a) Project Outline and Database Outline - Updated Version:

Project Outline

As the a college household in Corvallis with the issue of food waste and wasted space inside their fridge, Fridge Space is a database-driven website that allows a single household of people to view what they have in their fridge, add or remove grocery items, and give or restrict access to items for their roommates. In many households, groceries at home may be misused or forgotten about if they are not documented. This website should support a household of 1-10 people. An average household can expect to buy $100-300 worth of groceries which can lead to the household tracking 20 - 300 different items simultaneously. For each of these items, groceries can fit into 14 categories: dairy, meat, vegetables, fruits, beverages, snacks, condiments, leftovers, baked goods, breakfast items, baking ingredients, sauces, spices, and others. Lastly, roommates may use groceries for snacking and cooking; this database should track that activity seeing ~4 logs from each roommate each day (120 logs per month per roommate).

Fridge Space would like to track grocery items, grocery categories, owner information, and activities that happen between groceries and owners. Grocery items will be stored in an object table called Groceries and will store what it is, the expiration date, if it is still remaining, what category it belongs to, and who owns it. Grocery categories will be stored in an category table called Grocery_Categories which will store each unique category of food items that exist. Owner information will be stored in an object table called Owners which will store basic information such as their name and email address. Lastly, activities will be stored in an transaction table called Activity_Logs where it will store who is responsible for an action and what groceries they involved in that action.

Database Outline

<u>Groceries:</u>

Purpose: Records the individual grocery item that is being added to the fridge

Attributes:

- grocery_id: int, auto_increment, unique, not NULL, PK
- grocery_name: varchar(255), not NULL
- grocery_category: int, Not NULL, FK
- expiration_date: DATE, NULL
- remaining: tinyint(1), not NULL, DEFAULT Yes

Relationships:
- M:M with Owners because each grocery item must be owned by one or many owners, but each owner may own many or no grocery items
- M:1 with Grocery_Catagories because each grocery item must belong to one category, but a category can have many grocery items or none.
- M:M with Activity_Logs because a grocery item can be used in multiple actions, and an activity can use many groceries.

<u>Owners:</u>

Purpose: The person who purchased/owns groceries or is responsible for an action

Attributes:
- owner_id: int, auto_increment, unique, not NULL, PK
- fname: varchar(255), not NULL
- lname: varchar(255), not NULL
- email: varchar(255), not NULL;

Relationships:
- M:M with Groceries because each grocery item must be owned by one or many owners, but each owner may own many or no grocery items
- 1:M with Activity_Logs because an owner can have zero or many activities that they logged, but an activity log must correspond to only one owner

<u>Groceries_Owners (intersection table):</u>

Purpose: Acts as an intersection table for the M:M relationship between Owners and Groceries

Attributes:
- groceries_owners_id: int, auto_increment, not NULL, PK
- grocery_id: int, not NULL, FK ON DELETE CASCADE
- owner_id: int, not NULL, FK ON DELETE CASCADE

Relationship: (relationships explained in both Groceries and Owners)
- M:1 relationship between Groceries_Owners and Groceries
- M:1 relationship between Groceries_Owners and Owners

<u>Activity_Logs:</u>

Purpose: A record of logs about who used which item

Attributes:
- activity_id: int, unique, not NULL, auto_increment, PK
- activity_name: varchar(255), not NULL

- description: LONGTEXT, NULL
- owner_id: int, NULL, FK

Relationships:

- M:1 with Owners because an owner can have zero or many activities that they logged, but an activity log must correspond to only one owner
- M:M with Groceries because in an activity log, multiple groceries can be used to create a dish, and the grocery item can be split so it in in multiple logs

Activity_Logs_Groceries (intersection_table):

Purpose: Acts as an intersection table for the M:M relationship between Activity_Logs and Groceries

Attributes:

- activity_logs_groceries_id: int, unique, auto_increment, not NULL, PK
- activity_id: int, not NULL, FK ON DELETE CASCADE
- grocery_id: int, not NULL, FK ON DELETE CASCADE

Relationships: (relationships explained in both Groceries and Activity_Logs)

- M:1 relationship between Activity_Logs_Groceries and Activity_Logs
- M:1 relationship between Activity_Logs_Groceries and Groceries

Grocery_Categories:

Purpose: Record which of the food categories the newly entered grocery is. The categories are dairy, meat, vegetables, fruit, beverages, snacks, condiments, leftovers, baked goods, breakfast items, baking ingredients, sauces and spices, and others.

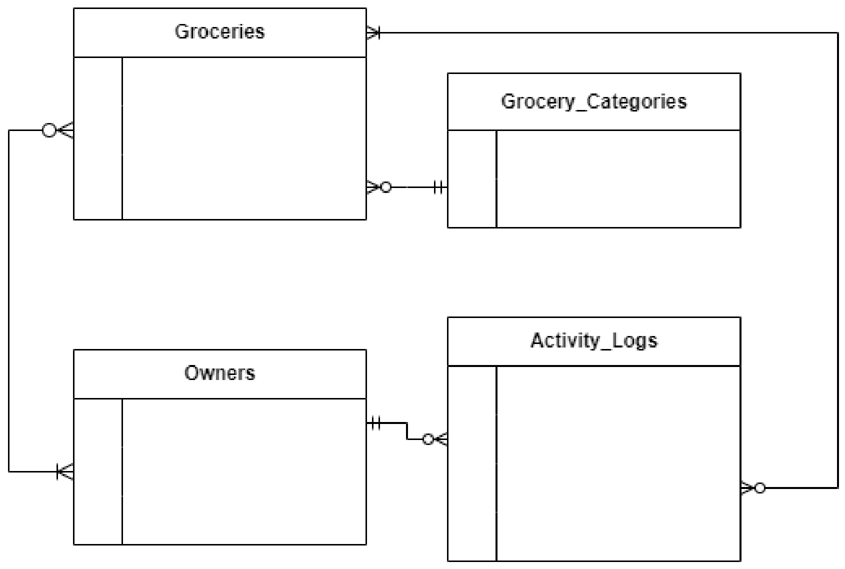
Attributes

- category_id (PK for type_of_groc): int, unique, auto_increment, not NULL, PK
- category_name: varchar(255)

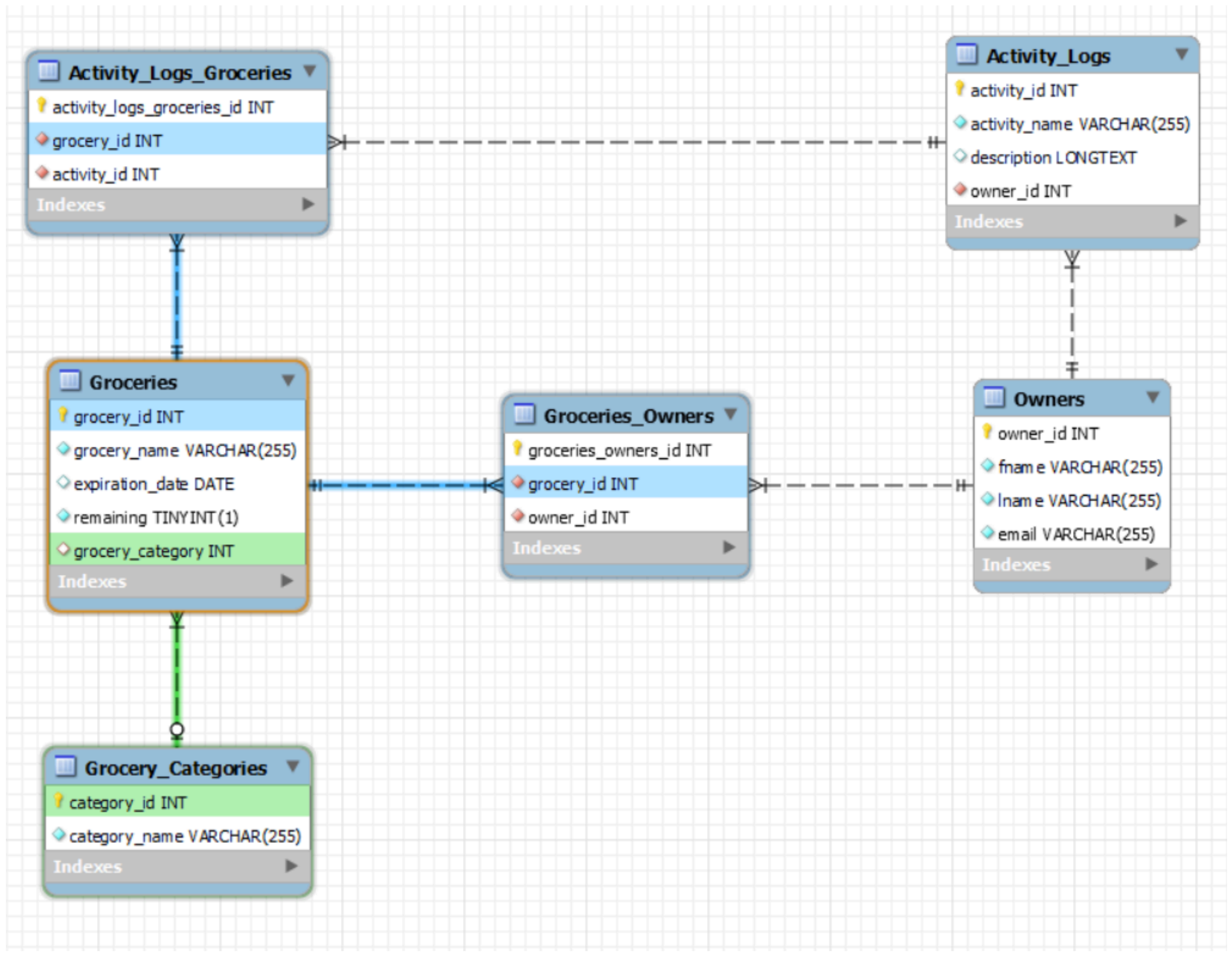
  Relationships:

  - 1:M with Groceries because each grocery item must belong to one category, but a category can have many grocery items or none.

c. Entity Relationships Diagram



d. Schema


Activity_Logs_Groceries
activity_logs_groceries_id INT
grocery_id INT
activity_id INT
Indexes
Activity_Logs
activity_id INT
activity_name VARCHAR(255)
description LONGTEXT
owner_id INT
Indexes
Groceries
grocery_id INT
grocery_name VARCHAR(255)
expiration_date DATE
remaining TINYINT(1)
grocery_category INT
Indexes
Groceries_Owners
groceries_owners_id INT
grocery_id INT
owner_id INT
Indexes
Owners
owner_id INT
fname VARCHAR(255)
lname VARCHAR(255)
email VARCHAR(255)
Indexes
Grocery_Categories
category_id INT
category_name VARCHAR(255)
Indexes

e. Example Data

Groceries

grocery_id	grocery_name	grocery_category	expiration_date	remaining
1	Rolled Oats	10	None	Yes
2	Greek Yogurt	1	2024-02-14	Yes
3	Almond Milk	10	2024-03-01	Yes
4	Honey	7	None	Yes

Grocery_Categories

category_id	category_name
1	Dairy
2	Meat
3	Vegetables
4	Fruits
5	Beverages
6	Snacks
7	Condiments
8	Leftovers
9	Baked Goods
10	Breakfast Items
11	Baking Ingredients
12	Sauces
13	Spices
14	Others

Owners

owner_id	fname	lname	email
1	Justin	Pham	phamjus@oregonstate.edu
2	Thuy Duyen	Doan	doant@oregonstate.edu
3	Bean	Dog	mydoglol2014@gmail.com

Activity_Logs

activity_id	activity_name	description	owner_id
1	Justin Added Groceries	Items Added: Rolled Oats, Greek Yogurt. Greek Yogurt for me (Justin) but Rolled Oats for Everybody	1
2	Duyen Added Groceries	Items Added: Honey. Everybody will be able to use it	2
3	Justin Used 4 Groceries	Made 4 jars of overnight oats for myself only. please do not eat	1

Activity_Logs_Groceries

activity_logs_groceries_id	grocery_id	activity_id
1	1	1
2	2	1
3	3	2
4	1	3
5	2	3
6	3	3
7	4	3

Groceries_Owners

groceries_owners_id	grocery_id	owner_id
1	1	2
2	1	1
3	1	3
4	2	1
5	3	1
6	3	2
7	3	3
8	4	1
9	4	2

B.1) Fixes based on Feedback from Step 1:

