Project Group #1: Team Oaklaura

Grant Wu and Jessica Ramirez

Project Step 5 - Draft

URL to Oaklaura Bike Co-Op UI:

http://classwork.engr.oregonstate.edu:7018/

1. Feedback by TAs and Peer Reviewers

Step 4: TA Review - Madelyn Lazar

"Great work team 1! Your DDL imports and your database is normalized.

Something to note: In future steps, we will check to make sure you use subqueries for your INSERT statements rather than hardcoding FK values. Make sure to implement this for the next step in your DDL."

Step 4: Peer Review 2 - Jacob Hopkins

"*Do all SELECT operations work as expected? In other words, is data from all the database tables displayed in the web application? If not, please describe which tables a data is not being displayed from.*
Each of the tables in your report SELECT data from the database appropriately. Items disappear from a table upon deletion.

Is a RESET button/link implemented that properly resets the database? If not, describe the behavior/what data is not being reset.
The RESET button works appropriately. After deleting several items, I pressed the reset button and they were repopulated.

For any additional CRUD operations that the team listed as operational, do they function as the team expects (e.g. if the team stated that a CRUD step worked, but you found an error, please tell them)?
The delete functionality for both the Repair Reports and Contacts tables functioned as intended.

Would a user easily be able to use the UI to complete the CRUD operations that the team listed as working? If not, please elaborate and provide helpful suggestions for how the UI can be improved.
Yes. The interface is understandable. The pop up form when clicking the edit button is a nice touch. These icons will of course need to be added to the pages without CRUD functions yet.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed, helpful feedback will receive higher credit. If the team is not blocked or is having difficulty, encouraging and supportive comments would be a better response than NO feedback.
I think that you've made a lot of progress where you were blocked. My advice would be to repeat your format from the two pages that delete functionality is working on. It was very user friendly.

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).
All work is my own without the help of AI."

Step 4: Peer Review 3 - Drew Schlabach

"*Do all SELECT operations work as expected? In other words, is data from all the database tables displayed in the web application? If not, please describe which tables a data is not being displayed from.*
Yes, each table displays the appropriate sample data.

Is a RESET button/link implemented that properly resets the database? If not, describe the behavior/what data is not being reset.
Yes. I deleted a repair record, and pressing the "Reset Database" button restored the database to its original state.

For any additional CRUD operations that the team listed as operational, do they function as the team expects (e.g. if the team stated that a CRUD step worked, but you found an error, please tell them)?
All CRUD steps which have been listed as operational work as expected. The team reports that deleting rows is operational. I can confirm that every table which offers the ability to delete a row is able to do so without problems.

Would a user easily be able to use the UI to complete the CRUD operations that the team listed as working? If not, please elaborate and provide helpful suggestions for how the UI can be improved.
Yes, the UI is very impressive and intuitive, and goes above and beyond the project requirements. Deleting is done through a simple button press, and editing brings up a pop-up form. Creating a new row is as simple as opening a form with the button at the bottom of the table. One thing I would suggest is to consider putting the Add button at the bottom of tables with the CREATE feature at the top of the table. While it is intuitive to put it at the bottom since that is where the new row will be added, it may present a problem if the table has many rows and requires the user to scroll to the bottom to add a new row. Perhaps organizing the table from most recent to oldest would fix this.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed, helpful feedback will receive higher credit. If the team is not blocked or is having difficulty, encouraging and supportive comments would be a better response than NO feedback.
Overall, I am extremely impressed with the progress that you have made on your project. The website is slick and intuitive, and every CRUD feature which you have implemented works correctly. I also love that you included such a detailed homepage. It is helpful to know what framework you are using as that helps me to better understand how your website functions. One suggestion that I have is to fix the centering for the navigation bar and the tables. In terms of the blockages that you mention when following the module examples: I have had some similar problems, but it seems like you guys figured it out. Great job!

Best regards,
Drew Schlabach

This review is entirely original, and AI was not used in any capacity to write it."

Step 4: Peer Review 3 - Ethan Van Hao

"*Do all SELECT operations work as expected? In other words, is data from all the database tables displayed in the web application? If not, please describe which tables a data is not being displayed from.*
Yes, all the data from all the database tables are displayed

Is a RESET button/link implemented that properly resets the database? If not, describe the behavior/what data is not being reset.
Yes, the RESET button properly resets the database.

For any additional CRUD operations that the team listed as operational, do they function as the team expects (e.g. if the team stated that a CRUD step worked, but you found an error, please tell them)?
Yes, the delete/trash icons work as expected.

Would a user easily be able to use the UI to complete the CRUD operations that the team listed as working? If not, please elaborate and provide helpful suggestions for how the UI can be improved.
Yes, the UI is amazing and looks wonderful.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed, helpful feedback will receive higher credit. If the team is not blocked or is having difficulty, encouraging and supportive comments would be a better response than NO feedback.
Great work on the UI and the database. Everything looks great and really reflects how much work was put in. I'm excited to see the finished product as this looks to be the most refined project I have seen so far. Great work! The only thing I can say is that the bike icons in your title are different and is a little off-putting.

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).
All of the above is my work"

Step 4: Peer Review 4 - Noah Pragin

"*Do all SELECT operations work as expected? In other words, is data from all the database tables displayed in the web application? If not, please describe which tables a data is not being displayed from.*
Yes.

Is a RESET button/link implemented that properly resets the database? If not, describe the behavior/what data is not being reset.
Yes.

For any additional CRUD operations that the team listed as operational, do they function as the team expects (e.g. if the team stated that a CRUD step worked, but you found an error, please tell them)?
Yes, good work making sure intersection tables and relationships stay updated!

