CSE 111 – DATABASE SYSTEMS
Final Exam

Consider the following relational schema:

Product (*model*, *type*, *maker*)
Distributor (*model*, *name*, *price*)
Price_Cube (*distributor_type*, *product_type*, *num_prod*, *tot_price*)

in which the keys are underlined. *Distributor.model* is a foreign key to *Product.model*. Referential integrity is enforced through `CASCADE` operations.

Product and **Distributor** are base tables, while **Price_Cube** is a data cube with two dimensions – *distributor_type* and *product_type* – and two measure attributes—*num_prod* and *tot_price*. *distributor_type* takes only two values – "`producer`" and "`distributor`" – where "`producer`" corresponds to a product maker, while "`distributor`" to an entity that does not make any product. *product_type* corresponds to attribute *type* from **Product**, which can be only one of {"`pc`", "`laptop`", "`printer`"}. *num_prod* is the total number of products of a given type sold by "`producer`"/"`distributor`", while *tot_price* is the total price corresponding to these products. Given these, we know that **Price_Cube** has exactly 12 tuples, including the values "`ALL`" or "`*`".

Sample data for the base tables **Product** and **Distributor** is given below:

model	*type*	*maker*
100001	pc	P1
100002	pc	P2
100003	pc	P2
100004	pc	P3
100005	pc	P1
200001	laptop	P4
200002	laptop	P5
200003	laptop	P1
200004	laptop	P3
300001	printer	P6
300002	printer	P6
300003	printer	P2

model	*name*	*price*
100001	P1	NULL
100002	P2	2874
100003	P2	NULL
100004	P3	NULL
100005	P1	878
200001	P4	534
200002	P5	998
200003	P1	NULL
200004	P3	NULL
300001	P6	NULL
300002	P6	NULL
300003	P2	287
100001	D1	1239
100003	D2	1645
100004	D2	674
100003	D3	1245
100004	D3	874
200003	D1	1875
200004	D3	3421
300001	D2	223
300002	D2	167

The model for a `pc` starts with 1 and contains 6 digits; for a `laptop` starts with 2 and contains 6 digits; for a `printer` starts with 3 and contains 6 digits. It is important to notice that the `maker` of a product is always a `distributor` of the product, even when the `price` is unknown, i.e., `NULL`. This constraint has to be maintained at all times as follows. For every single tuple in **Product**, there is a corresponding tuple in **Distributor** having the same value for `model` and `name` in **Distributor** is equal to `maker` in **Product**. The value of `price` is set to `NULL` by default. `INSERT` on **Distributor** with the `maker` of a product is treated as

an `UPDATE` of the `price` attribute. `DELETE` from **Distributor** with the `maker` of a product is treated as an `UPDATE` of the `price` to `NULL`.

Given the skeleton code in `Final.java` and `Final.py`, you have to implement the following methods/functions:

- `create_tables` creates the tables in the schema. Additionally, index, view, and trigger creation statements should also be included here. Their creation is entirely up to you.
- `populate_tables` populates the base tables **Product** and **Distributor** with corresponding data provided in the files `product.txt` and `distributor.txt`, respectively. You have to read the tuples from these files and load them in the database, while making sure that the constraints are satisfied. Your code should work in the general case, not only for the provided sample files. The testing will be done on different data.
- `build_data_cube` builds the **Price_Cube** data cube from the base tables. This does not necessarily have to be materialized.
- `print_Product` prints the tuples in the **Product** table.
- `print_Distributor` prints the tuples in the **Distributor** table.
- `print_Cube` prints the tuples in **Price_Cube**.
- `modifications` performs the `INSERT` and `DELETE` operations specified in file `modifications.txt`. There is an operation specified on every line. The operation gives the table on which the operation is performed, followed by the operation type (`I` or `D`), and the arguments of the operation. When executing these operations, all the constraints have to be satisfied. While we provide you sample operations, your code will be tested on a different set of operations. Thus, it has to be general. However, the only operations are `INSERT` and `DELETE` on **Product** and **Distributor**, respectively.

Your code has to be written in the above methods/functions. Additionally, you can create whichever methods/functions you find necessary. However, you cannot change the `main` method/function. Your code will be tested on an empty database by running the file `./test.sh` in the terminal (this file is provided). We will check the output of your code based on different input files. Moreover, make sure that your code is safe to SQL injection attacks. Finally, your submission will be run through a plagiarism detection software.

The only file you have to submit is `Final.java` or `Final.py`, whichever you decide to implement. Do not include any other files in your submission because we will not consider them. We will exclusively grade your code with `./test.sh` in the terminal.