mt_order

PK	order_id	SERIAL
FK	customer_id	INTEGER
Key	total_amount	NUMERIC(8,2)

there will be many
different foods in mt_item,
but each food can
only be connected
to one specific item_id

mt_concessions

PK	food_id	SERIAL
Key	food_name	VARCHAR(50)
Key	food_price	NUMERIC(8,2)
FK	item_id	INTEGER

an order can contain many
items, but there is only
one order because
each order is unique

a customer can make many
orders, but each order has
only one customer

mt_item

PK	item_id	SERIAL

FK order_id   INTEGER

mt_movie

PK	movie_id	SERIAL
Key	movie_title	VARCHAR(50)

mt_customer

PK	customer_id	SERIAL
Key	name_on_card	VARCHAR(50)

mt_ticket

PK	ticket_id	SERIAL
Key	ticket_price	NUMERIC(8,2)
FK	movie_id	INTEGER
FK	item_id	INTEGER

a ticket can only
refer to one movie,
but one movie
can have many tickets

there will be many
tickets in mt_item,
but each ticket can
only be connected to one
speicifc item_id

mt_item acts as an "inventory" where each ticket or food item has it's own unique id. This is probably not realistic, but I decided to use this instead of having an order table and a purchase table. If I had an order table and a purchase table, then the order would be the joining table instead of mt_item? I am still wrapping my head around how this works.... How do I connect my customer to an order, and calculate the total of an order without doing the math myself?