Would a user easily be able to use the UI to complete the CRUD operations that the team listed as working? If not, please elaborate and provide helpful suggestions for how the UI can be improved.
Yes, great UI!

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed, helpful feedback will receive higher credit. If the team is not blocked or is having difficulty, encouraging and supportive comments would be a better response than NO feedback.
Good work perservering through your challenges! Your website and database look excellent, way to go! The only note I have for you is check the padding in your navbar, the right padding for the Store Personnel button is not the same as the others.

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).
All my work"

Step 3: TA Review - Madelyn Lazar

"Team 1, this is very impressive! Both your DDL and DML had all the criteria needed and I can tell you put a lot of work into your UI pages already. Very excited to see your next steps!."

Step 3: Peer Review 1 - Scott Dispensa

"*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, all the tables have SELECTs.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.
Yes.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?
Yes, you can delete from several tables.

Does the UI have at least one DELETE that will 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, you can delete a Repair Record.

Is there at least one UPDATE form in the UI 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, Repair Record and Contacts have UPDATE options.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?
Yes, for Repair Record.

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.
Great work! The UI looks amazing, and everything seems to work well. The only suggestion I'd make would be to replace some of the IDs with the associated names for clarity. For example, in Customers it might be nice to replace the Contact ID with the name and the boolean with Yes or No in case the person using the database isn't familiar with that notation. Also in Sales Reports, maybe replace PersonnelID and CustomerID with the names. Other than that, it's a very good project!!!

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).
This is all my work."

Step 3: Peer Review 2 - Paula Tica

"*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.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.
Yes, the UI implements INSERT for two tables in the schema. There is an Add a Contact form for Contacts and a Create a Repair Record form for Repair Reports.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?
Yes, the UI implements DELETE buttons for Contacts and Repair Reports.

Does the UI have at least one DELETE that will 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, the UI has DELETE for Repair Reports. This table facilitates the M:M relationship between Bikes and Store Personnel.

Is there at least one UPDATE form in the UI 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 are UPDATE forms for Contacts and Repair Reports.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?
Yes, there is an UPDATE form for Repair Reports.

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.
My suggestion is to use AS aliases for the column names (e.g. dateReceived changed to Date Received).
All of the above is my original work."

Step 3: Peer Review 3 - Fidella Wu

"*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 SELECT for every table to display the data.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.
Yes, there is an insert form for Contacts and Repair Reports.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?
Yes, there is a delete button for Contacts and Repair Reports.

Does the UI have at least one DELETE that will 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 button for Repair Reports, which is the intersection table for the M:M relationship.

Is there at least one UPDATE form in the UI 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 form for Contacts and Repair Reports.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?
Yes, there is an update form for Repair Reports.

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.
In the edit repair report form, None is showing for the bike input, even though some rows have a bikeID. In the bike dropdown, there is currently only 3 showing, and I would suggest keeping the date format consistent.

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).
All my work."

Step 3: Peer Review 4 - Daniel Guardado

"*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 uses select queries to get database information to fill tables in different pages.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.
Yes, the repair reports page has functionality for creating new rows/records.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?
Yes, repair reports and contacts have delete buttons in the UI.

Does the UI have at least one DELETE that will 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 button in the repair reports page.

Is there at least one UPDATE form in the UI 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, the contacts page has a pencil icon for editing a record.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?
Yes, the repair reports page has an update form for editing a record.

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 would be to use aliases for the column/attribute names.

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).
All my work."

Step 2: TA Review - Madelyn Lazar

"Great work team 1! Your DDL imports and your database is normalized.

Something to note: In future steps, we will check to make sure you use subqueries for your INSERT statements rather than hardcoding FK values. Make sure to implement this for the next step in your DDL."

Step 2: Peer Review 1 - Joseph Gilmore

"*Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?*
Yes, the schema acts as a more detailed and implemented version of the ERD. All keys and naming match between the two.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
Both the ERD and Schema are properly set up with all of naming being uniform (plural entities and singular attributes, capitalized naming of entities and lowercase attributes).

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?
Yes, the schema is very easy to read and understand.

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?
Yes, intersection tables (and optional relationships) are properly formed. The foreign keys in these intersection tables are also clearly and uniformly created.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?
No, it does not appear that there are any abnormalities. All of the sample data covers each entity and attribute and inserted data appears to properly reference keys (both foreign and primary).

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)
Yes, I was able to load this into a database and test it. All of the table creations along with data insertions are working.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?
Yes, all data is in accordance with the specified data types. This group chose to use a lot of ENUM's which seems to be a great decision as there are certain categories that will be very consistent. For example, they status attribute in the Bikes table only allows for the following: 'In Repair', 'In Review', 'For Sale', 'Sold'. This will help keep the database clean of duplicated data.

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?
Yes, all primary and foreign keys are declared in the SQL file and match that of the schema. Each table has a corresponding CASCADE operation.

In the SQL, are relationship tables present when compared to the ERD/Schema?
Yes, relationship tables exist in the SQL similar to that of the schema. Specifically, this would be the SalesReport table as it utilizes foreign keys from both the Customers and Bikes table.

In the SQL, is all example data shown in the PDF INSERTED?
Yes, all of the data in the screenshots is also inserted in the DDL.sql file.

Is the SQL well-structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?
Yes, all of the data is hand authored, well structured and commented. All of the sections are divided in to clean sections with similar data.

All of the above work is my own; no AI has been used."

