EXPERIMENT - 6

Aim:

Develop a MapReduce program to find the maximum electrical consumption in each year given a consumption dataset.

Theory:

Analyzing utility usage data is a common Big Data use case. The objective is to aggregate large volumes of consumption logs (indexed by year) to determine peak usage periods. In MapReduce, the 'Map' phase extracts the Year as the Key and the Consumption Units as the Value. The 'Shuffle and Sort' phase groups all consumption readings for a specific year. The 'Reduce' phase then iterates through these readings to find the maximum integer value, representing the highest consumption for that year.

Description:

The program processes a text file where each line represents a record containing the Year and the Electricity Consumption (in units/kWh). The Mapper tokenizes the line, extracts the Year (index 0) and Consumption (index 1), and emits a <Text, IntWritable> pair. The Reducer receives <Year, List<ConsumptionValues>>, iterates through the list, finds the maximum value using standard comparison logic, and writes the final Key-Value pair to the output.

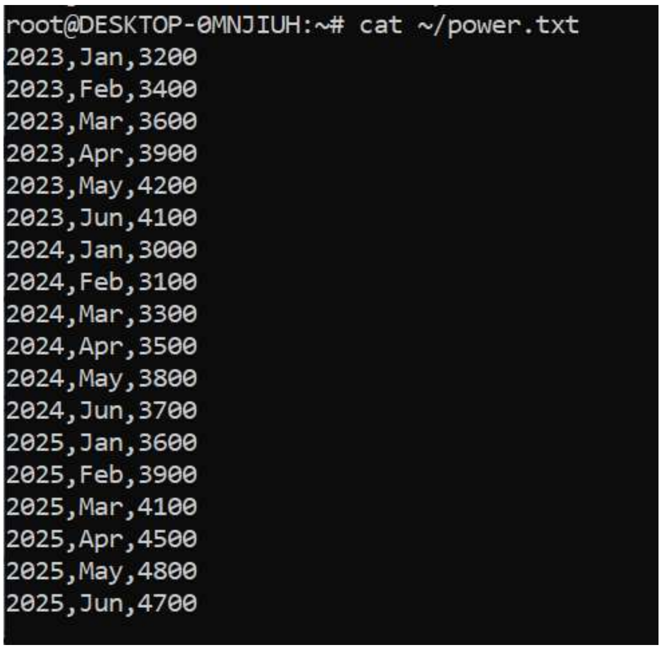
Code:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxElectricity {

    // 1. Mapper Class
    public static class ElectricMapper extends Mapper<LongWritable, Text,
Text, IntWritable> {

        private Text year = new Text();
        private IntWritable consumption = new IntWritable();

        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            String line = value.toString();
            // Assuming Input Format: Year,Consumption (e.g., "2020,4500")
            String[] parts = line.split(",");

            if (parts.length >= 2) {
                try {
                    year.set(parts[0].trim());
                    int units = Integer.parseInt(parts[1].trim());
                    consumption.set(units);
                    context.write(year, consumption);
                } catch (NumberFormatException e) {
                    // Ignore malformed lines
                }
            }
        }
    }

    // 2. Reducer Class
    public static class ElectricReducer extends Reducer<Text, IntWritable,
Text, IntWritable> {

        public void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
            int maxConsumption = 0;

            for (IntWritable val : values) {
                if (val.get() > maxConsumption) {
                    maxConsumption = val.get();
                }
            }
            context.write(key, new IntWritable(maxConsumption));
        }
    }

    // 3. Driver Method
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Max Electricity Consumption");

        job.setJarByClass(MaxElectricity.class);
        job.setMapperClass(ElectricMapper.class);
        job.setCombinerClass(ElectricReducer.class); // Optional: Combiner
helps optimization
        job.setReducerClass(ElectricReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

```
}
```

Input:

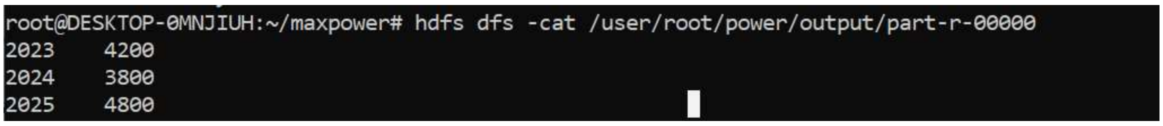
A text file named consumption.txt containing (Year, Units):



```
root@DESKTOP-0MNJIUH:~# cat ~/power.txt
2023,Jan,3200
2023,Feb,3400
2023,Mar,3600
2023,Apr,3900
2023,May,4200
2023,Jun,4100
2024,Jan,3000
2024,Feb,3100
2024,Mar,3300
2024,Apr,3500
2024,May,3800
2024,Jun,3700
2025,Jan,3600
2025,Feb,3900
2025,Mar,4100
2025,Apr,4500
2025,May,4800
2025,Jun,4700
```

Output:

The content of the part-r-00000 file generated by Hadoop:



```
root@DESKTOP-0MNJIUH:~/maxpower# hdfs dfs -cat /user/root/power/output/part-r-00000
2023    4200
2024    3800
2025    4800
```

Commands:

Description	Command
Create input directory:	hdfs dfs -mkdir /electric_input
Upload input file:	hdfs dfs -put consumption.txt /electric_input
Compile Java program:	javac -classpath $(hadoop classpath) -d . MaxElectricity.java
Create JAR file:	jar -cvf electric.jar *.class
Run Hadoop Job:	hadoop jar electric.jar MaxElectricity /electric_input /electric_output
View Output:	hdfs dfs -cat /electric_output/part-r-00000

Conclusion:

In this experiment, we successfully developed a MapReduce program to analyze electrical consumption data. We learned how to handle comma-separated data in the Mapper and how to apply aggregation logic (finding the maximum) in the Reducer phase to solve real-world data analysis problems.