```
A small bakery owner with 3 stores would like to implement a Point-of-Sales
(POS) system to track sales & inventory of the business. Every day, there are a
limited number of cakes being produced for each store. Design a relational
database design to help the bakery owner implement the POS. List down the
required tables and columns along with their data types.

You may assume that cakes have some types (ex. Chocolate, Pineapple, Truffle
etc) and sizes (0.5kg, 1kg, 2kg etc)

Based on your relational design, the store owner should be able to answer
questions like following. You have to write SQL statements for answering these
questions. 1. Find the total revenue from all stores for a given month. 2. Find
revenue per month per branch ordered by branch with maximum revenue at the top.
3. Find the best selling cake per branch in a given month. Best selling is
calculated based on number of units sold.
--------------------------------------------------------------------------------
```

Store

-> id (int) (PK)

-> name (varchar(255))

-> location (varchar(255))

Category

-> id (int) (PK)

-> name (varchar(255))

Why category?

Items in bakery -

Deserts – cake, cheesecake, pies

breads – bagles, buns, rolls, loaf breads

sweet goods – doughnuts, sweet rolls

ex:

id	**name**
1	deserts
2	breads

Item

-> id (int) (PK)

→ category_id (int)

-> name varchar(255))

-> price (int)
-> weight (numeric)
> UNIQUE(category_id, name, weight)
> FK(category_id)

why UNIQUE(category_id, name, weight)?

id	**category_id**	**name**	**price**	**weight**
1	1	Choclate cake	100	0.5
2	1	Choclate cake	200	1
3	1	Truffle	300	1
4	2	bun	10	-

I’m consedering same cake with different weights as new item.

inventory

-> id   (int) (PK)
-> store_id (int)
-> item_id (int)
-> quantity (int)
> FK(store_id, item_id)

why store_id?
What if we want to get stock of item(s) in specific store?

id	**store_id**	**item_id**	**quantity**
1	2	1	5
2	3	1	8
3	2	3	4
4	1	2	10

sale

→ id (int) (PK)
→ store_id (int)
→ item_id (int)
→ date (timestamp)
→ quantity (int)
→ total_price (int)
> FK(store_id, item_id)

what is ‘total_price’?

total_price = quantity * (SELECT price FROM item where id = item_id;)

1. Find the total revenue from all stores for a given month.

SELECT SUM(total_price) as 'revenue' FROM sale

WHERE EXTRACT(MONTH FROM date) = 'given-month' ;

2. Find revenue per month per branch ordered by branch with maximum revenue at the top.

```
SELECT store_id, EXTRACT(MONTH FROM date) as month,
                SUM(total_price) as revenue FROM sale
GROUP BY store_id, EXTRACT(MONTH FROM date)
ORDER BY store_id, revenue DESC;
```

3. Find the best selling cake per branch in a given month.

```
SELECT store_id, item_id,
MAX(COUNT(item_id)) as 'No. Of times item sold' FROM sale
where EXTRACT(MONTH FROM date) = 'given-month' GROUP BY
store_id, item_id;
```

garbage

-> id (int) (PK)
-> store_id (int)
-> item_id (int)
-> date (timestamp)
-> quantity (int)
-> reason_id
> FK(store_id, item_id, reason_id)

reason

-> id (int) (PK)
-> description (varchar(255))

why 'reason'?
There might be a well know reasons as follows -

id	**description**
1	Baking mistake.

2	Self item expiry.