Step 2: Peer Review 2 - Daniel Guardado

"*Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?*
Yes, the schema follows the database outline and ER logical diagram. The schema clearly depicts all tables, attributes, and datatypes as shown in the outline.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
Naming is almost consistent. Entity table names are capitalized, follow pascal case, and are plural. Attributes use camel case and are singular. Two attributes that could be changed would be bikes_bikeID and customers_customerID as they follow snake case.

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?
The schema is easy to read as relationship lines are not interfering with others. Everything is readable.

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?
Yes, the intersection tables are properly formed, showing a primary key and two foreign keys with proper relationship notations used.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?
No non-normalized issues appear. No composite keys are used, and no non-prime attributes seem to depend on one another.

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)
The SQL file is syntactically correct as no issues occur when trying to import to phpMyAdmin.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?
Yes, the data types are appropriate for the attributes described in the outline.

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?
Yes, primary and foreign keys are used correctly in the schema and outline. Cascade operations are declared correctly for foreign keys used in intersection tables or other master tables.

In the SQL, are relationship tables present when compared to the ERD/Schema?
Yes, 1:1, 1:M, and M:M relationships are presented correctly in the Schema and EDR.

In the SQL, is all example data shown in the PDF INSERTED?
Yes, all example data shown in the pdf was also present in the SQL file.

Is the SQL well-structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?
Yes, the DDL.sql file is well documented showing comments about the authors, the tables, and the example data.

All my work."

Step 2: Peer Review 3 - Tyler Grzymalski

"*Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?*
Yes.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
There is a consistency in naming throughout the project. Entities and attributes are plural and singular. Capitalization is used appropriately.

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?
The schema is very well done, it is formatted perfectly.

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?
The intersection tables that are used are used properly.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?
I cannot find any issues regarding normalization.

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)
The SQL file is proper. It ran without issues for me.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?
Looking through the attributes, there are no instances where I would suggest changing the data type.

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?
Every PK and FK is defined appropriately based on the Schema. CASCADE operations are correctly utilized.

In the SQL, are relationship tables present when compared to the ERD/Schema?
Relationship tables are defined as they should be when compared to the ERD.

In the SQL, is all example data shown in the PDF INSERTED?
ALl data used in the SQL is used in the PDF.

Is the SQL well-structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?
The structure of the SQL is perfect, comments allow the reader to skim quickly and understand what is happening.

This review is all my own work."

Step 1: TA Review - Madelyn Lazar

"This is great work team 1! Your naming conventions were consistent throughout, you meet all the requirements for an M:M relationship + enough entity tables, and you clearly explained how each of your relationships will work in your outline. I also love how well formatted and easy to read your outline is!

Some notes:

-You lost points for not including numerical data in your overview. I left some specific questions that might help you when deciding what to include/answer in your overview within the rubric.

-For future steps, you will be required to have at least one nullable FK. So I recommend you start thinking about which table/FK this requirement could apply to in your data structure."

Step 1: Peer Review 1 - Tyler Grzymalski

"Your overview does a great job of explaining how this database would help streamline repair efforts by helping to ensure no aspect of repair gets overlooked and increasing accountability between repairs and volunteers that work on them.

There are several facts in the overview that directly connect to the database solution. Limited funding explain why the sales invoices cannot contain more than one bike, there is a detailed explanation on how the repair process goes and it aligns with the structure of the database, handwritten notecard's were getting unruly which gives the reasoning for why a database needs to be implemented.

There are five entities described and they all align with course material. The bikes table does a good job of covering important consumer information regarding the brand and type of bike, and it elaborates on the current condition of the bike. The store personnel table and customers table accurately track standard personal information. Perhaps there is a way to combine them by adding a customer role to the StorePersonnel Table? The repair reports table keeps track of the repair process on bikes. The sales report table does a good job of outlining a traditional invoice.

The relationships are correctly formulated and make sense. There is a M:M relationship between StorePersonnel and Bikes since multiple volunteers typically work on one bike. There is consistency in naming: entities are plural and attributes are singular. Overall, great start to the project!

This review is all original work."

Step 1: Peer Review 2 - Andrew Mathena

“Hi Team,

Great work on the draft here. I appreciated the unique and well-thought out use case.

The overview does a good job of describing the problem to be solved by the website and database back end. It sounds like the financial constraints and volunteer-powered staff necessitate the use of a system to track updates on work across shifts. The system replaces a manual and likely error-prone note card system. So it sounds like a great use case for a database powered application which can help the organization drive revenue.

The specific facts around mechanical aptitude of the staff, the workforce, and the revenue stream of the organization help illustrate the scope and scale of the solution. Some additional numerical facts which could add further clarity include the total workforce count, the total number of annual orders or revenue, and maybe even a comparison between revenue and the cost of a POS system to help illustrate why that’s not a viable option.

Three tables in your design appear to describe objects: StorePersonnel, Customers, and Bikes. These tables include fields for information I would believe to be cricial to support the described use case. I was especially interested by your use of the enum field types for a number of fields, such as bike color; this has made me reconsider where I can use enum types to refine & simplify my own group’s database design. RepairReports and Salesreports appear to be intersection/reporting tables which support the M:N requirements of the project design and contain information critical to track repair progress and sales.

The outline of entity details describes the purpose and attributes of each: Bikes includes color and styling info, as well as repair completion status. StorePersonnel includes contact info for staff. Likewise, Customers includes contact info for customers. The report tables include attributes to connect out to the staff and customer tables, and critical date information.

1:M relationships are correctly formulated. As mentioned above, RepairReports and Salesreports appear to be intersection/reporting tables which support the M:N requirements of the project design. All information in the ERD represents the data accurately, assuming that a customer must have at least one sale in order to be represented in the system (as another reviewer pointed out).

