Paula’s Pizza Palace

Nick Herman

Brian Walsh

URL: https://web.engr.oregonstate.edu/~walsbria/cs340/pizzas.html

Feedback:

Marc Belinga:

- *Does the UI utilize a SELECT for every table in the schema?* In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Yes it does

- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*

Yes

- *Does the UI implement an INSERT for every table in the schema?* In other words, there should be UI input fields that correspond to each table and attribute in that table.

Yes

- *Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?* In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

No, the INSERT for a new order does not include INSERT statements for the intersection table (OrderDetails).

- *Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?* In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes, there is but no cascade delete

- *Is there at least one UPDATE for any one entity?* In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Yes, there is

- *Is at least one relationship NULLable?* In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

Not that i can see

- *Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.*

the UI is easy to navigate, just add the needed functionalities and you will have a good project in your hands

Trevor Dunn:

- *Does the UI utilize a SELECT for every table in the schema?* In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Looking through the UI, yes, there is a SELECT for every table in the schema. A few tables are included in other tables/pages. This is done in a logical manner, which is fine per the Ed post on Project Step 3. Pizzas and pizzaToppings have a M:M relationship and it makes sense to display them on a single page.

- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*

Yes. Update Customer, Delete Customer, and Create New Order all require a dynamically populated list of properties that are placed in drop-down menus. There is a SELECT in the DML that utilizes a search/filter to update orders as well. I did not see this on the HTML page, so it might be worth adding.

- *Does the UI implement an INSERT for every table in the schema?* In other words, there should be UI input fields that correspond to each table and attribute in that table.

Yes, there is an "add ____" button which correlates to an INSERT for every table.

- *Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?* In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

I think this part could be improved. For example, when INSERTing into Pizzas, the toppings field could be filled in as well.

- *Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?* In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

There is one DELETE, for customers. However, there are no DELETEs in the DML or HTML that would relate to a M:M relationship. I recommend that a DELETE be added to something with an M:M relationship.

- *Is there at least one UPDATE for any one entity?* In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Yes, there is an UPDATE for customers in both the HTML and the DML. There is an additional UPDATE in the DML for Orders, but I do not see it in the HTML yet.

- *Is at least one relationship NULLable?* In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

I do not think there is a NULLable relationship. Updating Orders only changes the price of the order. Updating Customers only changes the details of that Customer, but does not manage any optional relationships. Perhaps by adding functionality for more attributes of Orders to be updated, this requirement would be fulfilled.

- *Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.*

I think the HTML is great. It is very clear what each table will eventually be pulling from. There is a clear relationship between the schema, HTML, and DML. I would consider using the course BSG files as inspiration, citing them as a source, and adding some functionality to display OrderItems as Order Details by clicking a button and pulling up only that one Order. This would be a more user-friendly way of displaying data and would reduce page clutter.

Overall, I think this looks great. All this talk of pizza is making me hungry. Nice wor

Damiant Chauhan:

- *Does the UI utilize a SELECT for every table in the schema?* In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.
    - There seems to be a select for every table in the schema
- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*
    - Yes, create new order requires a dynamically populated list
- *Does the UI implement an INSERT for every table in the schema?* In other words, there should be UI input fields that correspond to each table and attribute in that table.
    - Yes, you can add on the UI
- *Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?* In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).
    - The inserts do add the corresponding FK attributes and one M:M relationship
- *Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?* In other words, if an order is deleted from the Orders table, it should also delete

the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

  - Yes there is a delete that removes

- *Is there at least one UPDATE for any one entity?* In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?
  - Yes there is an update in the DML
- *Is at least one relationship NULLable?* In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.
  - Yes, there is at least one relation that is NULLable
- *Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.*
  - There UI looks great and it goes well with there DDL and DML

Rebecca Chen:

- ***Does the UI utilize a SELECT for every table in the schema?* In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.**
  - Yes there is SELECT present for every table in the schema I see in the UI. I can see SELECT, meaning all data shown for tables: Customers, Drivers, orders, Pizzas&Toppings. This means all the data is shown via SELECT. This is present in the **DML as well.**
- ***Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?***
  - No there is no SELECT on the UI end to dynamically utilize a search/filter with for selected entries. The UI simply shows a prepopulated table, containing all data for each table, but there is no way on the UI end to filter the entries. I recommend you add the UI option (and the corresponding SQL) later to allow for dynamic population.
- ***Does the UI implement an INSERT for every table in the schema?* In other words, there should be UI input fields that correspond to each table and attribute in that table.**
  - Yes there is INSERT for every table on the schema via the UI/website end. The fields to INSERT are available where one can enter the attributes in to create a new entry for each table in the schema.
- ***Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?* In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT**

row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

  - Each INSERT in every table does add the corresponding FK attributes. For example, in the Orders table, you can add the CustomerID, Driver ID for the FK values. It also appears to be that the Orders table could be a M:M relation for Customers and Drivers, since there are two FKs in there, however this is not an intersection table. You might consider adding an intersection table in your schema.

- ***Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?* In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.**
  - There is one DELETE option for customers table, however this one DELETE doesn't correspond to the M:M relationship. The plausible M:M could be in the Orders or OrderDetails table, but it is not properly written in the DML and not shown in the UI as well. Thus I recommend creating an M:M relationship that can be deleted.
- ***Is there at least one UPDATE for any one entity?* In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?**
  - Yes there is at least one UPDATE for any one entity. This is shown in the Customers entity whee one can update the name, address, phone number, and email of a customer.
- ***Is at least one relationship NULLable?* In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.**
  - There is no NULLable relationship present. No relationship is considered optional via the schema. Only some attributes are, like address, phoneNumber, emailAddress, but not one whole entity is considered optional for another.
- ***Do you have any other suggestions for the team to help with their HTML UI? For example using AS aliases to replace obscure column names such as fname with First Name.***
  - One suggestion I have is to try splitting the tables of Orders and OrderDetails into their own subpages. Per assignment directions, each entity needs its own page. I would also recommend the same for the Pizzas&Toppings subpage, as it holds two entities in their as well. Likewise, I recommend creating an intersection table as well to facilitate the M:M relationship between Customers and Drivers, and have that be its own page as well.

Actions based on feedback:

1. Add a ‘toppings’ entity to serve as a category table. This entity can track quantities of toppings and can allow for inventory reports and prompt the pizza palace to order more when required.
2. Add a pick-up option. This will be accomplished by creating a dummy ‘drivers’ entry named “pick-up” that can be assigned to an order.
3. Add a boolean attribute to the orders entity to record if an order was cancelled.
4. Change the data type for the order timestamp to ‘datetime’ to improve searchability.
5. **Added a select last insert ID to Orders and Pizzas inserts that can then be passed to the OrderItems and PizzaToppings insert queries automatically**
6. **Added delete for Pizzas. This will cascade correctly without causing anomalies in PizzaToppings**
7. **Changed update query for Orders to allow driver to be updated. A driver is optional and can be set NULL on creation, but it can now be updated to NULL as well.**

Feedback actions not taken:

1. Status of order. We are choosing not to implement a status attribute where the customer can track the progress of the order. This attribute would need to be updated manually by the employees of the restaurant, who would prefer to focus on cooking the pizza. In addition, any manually updated status will be prone to errors and could lead to customer dissatisfaction were it not updated correctly.

2. **Feedback on SELECT utilizing a search/filter with a dynamically populated list of properties. There are quite a few dynamically populated dropdowns for selecting filter parameters on the UI and in the queries that should count for this.**

Changes made to draft:

1. ERD updated with ‘toppings’ entity. This change effectively makes ‘pizzas’ an interface table between ‘toppings’ and ‘orderItems.’ What is a pizza if not a specific collection of toppings?
2. Added ‘int’ datatype to the orderItems.quantity attribute.
3. Create an intersection table PizzaToppings between Pizza and Toppings in order to properly normalize. This removes the duplicate ‘topping1, topping2, …’ from the pizza table to get it to 1NF.
4. Add an auto-increment primary key to OrderItems to bring it to 2NF.
5. Add not null constraint to a number of attributes to prevent creating entities without necessary data.

