Customer

PK	customer_id	SERIAL
	first_name	VARCHAR(50)
	last_name	VARCHAR(50)
	email	VARCHAR(100)
	phone	VARCHAR(25)

Movie

PK	movie_id	SERIAL
	movie_name	VARCHAR(100)
	movie_genre	VARCHAR(25)
	movie_rating	VARCHAR(10)

Staff

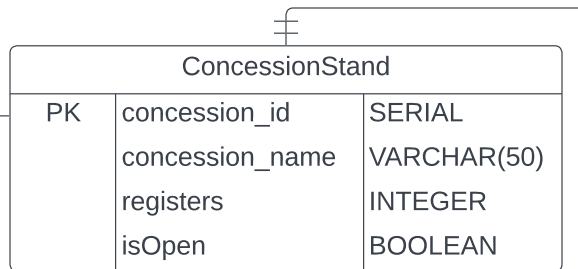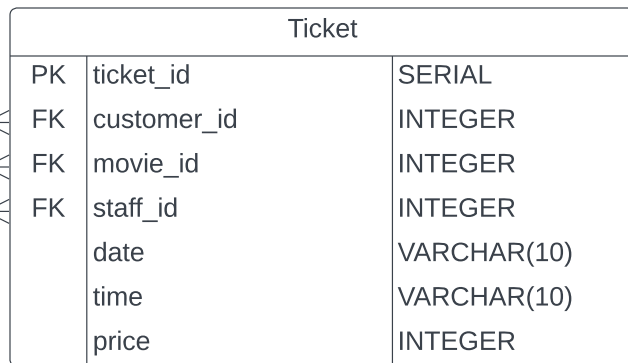| PK	staff_id	SERIAL
	first_name	VARCHAR(50)
	last_name	VARCHAR(50)
	email	VARCHAR(100)
	phone	VARCHAR(25)
FK	concession_id	INTEGER

So, the ticket table has a one and only one relationship with customer, movie, and staff. Sold to 1 customer, by 1 employee, for 1 movie,. The customer has a one or many relationship in return, because you aren't considered an actual customer unless you bought something. The movie and staff share a zero or many relationship, because it's possible for a movie to not sell at all.

Ticket

PK	ticket_id	SERIAL
FK	customer_id	INTEGER
FK	movie_id	INTEGER
FK	staff_id	INTEGER
	date	VARCHAR(10)
	time	VARCHAR(10)
	price	INTEGER

ConcessionStand

PK	concession_id	SERIAL
	concession_name	VARCHAR(50)
	registers	INTEGER
	isOpen	BOOLEAN

Product

PK	product_id	SERIAL
	product_brand	VARCHAR(50)
	product_name	VARCHAR(50)
	price	INTEGER
FK	concession_id	INTEGER

I debated for a while whether concessions should have a "products" column. In the end, I decided that each product should have a foreign key that points to the concession stand that it's available for purchase at. This just made more sense for me. In return, concessions has a zero or many relationship, because it's possible to not sell anything. Staff shares a zero or one relationship with concessions because you may or may not be assigned there for the day, and if you are, you can only mind one stand at a time. Whereas a concession stand can have many employees.