Naming consistency between entities & attributes is good. Pluralization is correct for all items, and camel case is used as a standard convention in the system.

Also kudos on the beautiful LaTex formatting!

All content here was my original work.

Best,

Andrew”

Step 1: Peer Review 3 - Drew Schlabach

"Grant and Jessica,

Before I get to the technical details of my review, I want to first commend you for the excellent formatting of your draft. It is easy to read, visually appealing, and extremely organized. Great job! Moving on…

The information provided in your overview does a fantastic job of justifying your database design. Between the basic POS, the involvement of many volunteers per bike, and the co-op's intriguing business model, a database appears to offer improvement in plenty of areas. A well-made database will allow the co-op to more efficiently track sales, and perhaps by storing repair logs will allow bikes to be completed more quickly as well.

At least four entities are described in your outline (Bikes, StorePersonnel, Customers, RepairReports, SalesReports), and each one does in fact represent a single idea to be stored as a list. If I am not mistaken, the Bikes, StorePersonnel, and Customers entities will function as objects, and the RepairReports and SalesReports entities will function as transactions. Additionally, your inclusion of a fifth entity despite the project's minimum of four is in line with the same trend I have seen in other groups and shows a desire to exceed expectations.

The outline of each entity includes sufficient details about its purpose, and each attribute is given an appropriate data type and constraints (not NULL for everything, auto_increment for PKs, and various categorical constraints for such attributes as color, style, and status). One detail that I really appreciate is the constraint of UNIQUE for the email attribute in the Customers entity. That's not something that I would have thought of myself, but it makes perfect sense considering that many organizations limit only one account per email address. Additionally, you have provided for each entity a thorough summary of its relationship to other entities.

The 1:M relationships appear to be correctly formatted, and I notice that while each StorePersonnel can be tied to zero or more SalesReports, each Customer must be tied to at least one SalesReport. I may be wrong, but I believe that this would result in an insertion anomaly if the co-op wishes to store information about a potential customer who has not yet made a purchase. Your M:M relationship between StorePersonnel and Bikes is cleverly implemented with foreign keys in both RepairReports and SalesReports, and your accompanying ERD does a great job of visualizing your outline.

There is consistency in all required areas: the entity and attribute names match between the outline and ERD, the entities are plural and the attributes are singular, and the entities use UpperCamelCase while the attributes use lowerCamelCase.

I enjoyed reading your Step 1 draft, and I wish you the best of luck in developing your project.

Best regards,

Drew Schlabach

This review is entirely original, and AI was not used in any capacity to write it."

2. Actions Based on the Feedback

Step 4 –> Step 5

Based on external feedback we received on Project Step 4 Draft, the following actions were taken:

- **Suggestion from NP:** As suggested, we fixed the padding on the navbar, to make it more aesthetically pleasing.
- **Suggestions from EH:** As suggested, we changed the bike icons in the navbar to be the same.
- **Suggestions from DS:** As suggested, moved the create buttons to the top of the table, and fixed the centering of the navbar and tables.
- **Suggestions from JH:** It was suggested that we should add CUD functionality to all tables, but we chose not to implement this suggestion to avoid over-complicating our project. We are currently already implementing CUD functionality for two entities, which surpasses the project requirements.

Step 3 –> Step 4

Based on external feedback we received on Project Step 3 Draft, the following actions were taken:

- **Suggestion from PT/DG:** As suggested, we utilized aliases for column names being displayed in tables.
- **Suggestions from SD:** As suggested, we utilized included names with IDs associate with Customers or Store Personnel. We also replaced boolean 0's and 1's with "Yes" or "No" for the receiveNewsletter attribute within the Customers table.
- **Suggestions from FW:** As suggested, we fixed the bug in the update form for Repair Reports, where an incomplete bike set was being displayed. We also changed the date formats to be consistent amongst the tables and forms

Step 2 –> Step 3

Based on external feedback we received on Project Step 2 Draft, the following actions were taken:

- **Suggestion from ML:** As suggested, in our DDL.sql file we replaced all hardcoded foreign keys within INSERT statements with subqueries.
- **Suggestion from DG:** To be consistent in our naming we updated the FKs bikes_bikesID to bikesID and customer_customerID to customerID.

Step 1 –> Step 2

Based on external feedback we received on Project Step 1 Draft, the following actions were taken:

- **Suggestion from ML/AM:** Numerical facts have been added to the overview to provide context and establish the scope of the database.
- **Suggestion from ML:** bikeID in RepairReports was changed to a nullable FK, to facilitate documentation of non-bike specific work, such as cleaning and fixing stock parts such as cassettes, wheels and derailleurs.
- **Suggestion from TG:** It was suggested that we combine the StorePersonnel and Customers entities. While combining customers and store personnel into the same entity is an interesting idea, we think differentiating between them is advantageous in that: a) it distinguishes between service providers and service recipients, b) it allows flexibility to add special attributes to these entities later in the future (skills for StorePersonnel, payment info for Customers), and c) Customer info may require additional privacy/security considerations compared to StorePersonnel. For these reasons, we decided to keep the separation between Customers and StorePersonnel as initially proposed.
- **Suggestion from DS:** An insertion anomaly was identified in the previous formulation of the Customer-SalesReports relationship, where each Customer corresponded to one-to-many SalesReports. This has been altered here such that now each Customer corresponds to zero-to-many SalesReports. This change now allows for a Customer to be created without necessitating a linked SalesReport, to handle situations where we may want to acquire information from a potential Customer.