Plan:

Paula’s Pizza Palace needs a database to track their sales, their pizzas, and the specifics of each order. They want to track who ordered the pizza, who delivered it, what kind of pizza they ordered, and the total sale price. Customers should be allowed to add themselves to the database without a purchase to receive updates on sales or upcoming specialty pies. The business started slow but is now churning out dozens of pizzas per day. As a small and local business they only make a profit of around $1000 a week, but it’s enough to pay the bills.

We will need the following entities to enable this:

- Customers: This will track the people who order from the restaurant or who sign up for an account via the website even if they don’t place an order. Customers will have a 1:N relationship with Orders with CustomerID used as a foreign key. This relationship is optional as customers can sign up for an account without needing to order a pizza.
    - customerID, int, auto_increment, unique, not NULL, PK
    - name, varchar, not NULL
    - address, varchar
    - phoneNumber, varchar
    - emailAddress, varchar

- Drivers: The people who deliver the pizzas. It captures some detail about the drivers and the car they use to deliver the pizzas. Drivers will have a 1:N relationship with orders. Each order will have only one driver, but drivers can deliver many orders. ‘DriverID’ will be used as a foreign key in ‘Orders’.
    - driverID, int, auto_increment, unique, not NULL, PK
    - name, varchar, not NULL
    - phoneNumber, varchar, not NULL
    - carModel, varchar, not NULL

    .
- Orders: A record of all the orders. This entity will capture the pizza(s) on the order (multiple are allowed), the customer who ordered it, and the driver who delivered it. This entity has a M:N relationship with pizzas.
    - orderNum, int, auto_increment, unique, not NULL, PK
    - customerID, int, not NULL, FK
    - driverID, int, not NULL, FK
    - price, decimal, not NULL
    - date, varchar, not NULL
    - cancelled, boolean, not NULL, default 0

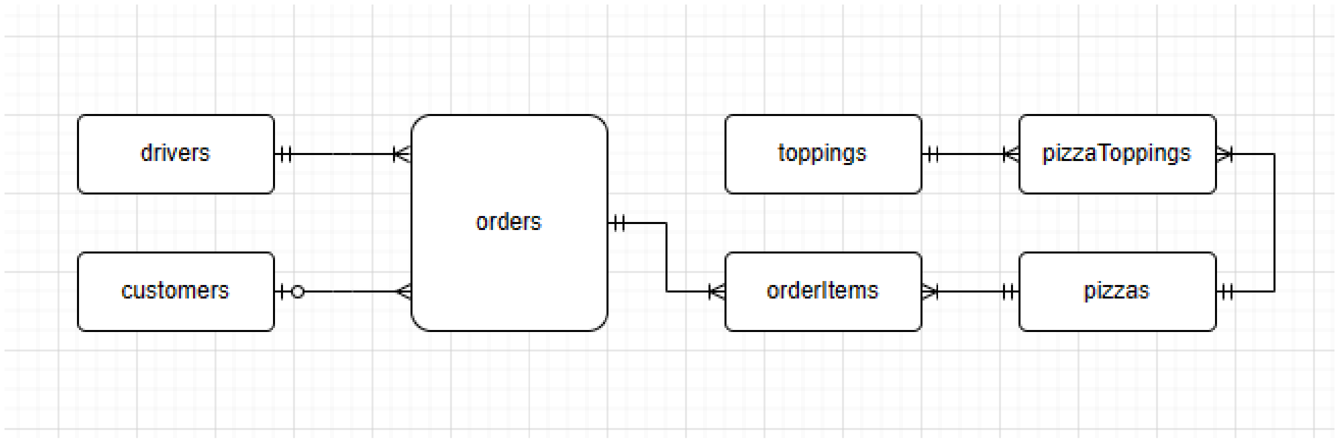
- OrderItems. This entity will hold details of a specific order and acts to resolve M:N relationships between orders, and pizzas. It will use driverID and pizzaID as foreign keys and has a 1:N relationship with both orders and pizzas.
    - orderItemID, int, auto_increment, unique, not NULL, PK
    - orderNum, int, not NULL, FK
    - pizzaID, int, not NULL, FK
    - quantity, int, not NULL

