Task 2 (30%). Columnar representation over Apache Spark

Implement a columnar representation over Spark, which will allow loading and querying data. The queries that you need to support will be simple SQL queries of the following form:

```
SELECT attr1, attr2, ...
FROM table
WHERE cond1 AND cond2 AND …
```

Questions:

a. Suppose that you are given an input CSV file and an input schema of the form "attr1:DataType,attr2:DataType,…", where DataType can take the values Int, String, Float. For example, an input schema could be "name:String,age:Int". Load the input file using Spark RDDs, such that each column corresponds to a separate RDD. The mapping of columns to records will be done using virtual IDs.

b. Given the structure of a SQL query, compute the answer of the query using the representation of question a. The structure of the input SQL query will involve the following parts:
   - a *projectionList* corresponding to the attributes that will be returned to the user separated by comma. For example "attr1,attr3".
   - a *whereList* corresponding to the conditions involved in the query separated by comma. The comma of the *whereList* represents the AND clause. The conditions that you need to support are conditions of the following forms:
     i. *attr*=value
     ii. *attr<value*
     iii. *attr<=value*
     iv. *attr>value*
     v. *attr>=value*

     each condition will be given in the form attr|op|value. For example, a possible whereList could be "attr1|=|3,attr2|<|7".

   The plan that you will follow for the query evaluation will first load the required attributes from the input file, then the plan will evaluate the WHERE clause and at the end it will materialize the results.

Output: The format of the result must be CSV (separated by ',').

Note: Your implementation must rely only on Spark RDDs. You are not allowed to use Spark SQL and Dataframes.

Data: "lineitem.csv" (download here), CSV format where the fields of each tuple are separated by ','. The relation LINEITEM has the following schema:

```
LINEITEM( L_ORDERKEY          INTEGER NOT NULL,
          L_PARTKEY           INTEGER NOT NULL,
          L_SUPPKEY           INTEGER NOT NULL,
```

```
L_LINENUMBER        INTEGER NOT NULL,
L_QUANTITY          DECIMAL(15,2) NOT NULL,
L_EXTENDEDPRICE     DECIMAL(15,2) NOT NULL,
L_DISCOUNT          DECIMAL(15,2) NOT NULL,
L_TAX               DECIMAL(15,2) NOT NULL,
L_RETURNFLAG        CHAR(1) NOT NULL,
L_LINESTATUS        CHAR(1) NOT NULL,
L_SHIPDATE          DATE NOT NULL,
L_COMMITDATE        DATE NOT NULL,
L_RECEIPTDATE       DATE NOT NULL,
L_SHIPINSTRUCT      CHAR(25) NOT NULL,
L_SHIPMODE          CHAR(10) NOT NULL,
L_COMMENT           VARCHAR(44) NOT NULL)
```

Deliverables:

- ColumnStore.java, the program should receive five arguments:
  - <input>: denoting the input dataset.
  - <output>: denoting the output file.
  - <schema>: a string containing the schema of the dataset (e.g., “attr1:String,attr2:Int”).
  - <projectionList>: a string containing the arguments projected (e.g., “attr1,attr3”).
  - <whereList>: a string containing the where clause (e.g., “attr1|=|3,attr2|<|7”).

Grading: We will harshly penalize submissions for which the input, output paths, or other parameters are hardcoded or are not abiding with the required format.