3. Upgrades from Previous Versions

Step 4 –> Step 5

The UI was was connected to the backend:

- **Create Functionality:** Create functionality has been fully implemented for RepairReports and Contacts.
- **Update Functionality:** Update functionality has been fully implemented for RepairReports and Contacts.
- **Data Validation:** Code was deployed to verify phone number and email data is in the correct format when entered.

Step 3 –> Step 4

The UI was further developed:

- **Delete Implemented:** Delete functionality was implemented for RepairReports and Contacts.
- **Reset Implemented:** A reset button was added to the navigation bar to allow for a rest of the database to its original state.

A PL.sql file has been created to store the PL-SQL statements called upon by the Database UI.

Step 2 –> Step 3

A UI for the database was created, featuring:

- **Read Operations:** A separate page/table is displayed for each entity.
- **Create Operations:** Create forms allow for insertion of new records into the RepairReports and Contacts tables.
- **Update Operations:** Update forms allow for the alteration of records from the RepairReports and Contacts tables.
- **Delete Operations:** Delete buttons allow for deletion of records from the RepairReports and Contacts tables.

DDL.sql and DML.sql files have been created to store the SQL statements used to define and manipulate the database.

Step 1 –> Step 2

Based on internal conversations, the following actions were made:

- **Removed personnelID as FK within SalesReports:** To avoid documenting an additional M:M relationship between StorePersonnel and Customers we removed the tracking of the StorePersonnel from the SalesReports entity. We do not consider this to be important information to track and removal of it greatly simplifies the database design.
- **Modified ERD:** The ERD was modified such that the relationship between StorePersonnel and SalesReports was removed, as discussed above.
- **Reclassified relationship between Customers and SalesReports:** We have reclassified the relationship between Customers and SaleReports to be non-identifying since they do not share a primary key. We believe we misunderstood this concept before and erroneously classified the relationship as an identifying one.

Application of Normalization principles to our database design led to the following changes:

- **Addition of new Contacts Entity:** We identified a situation that introduces data redundancy: if a StorePersonnel (employee or volunteer) decided to purchase a bike from the co-op, their contact information would have to be re-entered as a new record in the Customers table (and vice versa if a Customer became an Employee or Volunteer). This means that the contact information of that person may exist in two places, and any updates to that contact information would create an update anomaly—requiring their information to be updated in more than one location. To alleviate this issue we introduced an additional entity, Contacts, which stores the contact information of a single person. Each Customer and SalesPersonnel is linked to exactly zero or one Contacts entity. The addition of a Contacts table ensures our database follows 3rd Normal Form (3NF) by eliminating transitive dependencies. Instead of storing contact information separately in both the Customers and StorePersonnel tables, we now store it in a single Contacts table. Each Customer and StorePersonnel entry references a contactID, ensuring that all contact details depend only on a single key. This design avoids redundancy, prevents update anomalies, and maintains data integrity.
- **Addition of receiveNewsletter attribute in Customers Entity:** In addition to the rationale provided for keeping Customers and StorePersonnel separate, above in Section 2, we added an additional Customer specific attribute—receiveNewsletter—to further delineate between the two entities. The attribute is a tinyint, representing a boolean value denoting whether a customer is subscribed to the newsletter (1) or not (0, default).
- **Removal of isCompleted attribute from Bikes Entity:** This attribute was originally used to store a boolean status of whether a bike was repaired and ready to be sold yet. We realized this was redundant information that depended on the status attribute, also within the Bikes entity. If the status attribute is 'For Sale', then isCompleted would always be 1 (true). This violates the principles of normalization, and so we removed it as an attribute.

4. The Oaklaura Bike Cooperative

The Oaklaura Bike Cooperative is a non-profit organization that accepts donations of old or broken bicycles, refurbishes them, and then sells them at an affordable price. The co-op sees 50-100 bicycles enter and exit their doors on a monthly basis. Due to limited funding, the co-op operates with a small team of 10 employees and relies heavily on volunteers to assist with bicycle repairs. At the current time there are about 100 volunteers that donate their time, and this is expected to grow over the course of the next five years. The co-op's limited funding also requires they use a basic POS system that can only process one bike sale at a time, setting a limit of one bike per sales order. The co-op brings in on average between $10,000 and $25,000 per month, and most of this revenue is used to support the small team of dedicated employees as well as to provide community engagement events.

Store personnel consist of both volunteers and employees. Most volunteers are not experienced bike mechanics, so they often can't fully repair a bike during the few hours the co-op is open for volunteer work. To maintain continuity and organization, volunteers are expected to repair what they can during their shift and document their progress in a report the same day the work was performed. This allows the next volunteer and/or employee to review the logs and continue the work where the previous one left off. Sometimes volunteers engage in non-bike specific work, such as cleaning and fixing stock parts such as cassettes, wheels and derailleurs, which gets documented in these repair reports as well. Once a volunteer believes a bike is fully repaired, a trained employee inspects it to ensure it meets safety standards before placing it on the sales floor. Once there, a bike can be sold, a transaction which is tracked through a sales report.

Historically, the co-op tracked repair progress and sales using handwritten notecards stored in a filing cabinet. However, as the organization grows, this system has become increasingly difficult to manage. Implementing a database would be an ideal solution for organizing and sharing information between volunteers and employees about the status of each bicycle, as well as for tracking the inventory and sales history of the ever expanding co-op.

5. Database Schema

Bikes Table

Contains details on a particular bike that resides within the co-op.

- **bikeID [PK]:** int, not NULL, auto_increment
- **color:** enum('Black', 'White', 'Red', 'Blue', 'Green', 'Pink', 'Purple', 'Yellow', 'Orange', 'Silver', 'Other'), not NULL
- **style:** enum('Mountain', 'Road', 'Fat', 'Hybrid', 'Enduro', 'BMX', 'Cruiser', 'Kids', 'Electric'), not NULL
- **brand:** varchar(45), not NULL
- **status:** enum('In Repair', 'In Review', 'For Sale', 'Sold'), not NULL
- **dateReceived:** date, not NULL

Relationships:

- M:M relationship between Bikes and StorePersonnel is implemented with bikeID and personnelID as FK's within both RepairReports and within SalesReports.
- 1:1 relationship between Bikes and SalesReports is implemented by bikeID as a FK within SalesReports. *Note: due to our outdated POS system, only one bike can be sold at a time (i.e. only one Bike can appear on each SalesReport).*
- 1:M relationship between Bikes and RepairReports is implemented with bikeID as a FK within RepairReports.

StorePersonnel Table

Holds information on store employees and volunteers that work within the co-op.

- **personnelID [PK]:** int, not NULL, auto_increment
- **contactID [FK - Contacts]:** int, not NULL
- **role:** enum('Employee', 'Volunteer'), not NULL

Relationships:

- M:M relationship between StorePersonnel and Bikes is implemented with bikeID and personnelID as FK's within both RepairReports and within SalesReports.
- 1:M relationship between StorePersonnel and RepairReports is implemented with personnelID as a FK within RepairReports.
- 1:1 relationship between StorePersonnel and Contacts is implemented with contactID as a FK within StorePersonnel.

Customers Table

Holds information relating to existing and potential customers.

- **customerID [PK]:** int, not NULL, auto_increment
- **contactID [FK - Contacts]:** int, not NULL
- **receiveNewsletter:** tinyint, not NULL, DEFAULT = 0 (false)

Relationships:

- 1:M relationship between Customers and SalesReports is implemented with customerID as a FK inside of SalesReports.
- 1:1 relationship between Customers and Contacts is implemented with contactID as a FK within Customers.

RepairReports Table

Holds repair information performed on a particular bikes (Bikes_StorePersonnel Intersection Table that includes additional repair information).

- **repairID [PK]:** int, not NULL, auto_increment
- **personnelID [FK - StorePersonnel]:** int, not NULL
- **bikeID [FK - Bikes]:** int
- **dateRepaired:** datetime, not NULL
- **hoursSpent:** decimal(4,2), not NULL
- **description:** varchar(255), not NULL

Relationships:

- 1:M relationship between RepairReports and StorePersonnel is implemented with personnelID as a FK inside RepairReports.
- 1:M relationship between RepairReports and Bikes is implemented with bikeID as a FK inside RepairReports.

SalesReports Table

Holds information pertaining to the sale of a particular bike.

- **salesID [PK]:** int, not NULL, auto_increment
- **bikeID [FK - Bikes]:** int, not NULL, unique
- **customerID [FK - Customers]:** int, not NULL
- **dateSold:** date, not NULL
- **price:** decimal(5,2), not NULL

Relationships:

- 1:1 relationship between Bikes and SalesReports is implemented by bikeID as a FK within SalesReports. *Note: due to our outdated POS system, only one bike can be sold at a time (i.e. only one Bike can appear on each SalesReport).*
- 1:M relationship between SalesReports and Customers is implemented with customerID as a FK within SalesReports.

Contacts Table

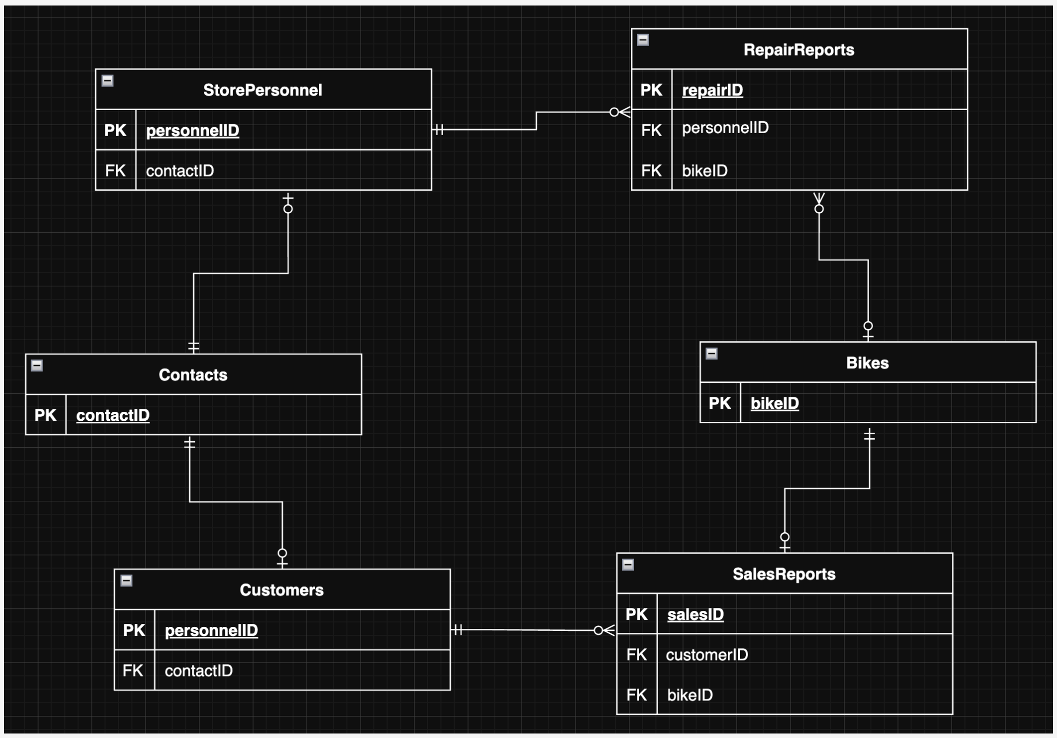
Holds contact information for any person in the system (both service providers and service recipients).