- Pizzas: The pizzas. This entity will capture the name of each pizza, a price, and description, as well as the toppings used to make the pizza. This entity will have a M:N relationship with orders as well as an M:N relationship with toppings.
    - pizzaID, int, auto_increment, unique, not NULL, PK
    - name, varchar, not NULL
    - price, decimal, not NULL
    - description, varchar

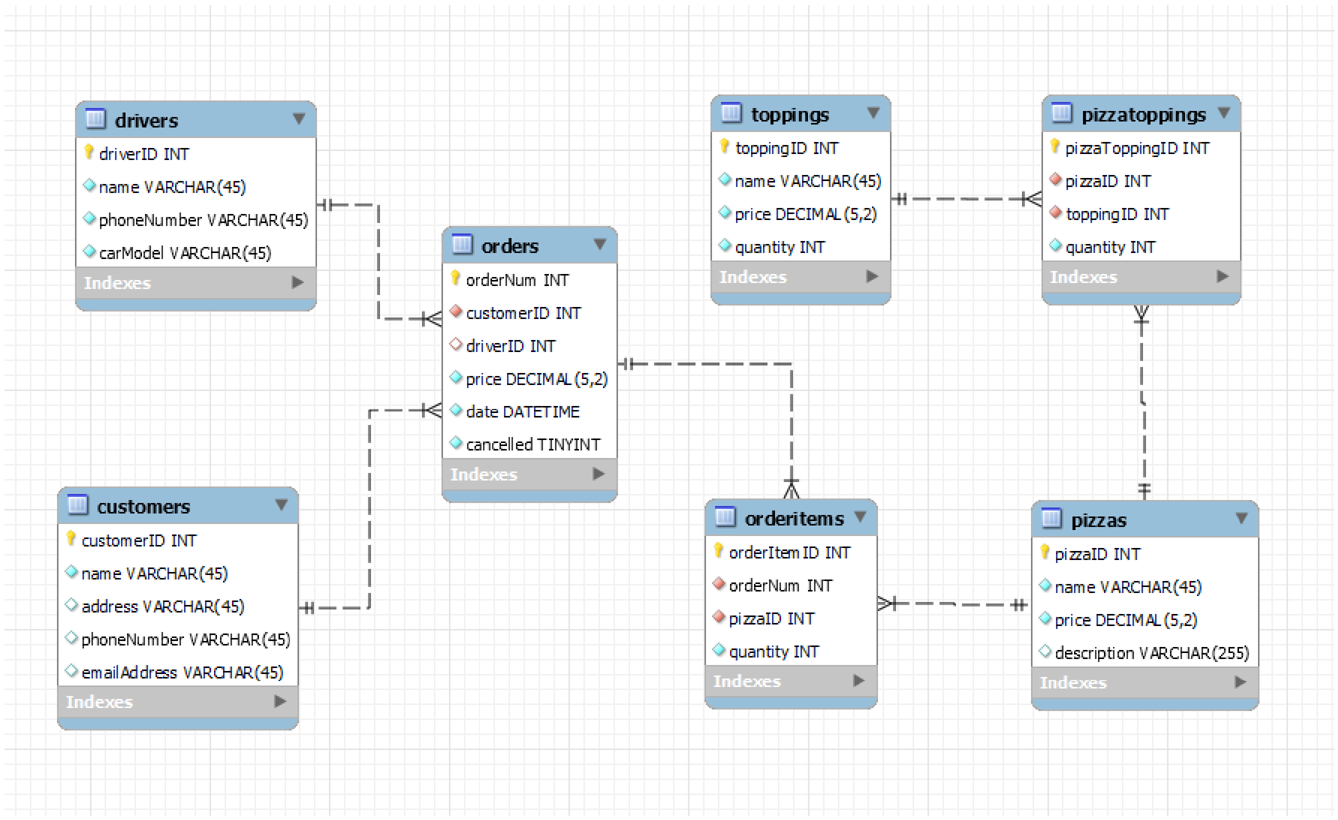
- Toppings: The available toppings. This entity serves as a category table for ‘pizzas’ and will track the inventory of each topping. It has a M:N relationship with ‘Pizzas’.
    - toppingID, int, auto_increment, unique, not NULL, PK
    - name, varchar, not NULL
    - price: decimal, not NULL
    - quantity, int, not NULL

