Paula's Pizza Palace
Nick Herman
Brian Walsh

Feedback:
This is the feedback received on the Ed Discussion board, copy and pasted to this file. There were 4 reviews of this plan and all 4 are included:

#1:

Hi Group 124,

- **Does the overview describe what problem is to be solved by a website with DB back end?**

Yes, the overview has mentioned a database to track sales, pizzas, and order specifics, including who ordered and delivered the pizza, the pizza type, and the total sale price.

- **Does the overview list specific facts?**

Yes, it mentions the business's profit, it makes around $1000 a week and has been growing in sales to dozens of pizzas per day.

- **Are at least four entities described and does each one represent a single idea to be stored as a list?**

Yes, I think there are four entities described, "Customers," "Drivers," "Orders," and "Pizzas," each representing a single entity idea in this database design.

- **Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints, and describe relationships between entities?**

Yes, details are clearly shown in the bullet points. In addition, each entity is described with its purpose, attributes, data types, and relationships.

- **Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?**

There are several 1:M relationships formulated in this database, between "Customers" and "Orders," and "Drivers" and "Orders." A M:M relationship is also present between "Orders" and "Pizzas" through the "orderItems" intersection table. Yes, the ERD presents a sufficient logical view of the database to understand its relationships.

- **Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?**

The naming between overview, entity, attributes, singular, etc is consistent in my opinion. The use of capitalization and camel case are great.

I think my only suggestion for improvement is the ERD, I suggest adding more information to the entity tables. This can make the primary/foreign keys and attributes clear so the relationships will become easier to inspect when implementing the diagram. Overall, I like this project, well done!

#2:
Nice job on the assignment.

For the Orders entity, instead of a varchar for Date use a datetime data type to have both the date and time the order was placed. This way you can see peak hours and possibly busier times of the month/season.

The orders entity could have an attribute labeled status where you can see the progress of making the pizza all the way to when the order is dropped off and complete.

#3:

- Does the overview describe what problem is to be solved by a website with DB back end?
    - Yes
- Does the overview list specific facts?
    - Yes
- Are at least four entities described and does each one represent a single idea to be stored a s a list?
    - Yes
- Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?
    - Yes
- Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?
    - Yes
- Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
    - Yes

I think your outline and ERD are consistent with one another, and every relationship is described clearly and logically sound. Using your current outline just with some minor changes as Mark suggested above is fine, but it may be interesting to think about the following questions to make your app even more practical:

- What if customers pick up their order instead? Since orders.driverID is a FK, it would make sense to add "customer pickup" as a driver in your drivers entity.
- You mentioned your pizza can have up to 5 toppings, so would it make sense to create a toppings table with a M:N relationship with pizza? This way, you may be able to track the most popular toppings and stock up accordingly. Otherwise, perhaps just limiting the pizza types to 5 total would eliminate this need.
- Consider that an order was cancelled; you may want to add a "valid" attribute populated with boolean values in your orders entity so cancelled orders (or orders where the money was not collected for whatever reason) are not included in your sales calculation.
- If you were to add the "status" attribute to orders, you would also need to add a status entity with a 1:1 relationship to orders.

#4:

Does the overview describe what problem is to be solved by a website with DB back end?

- Yes, the overview mentions solving tracking of sales, pizzas, and order specifics through the use of a database. The overview also moves into the details of what would be tracked.

Does the overview list specific facts?

- Yes, the overview lists specific facts such as the store's profit of "$1000 a week" and that the store is a "small and local business" among other facts in the overview.

Are at least four entities described and does each one represent a single idea to be stored as a list?

- Yes, there are at least 4 entities. The 5 entities are Customers, Drivers, Orders, orderItems, and pizzas. Each entity has a representation of a single idea that is stored as a list.

Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?

- Each entity has a description that follows that lists the reason for its existence. Under that description of the entity are listed attributes with respective datatypes that have constraints. Some attributes need to have descriptions of what type it is. For example: quantity needs a type such as "INT". Relationships between entities are listed after the description and before the attributes. I would recommend listing the relationships underneath the entity like you have the attributes for formatting and to find these relationships easier.

Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?

- Yes the 1:M relationships are correctly formulated. For example, Drivers have a 1:M relationship with Orders. There is also at least one M:M relationship. Ex: Orders have a M:N relationship with pizzas. The ERD is drawn well with relationships between entities correctly drawn. I would also suggest writing in the attributes for each entity though.

Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

- **a**. There are no inconsistencies with naming ideas in the overview to the entity and their attributes.
- **b.** Entities are correctly named to be plural and attributes are correctly named to be singular. This is consistent.
- **c.** There are some inconsistencies here in attribute naming. Some start with a capital letter and others start with a lowercase letter. I would recommend making each entity start with a capital letter.

Aside from a few minor things that need to be fixed, this is a great proposal and a cool idea for a pizza shop! I also love the name! Nice job!

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.

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.

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.

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
    - address, charvar
    - 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
    - phoneNumber, varchar
    - carModel, varchar
    .
- 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, FK
    - driverID, FK
    - price, decimal
    - Date, varchar

- 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.
    - orderNum, PK
    - pizzaID, PK
    - quantity, int

- pizzas: The pizzas. Paula's is a minimalist pizza establishment and limits the number of toppings to 5. This entity will have a N:M relationship with orders.
    - pizzaID: int, auto_increment, unique, not NULL, PK
    - name, varchar
    - price, decimal
    - description, varchar
    - topping1, int, FK
    - topping2 … topping5, int, FK

- Toppings: The available toppings. This entity serves as a category table for 'pizzas' and will track the inventory of each topping. It has a 1:N relationship with 'pizzas'.
    - toppingID: int, auto_increment, unique, not NULL, PK
    - name, varchar
    - price: decimal
    - quantity: int

ERD: