CS5421 Project 3 - XML
Tan Lai Chian Alan
A0174404L

Question 1a

The SQL query should be self explanatory for most parts.
We use xmlagg to aggregate the list of warehouses and the list of items.
We construct our XML document using the join of all 3 tables, Stock, Warehouse, and Item.

Question 1b

Our XML document is derived from the join of all 3 tables, Stock, Warehouse, and Item. By definition of lossless, it means that from our XML document, we should be able to reconstruct the tables Stock, Warehouse, and Item without any loss of information (e.g. we can recover all the items in the original Item table using the XML document).

For the given datasets in table Stock, Warehouse, and Item, since every row in the Item table is also listed in the Stock table, the XML document is lossless.

However, if we were to insert a new item in Item table and in the case where this new item is not available in any warehouse, it means that there is no s_qty and hence this item will not be listed in Stock table. In the case, when we construct our XML document, this new item will not be found in our XML document, making our XML document lossy.

For example, let's insert a new row into the Item table.

```
INSERT INTO item VALUES (51, 12345678, 'Panadol', 1.00);
```

Then using the follow SQL query, we can extract all the missing items in our XML document.

```
SELECT *
FROM item
WHERE i_id NOT IN (
        SELECT distinct(i_id)
        FROM stock
)
```

Question 2

We declare root element as warehouses using DOCTYPE
We declare warehouses to contain zero or more warehouse elements using *
We declare each warehouse element to contain exactly one element of each -- id, name, address, and items
We declare each address element to contain exactly one element of each -- street, city, country

We declare items to contain zero or more item elements using *
We declare each item element to contain exactly one element of each -- id, im_id, name, price, qty
We declare each element id, name, street, city, country, im_id, price, and qty to contain text to be parsed by a parser

Question 3a

First, we apply a template to get the list of warehouse elements whose country = ‘Singapore’.
Second, we print the warehouse name and country, and apply a template for the items element.
Third, we apply a template to get the list of item elements whose qty element value is above 975
Finally, we print the item name and the quantity value.

Question 3b

First, we apply a template to get the list of warehouse elements whose country = ‘Singapore’ or country = ‘Malaysia’.
Second, we print the warehouse name and country, and apply a template for the items element.
Third, we apply a template to get the item element whose qty is the highest using negation.
Finally, we print the item name and the quantity value.

Question 3c

We can directly using XPATH to compute the sum of all items named ‘Sunscreen’ whose warehouse is in ‘Indonesia’