- PizzaToppings: This entity will hold details of which toppings to use for a specific pizza and acts to resolve M:N relationships between toppings, and pizzas. It will use toppingID and pizzaID as foreign keys and has a 1:N relationship with both toppings and pizzas.
    - pizzaToppingID, int, auto_increment, unique, not NULL, PK
    - pizzaID, int, not NULL, FK
    - toppingID, int, not NULL, FK
    - quantity, int, not NULL

ERD:

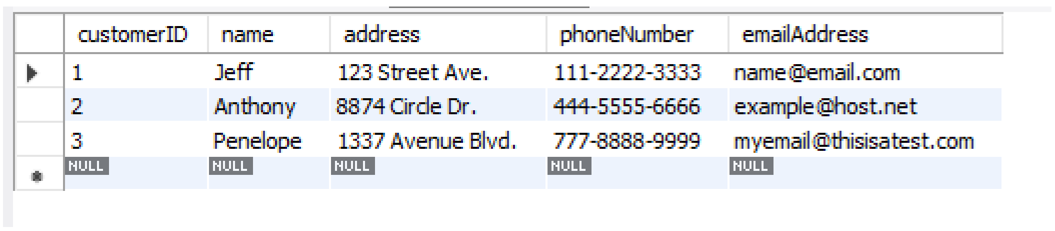

drivers
customers
orders
toppings
pizzaToppings
orderItems
pizzas

Schema:


drivers
driverID INT
name VARCHAR(45)
phoneNumber VARCHAR(45)
carModel VARCHAR(45)
Indexes
orders
orderNum INT
customerID INT
driverID INT
price DECIMAL(5,2)
date DATETIME
cancelled TINYINT
Indexes
toppings
toppingID INT
name VARCHAR(45)
price DECIMAL(5,2)
quantity INT
Indexes
pizzatoppings
pizzaToppingID INT
pizzaID INT
toppingID INT
quantity INT
Indexes
customers
customerID INT
name VARCHAR(45)
address VARCHAR(45)
phoneNumber VARCHAR(45)
emailAddress VARCHAR(45)
Indexes
orderitems
orderItemID INT
orderNum INT
pizzaID INT
quantity INT
Indexes
pizzas
pizzaID INT
name VARCHAR(45)
price DECIMAL(5,2)
description VARCHAR(255)
Indexes

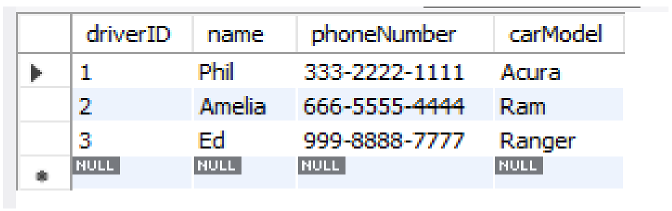
Example Data:

- customers:



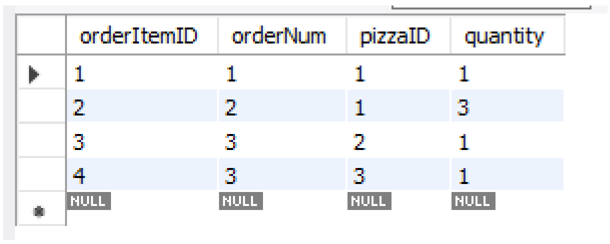
customerID	name	address	phoneNumber	emailAddress
1	Jeff	123 Street Ave.	111-2222-3333	name@email.com
2	Anthony	8874 Circle Dr.	444-5555-6666	example@host.net
3	Penelope	1337 Avenue Blvd.	777-8888-9999	myemail@thisisatest.com
NULL	NULL	NULL	NULL	NULL

