MDS
Winter Term 2024/25
Translating SQL to MapReduce Workflows

You work with the following relational schema (the primary key is underlined):

$$Students(\underline{sid}, s\_name, street, city, age)$$

We consider the following cardinalities (just toy numbers, for the sake of this exercise):

- The *Students* relation has 10,000 tuples.
- $|\pi_{city}(Students)| = 200$, i.e. there are 200 distinct values for city.
- $|\pi_{age}(Students)| = 50$.

Let's assume that 200 tuples from `Students` fit into one HDFS chunk.

We now consider the following SQL query:

```
SELECT count(*)
FROM Students
WHERE city IN ('Springfield', 'Los Angeles', 'Annapolis')
      AND age >= 30;
```

1. Sketch the physical query plan obtained by (1) the canonical translation to relational algebra, (2) selection pushing, (3) mapping each relational operator to a MapReduce job. It is enough to provide the result of step (3).

   How many Map tasks will be started by Hadoop to execute the first stage in the bottom-up evaluation of this plan? Give a simple estimate and briefly justify your answer.

Next, provide the MapReduce function code that computes the aggregation. If you don't need a Reducer, simply cross out the respective code.

The input data is available in HFDS as key-value pairs, as shown below. The key is the identifier of the HDFS chunk that this tuple is stored in, the value is a JSON-encoding of the tuple:

```
part-000 {"sid": 42, "s_name": "Lisa", "street": "742 Evergreen Terr.", "city": "Springfield", ...}
part-000 {"sid": 23, "s_name": "Alf",  "street": "16 Hemdale Street",   "city": "Los Angeles", ...}
part-000 {"sid": 52, "s_name": "Dana", "street": "3170 W. 53 Rd. #35",  "city": "Annapolis", ...}
part-000 {"sid": 53, "s_name": "Bob",  "street": "3170 W. 53 Rd. #35",  "city": "Annapolis", ...}
```

The output data should be formatted like this if the result were "5".

```
result    5
```

For simplicity, we assume that the aggregation can be performed within a single MapReduce job (rather than two chained MapReduce stages). Complete the Python skeleton code. Refer to the appendix to see the "wordcount" example with `luigi` code.

```
class CountAggr(luigi.contrib.hadoop.JobTask):

    ...

    def mapper(self, line):
        partid, tuple = line.split('\t')
        json_tuple = json.loads(tuple) # Python dictionary encoding 1 tuple.




        # e.g. yield(partid, json.dumps(json_tuple)) outputs the input
        ''' ................. fill in your code above ..................'''


    def reduce(self, key, values):

        ''' ................. fill in your code below ..................'''









        # e.g. yield("result", 5)
```

In your implementation, does it make sense to re-use the Reducer function code *unchanged* for a Combiner? Briefly justify your answer.

2. Let's consider class `CountAggr` just implemented (not using a combiner).

   What is the communication cost of the implemented algorithm? Measure the cost in number of tuples.

Appendix

The `WordCount` example as a `luigi` MapReduce job:

```
class WordCount(luigi.contrib.hadoop.JobTask):

    # Not shown: Declaring dependencies and output of this task.
    # ....

    def mapper(self, line):
        for word in line.strip().split():
            yield word, 1

    def reducer(self, key, values):
        yield key, sum(values)
```