- **contactID [PK]:** int, not NULL, auto_increment
- **firstName:** varchar(45) not NULL
- **lastName:** varchar(45) not NULL
- **phone:** varchar(45), not NULL
- **email:** varchar(100), not NULL, UNIQUE

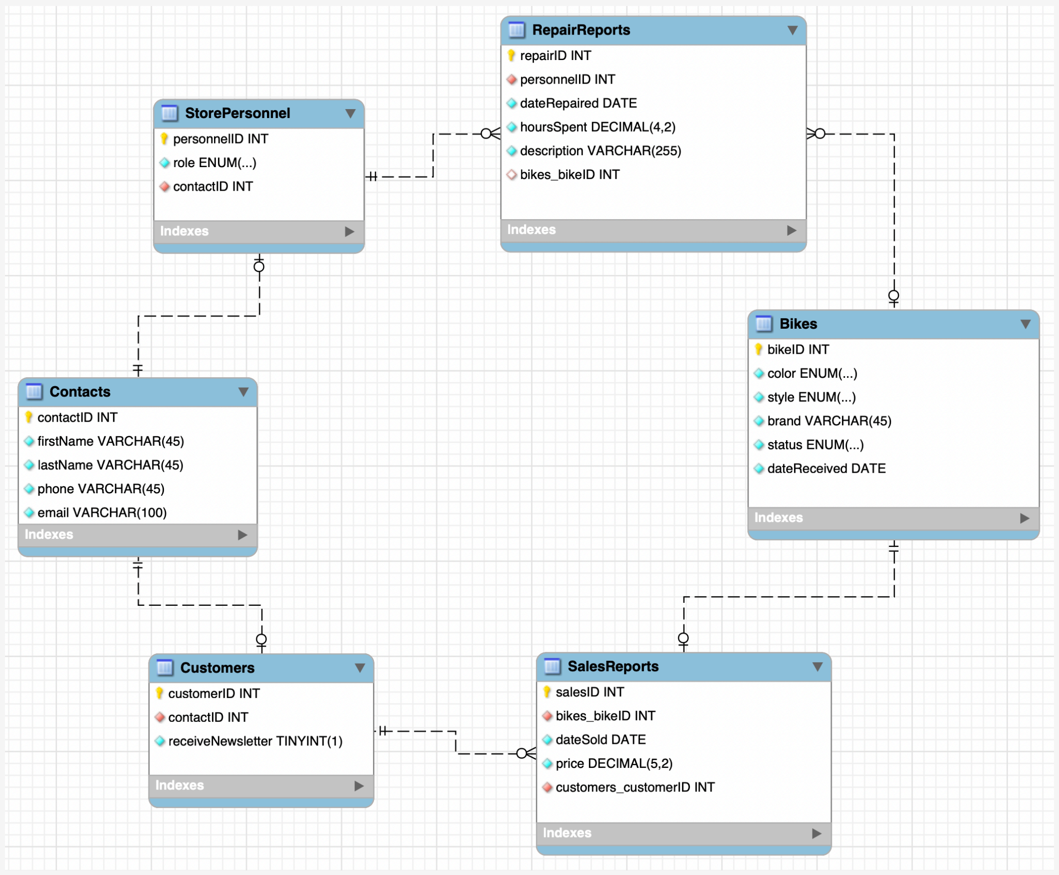
Relationships:

- 1:1 relationship between Contacts and StorePersonnel is implemented with contactID as a FK within Customers.
- 1:1 relationship between Contacts and Customers is implemented with contactID as a FK within Customers.