- drivers:



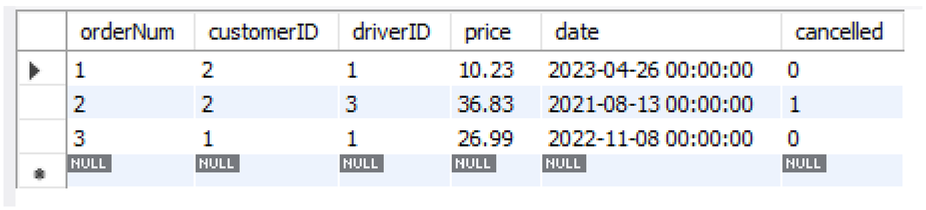
driverID	name	phoneNumber	carModel
1	Phil	333-2222-1111	Acura
2	Amelia	666-5555-4444	Ram
3	Ed	999-8888-7777	Ranger
NULL	NULL	NULL	NULL

- orderItems:



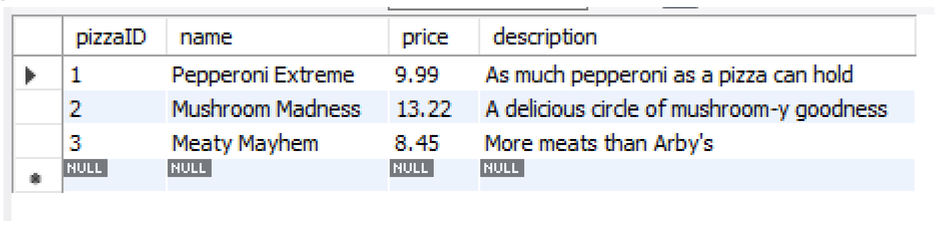
orderItemID	orderNum	pizzaID	quantity
1	1	1	1
2	2	1	3
3	3	2	1
4	3	3	1
NULL	NULL	NULL	NULL

- orders:



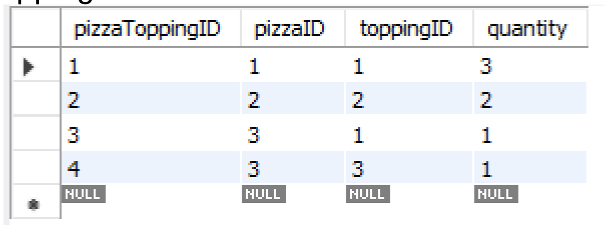
orderNum	customerID	driverID	price	date	cancelled
1	2	1	10.23	2023-04-26 00:00:00	0
2	2	3	36.83	2021-08-13 00:00:00	1
3	1	1	26.99	2022-11-08 00:00:00	0
NULL	NULL	NULL	NULL	NULL	NULL

- pizzas:



pizzaID	name	price	description
1	Pepperoni Extreme	9.99	As much pepperoni as a pizza can hold
2	Mushroom Madness	13.22	A delicious circle of mushroom-y goodness
3	Meaty Mayhem	8.45	More meats than Arby's
NULL	NULL	NULL	NULL

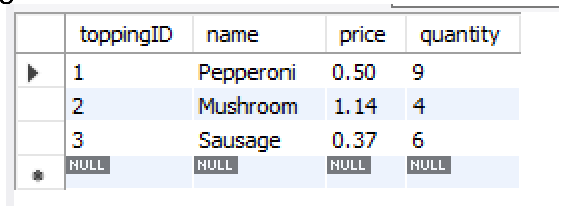
- pizzaToppings:



pizzaToppingID	pizzaID	toppingID	quantity
1	1	1	3
2	2	2	2
3	3	1	1
4	3	3	1
NULL	NULL	NULL	NULL

o

- toppings:



	toppingID	name	price	quantity
▶	1	Pepperoni	0.50	9
	2	Mushroom	1.14	4
	3	Sausage	0.37	6
*	NULL	NULL	NULL	NULL