Actions based on the feedback

Common Criticisms

- Overview section
    - Many of the criticisms about the overview that it should have expanded the scope (Estevan Sanchez) or that the overview should be more exact about the purposes of the database (Kenneth Huddleston). The changes we made was to change “Fridge Space is a database-driven website that allows a single household of college roommates” to “Fridge Space is a database-driven website that allows a single household of people”, and to add the second paragraph which goes into more detail about the database.
- Intersection Tables
    - Many people critized the initial draft for using two FK attributes in their respective entities to model an M:M relationship (Estevan Sanchez, Argent Alija). While before we didn’t understand how intersection tables were implemented, We added details in the database overview which outlines how the intersection tables will work and what relationships that they interact with. This will help reduce data redundancy in our database.
- Naming
    - Multiple people brought up the problem with our minor inconsistency with naming either with being consistent with naming foreign keys the same as primary keys (Estevan Sanchez), changing the old attribute groceries_used (Matthew Menold), or changing Activity_log to be more specific (Matthew Menold). Firstly, We changed all foreign keys to match their primary keys from the initial draft. For the next advice, We weren’t able to apply it because we removed the attribute grocery_used entirely to form an intersection table. Lastly, we chose not to apply it because we felt that dividing it into separate entities would be too redundant; we believe that having just the entity name and description we can separate one activity from another.
- Relationships
    - Multiple people brought up some problems with relationships within ourdatabase (other than those who relate to intersection tables). Either

with reducing M:M relationships or avoiding creating M:M relationships that were cyclic (Estevan Sanchez) and a misrepresentation in our ER diagram (Kenneth Huddleston). Firstly, we chose to change the relationships between Owners and Activity to M:1 to address the problem Estevan Sanchez had. Although the relationships are still cyclic, it must be or else data that is essential to know (like owners) may become redundant in our database. Next, I changed the ER diagram to correct for the mistake for the relationship between categories and groceries (We made it M:1 when it should have been 1:M).

Upgrades to the Draft version (From Step 1)

- Removed Owner_Credientials
  - We felt like having a credentials entity deviated from the purpose of the database. While it is useful to always have credentials within the database to ensure that the correct owner is making actions they personally did, it was not in the scope of the project nor the scope of our proposed project.
- Removed Attributes from ER diagram
  - While attributes were never needed in the ER diagram, We removed them in this iteration of the draft as it added a lot of confusion on how each of the entities were connected. We could have added our intersection tables to the ER diagram if we wanted to still have attributes, but it added more complexity to something that should have been simple.
- Changed varchar(1000) -> varchar(255) in any attribute
  - Did this change based on the fact that it would be easier to store as it would be one byte.
- Changed int(14) in category_id
  - Did this change because of a personal confusion.
- Added auto_increment for category_id
  - Did this change to make it easier to insert values
- Made it optional for owners to own groceries
  - Intuitively makes sense as someone can own no groceries or many groceries, but if someone brings groceries in the house, atleast one person has to own them (the person who brings them in most likely).

B.2) Fixes based on Feedback from Step 2

Actions based on the feedback

Common Criticisms

- **Example Data doesn't match Inserted SQL Data**
  - This is a simple fix, we simply changed the example data values in the spreadsheet to match the SQL insert entries.
- **Schema naming differences**
  - There were naming differences in the Activity_Logs table in the schema and the database outline. For example, the title of the table didn't follow the naming scheme ("Activity_logs") and the id attribute name didn't follow the outline ("activity_log_id").
- **Error sourcing the SQL file**
  - Some reviewers had problems sourcing the SQL file we provided. To fix this, we changed the order tables were created to allow for the tables to reference a foreign key to be created.

Feedback by the peer reviewer

Justin Holley

- 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
- Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?
  - Yes - there are lists prepopulated in Groceries and a few others.
- 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).
    - I believe it does.
- 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.
    - While there are deletes, I didn't see one on the Grocery Owners intersection table. Perhaps consider adding one?
- 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
- 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 don't believe so. Consider updating some of your ON DELETE CASCADE Statements in the DDL to instead a SET Null and update the attribute (if needed) to allow for Null.
- 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.
    - Honestly, I think your webpage looks great and has a lot of the features I think the assignment asks for. You could consider reducing some functionality where not needed to make it a little less work but I applaud the design and effort. I think you'll just need to make some of the other corrections listed above and you're in great shape!

Clinton Merritt

Great job on the HTML pages Justin and Thuy. This is a good way to keep track of who owns what groceries, especially in a communal living situation.

- 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 every page of the UI utilizes SELECT. Each page is broken down into different tables.

- Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?
    - Yes the groceries page has the grocery category dropdown that auto populates.
- 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 Groceries, Owners, Activity logs and Grocery categories all have an iINSERT implemented in 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).
    - Yes, there is a M:M relationship between grocries and owners. When INSERTING into groceries, there is an option to pick the owner.
- 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 an option to delete an owner from the owner list, groceries and categories.
- 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 on the groceries table list.
- 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 the expiration date on an item on the groceries list allows for the field to be N/A
- 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 would suggest using aliases on the grocery owner intersection table so people don't have to remember what IDs belong to what owner or grocerie.