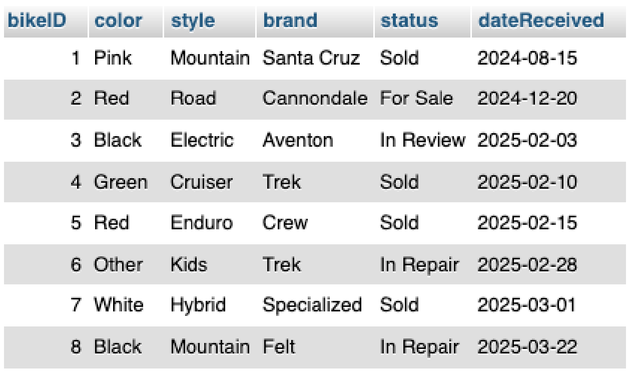
6. Entity-Relationship Diagram



7. Schema Diagram

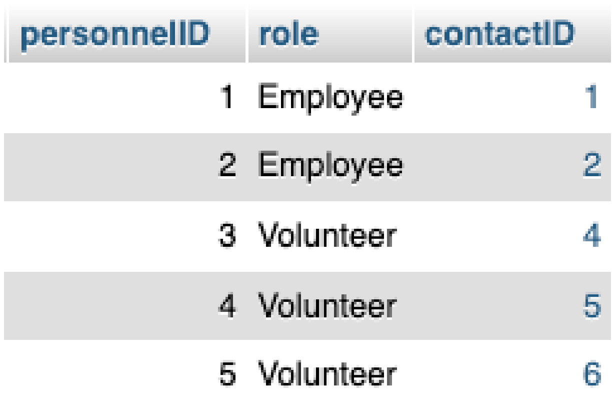


8. Sample Data



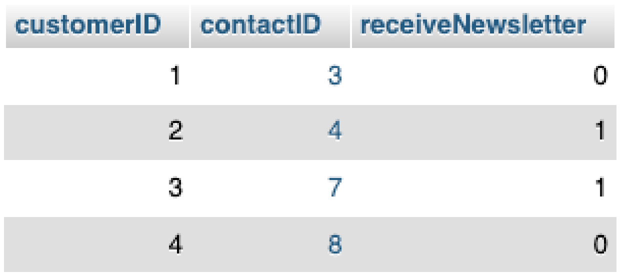
bikeID	color	style	brand	status	dateReceived
1	Pink	Mountain	Santa Cruz	Sold	2024-08-15
2	Red	Road	Cannondale	For Sale	2024-12-20
3	Black	Electric	Aventon	In Review	2025-02-03
4	Green	Cruiser	Trek	Sold	2025-02-10
5	Red	Enduro	Crew	Sold	2025-02-15
6	Other	Kids	Trek	In Repair	2025-02-28
7	White	Hybrid	Specialized	Sold	2025-03-01
8	Black	Mountain	Felt	In Repair	2025-03-22

Figure 1: Sample data inserted into the Bikes table



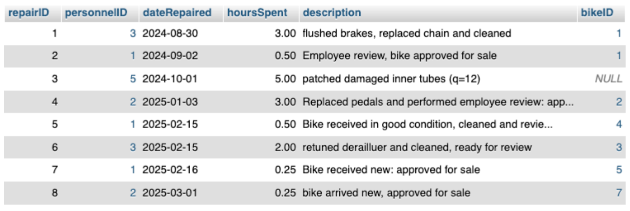
personnelID	role	contactID
1	Employee	1
2	Employee	2
3	Volunteer	4
4	Volunteer	5
5	Volunteer	6

Figure 2: Sample data inserted into the StorePersonnel table



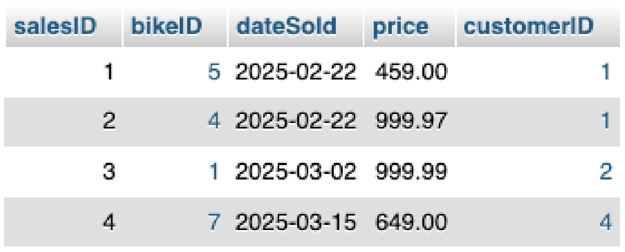
customerID	contactID	receiveNewsletter
1	3	0
2	4	1
3	7	1
4	8	0

Figure 3: Sample data inserted into the Customers table



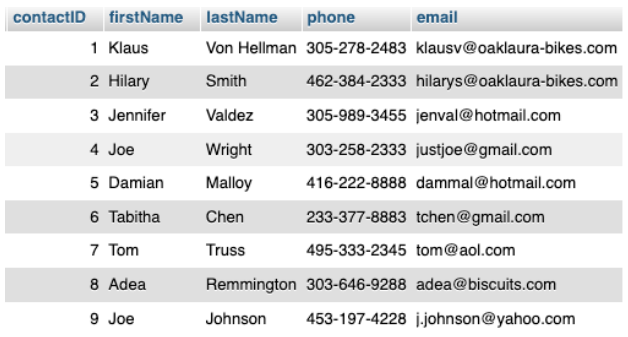
repairID	personnelID	dateRepaired	hoursSpent	description	bikeID
1	3	2024-08-30	3.00	flushed brakes, replaced chain and cleaned	1
2	1	2024-09-02	0.50	Employee review, bike approved for sale	1
3	5	2024-10-01	5.00	patched damaged inner tubes (q=12)	NULL
4	2	2025-01-03	3.00	Replaced pedals and performed employee review: app...	2
5	1	2025-02-15	0.50	Bike received in good condition, cleaned and revie...	4
6	3	2025-02-15	2.00	retuned derailluer and cleaned, ready for review	3
7	1	2025-02-16	0.25	Bike received new: approved for sale	5
8	2	2025-03-01	0.25	bike arrived new, approved for sale	7

Figure 4: Sample data inserted into the RepairReports table



salesID	bikeID	dateSold	price	customerID
1	5	2025-02-22	459.00	1
2	4	2025-02-22	999.97	1
3	1	2025-03-02	999.99	2
4	7	2025-03-15	649.00	4

Figure 5: Sample data inserted into the SalesReports table



contactID	firstName	lastName	phone	email
1	Klaus	Von Hellman	305-278-2483	klausv@oaklaura-bikes.com
2	Hilary	Smith	462-384-2333	hilarys@oaklaura-bikes.com
3	Jennifer	Valdez	305-989-3455	jenval@hotmail.com
4	Joe	Wright	303-258-2333	justjoe@gmail.com
5	Damian	Malloy	416-222-8888	dammal@hotmail.com
6	Tabitha	Chen	233-377-8883	tchen@gmail.com
7	Tom	Truss	495-333-2345	tom@aol.com
8	Adea	Remmington	303-646-9288	adea@biscuits.com
9	Joe	Johnson	453-197-4228	j.johnson@yahoo.com

Figure 6: Sample data inserted into the Contacts table

9. Citations

- Inspiration for the Bike Co-Op came from The Recyclery, a non-profit bike shop based out of Chicago, IL (last retrieved on 4/9/2025): https://www.therecyclery.org/
- MySQL workbench was used to create the ERD diagram shown above.
- The LaTeX template used here was adapted from the Cleese-Assignment template v.2.0 (retrieved on 4/2/2025): https://latextemplates.com/template/cleese-assignment
- TeXShop was used for all LaTeX related compilations.
- All database design work is original.