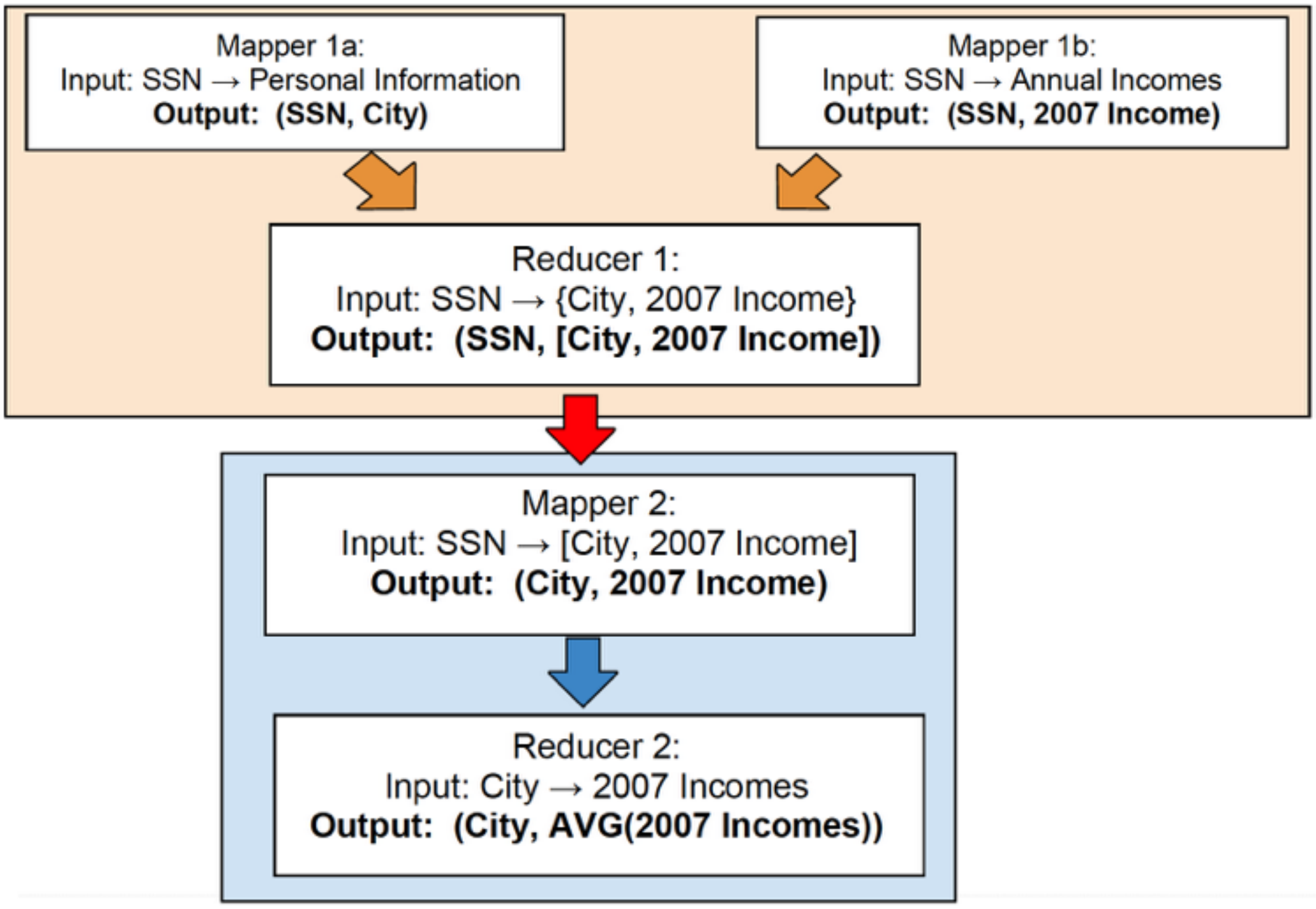
Average income: Mapper 2

```
private static class CityIncomeMapper extends Mapper<LongWritable, Text, Text, Text> {

    Text city = new Text();
    Text income = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] record = value.toString().split("[\t,]");
        city.set(record[1]);
        income.set(record[2]);
        context.write(city, income);
    }
}
```

Average Income in a City



Average income: Reducer 2

```
private static class CityAvgIncomeReducer extends Reducer<Text, Text, Text, Text> {

    Text averageIncome = new Text();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        List<Long> incomes = new ArrayList<>();
        Iterator<Text> iterator = values.iterator();
        while(iterator.hasNext()) {
            Text next = iterator.next();
            incomes.add(Long.parseLong(next.toString()));
        }
        double avg = incomes.stream().mapToLong(Long::longValue).average().getAsDouble();
        averageIncome.set(Double.toString(avg));
        context.write(key, averageIncome);
    }
}
```