Hafsa Akyol

- 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, the UI utilizes a SELECT for every table in the schema. Each table in the schema is displayed in the UI.

- Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?
    - Yes, the grocery category and owners in the Groceries page are dynamically populated.
- 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, the UI implements an INSERT statement for Groceries, Activity Logs, and Grocery Categories.
- 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).
    - Yes, each INSERT also adds the corresponding FK attributes. When adding a grocery and owner, these details are also added to the Grocery Owners intersection table.
- 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, Groceries and Owners have a M:M relationship. When a grocery or owner is deleted, it is deleted from the intersection table Grocery Owners.
- 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, you can update Groceries (id, name, category, owners, expiration date).
- 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, in Activity Logs, you can choose to add Groceries or not. I don't understand how the relationship between Groceries and Activity Logs can be optional as you need groceries to carry out an activity, but I might be mistaken.
- 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 really like the simple and easy-to-understand design. Looks great overall! A small suggestion would be to fix the "THanks for looking at our project so far!" to "Thanks..." :)

Regina Sanchez

- 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, the UI does utilize a select for every table in the schema.

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

Looking at your UI and dml file, it looks like on the activity log there is a dynamically populated list with owner names. Also, on the grocery list page of your UI, there is a drop down with populated data for grocery category, and update grocery where a user can select an item listed + change the category.

- 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.

From what I saw on the UI and dml, it appears that there is an insert for every 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).

Yes, on the dml file I see that there is an insert for the MM relationship of Groceries and Owners.

- 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.

For the delete, on the dml file, I see it for groceries, owners, and activity log. As for the M:M relationship, I think some cascade deleting could be helpful. I see it for the individual items but on the file can't seem to see where that issue is dealt with.

- 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, for the Groceries there is an update visible in the dml file and the ui website. It allows for a grocery to be updated with a new category, owner, expiration date.

- 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.

On the grocery attribute the date can be null, so thats an attribute relationship that can be nullable. However, for entities with other entities I do not see a nullable relationship.

- 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 thing I noticed is that on the home page it says "THanks".... and think thats a typo? Also, on the activity logs page I see ID, Activity name, description, and owner ID. If I am understanding correctly, the owner ID should come from the Owners entity so should that be an int value? Right now it says that Owner ID is dairy? Great project!!

B.3) Fixes based on Feedback (from Step 3)

Actions based on the feedback

List briefly the actions that you chose to take based on the above feedback. If you decided not to act on a specific suggestion, you need to describe in detail your reasoning.

- While there are deletes, I didn't see one on the Grocery Owners intersection table. Perhaps consider adding one?
    - We did not include a delete function in the ‘Grocery Owner’ intersection table because when an entity from either the Groceries or Owners, it should automatically delete that row in the intersection table
- I don't believe so. Consider updating some of your ON DELETE CASCADE Statements in the DDL to instead a SET Null and update the attribute (if needed) to allow for Null.
    - There should be one nullable relationship between Groceries and Category (formerly), however, we did change the nullable relationship to be between Activity_Log and Owners.

- You could consider reducing some functionality where not needed to make it a little less work
    - This feedback gave no specific examples of what functions they thought were unnecessary. To us, we kept it simple already. Users can add, update, and remove the necessary data. Other than that, we do not have any other functions that require the user to act on.

- I would suggest using aliases on the grocery owner intersection table so people don't have to remember what IDs belong to what owner or groceries.
    - We took this advice and assigned aliases. Now, when users view the intersection tables, they will see the owner's, groceries, or activity name instead of the number

- I don't understand how the relationship between Groceries and Activity Logs can be optional as you need groceries to carry out an activity, but I might be mistaken
    - We added that because we saw on the project requirement that we need a foreign key to be nullable. After reviewing our diagram, we decided that instead of’ groceries’ in ‘activity logs’ to be nullable, it would be ‘owner.’

Upgrades to the Draft version

If you are making any changes to the files based on your own changed design decisions, they should be listed under this section.

- Since we have some variables that are nullable, we added an optional disclaimer in the ‘create grocery’ for the expiration date. This will allow the user to identify that it is nullable too easily

At aliases into the intersections ID so user do not need to remember what each number references to