Project 3: Draft Version
Team 102
Team Members:
Yupheng V Xiong
Leo Zimmer

Topic Name #1 (Rare Car Imports)

Web Page URL: https://web.engr.oregonstate.edu/~xiongyup/group-project/index.html

Feedback/Reviews:

Joon Noh

Does the overview describe what problem is to be solved by a website with DB back end?
If yes, summarize. If not, what changes would better support describing the problem to be solved?

The overview describes the problem of needing a database website for a rare car imports system to help consumers place orders and record transaction details. Some possible changes I could suggest are diving a bit deeper into the problem by stating approximately the number of consumers that would use this system or an estimate of how frequent and how many orders will be made. I would also take another look at capitalization of certain words like countries.

Does the overview list specific facts?
If yes, summarize what the facts illustrate about the proposed DB solution. If not, what facts would better support illustrating the scope and scale of the proposed DB solution?

The overview lists a few examples of specific countries that the cars could be imported from/to. As mentioned above, stating the number of consumers that would use this system or an estimate of how frequent and how many orders will be made could be useful information to have in the overview.

Are at least four entities described, and does each one represent a single idea to be stored as a list?
If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

Yes, the outline lists four entities: Customer, Transaction, Cars, and Location, which are all relevant to the database system.

Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints, and describe relationships between entities?
If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

The entities are missing a description of their purpose. The attributes all have relevant datatypes and constraints, if applicable. I believe some attributes such as customerID and carID in Transaction should be labeled as foreign key FK instead of primary key PK. The outline also describes relationships between the entities. I would think the Location entity should have some relationship with another entity.

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

If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

The 1:M relationships such as Customers to Transactions are correct. I don't see at least one M:M relationship so I suggest looking into including that. The ERD does seem to match the entity relationships described in the outline.

Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

Yes, there is consistency in the naming of entities and attributes. All four entities begin with a capital letter while their respective attributes begin with a lowercase letter. Attributes use camel case convention as well. My suggestion would be that all the entities are plural. For example, "Customers" instead of "Customer" and "Transactions" instead of "Transaction."

Maddox Nehls

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

Yes, the overview effectively outlines the primary objective of the proposed system. It clearly identifies the need to facilitate the importing of cars from various regions (e.g., Japan, Europe, United States) to designated destinations.

Summary: Aims to develop a database-backed website that assists consumers in ordering and importing cars from specified origins to destinations. The system will handle essential transaction details, ensuring a smooth and organized importation process.

2. Specific Facts Listed

Does the overview list specific facts?

Yes, the overview includes specific details that define the scope and functionality of the proposed database solution. Key facts such as the types of data to be collected (names, prices, dates) and the generation of transaction receipts provide a clear understanding of the system's requirements. Also, the mention of importing cars from various regions shows the need for managing diverse data sets and ensuring accurate transaction records.

Summary: The project focuses on importing cars from multiple regions, handling data related to customer information, transaction details, car specifics, and locations.

3. Entities Described

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

Yes, the draft identifies four primary entities, each representing a distinct concept necessary for the system:

- **Customer**: Captures individual customer details.
- **Transaction**: Records information about each car importation transaction.
- **Cars**: Stores details about the cars available for import.
- **Location**: Manages information regarding the origin and destination points for car imports.

Summary: Each entity (Customer, Transaction, Cars, and Location) encapsulates a single, coherent idea, showing the need for organized data storage and retrieval within the database.

4. Outline of Entity Details

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

Partially. The draft does provides a basic structure of each entity with attributes, however, there are areas that need further elaboration:

- **Purpose Description**: The purpose of each entity is briefly mentioned but could benefit from a more detailed explanation to clarify how each contributes to the overall system.
- **Attribute Details**: Attributes are listed with data types and constraints; however, some inconsistencies and redundancies are present (e.g., carPrice appears in both Transaction and Cars entities).
- **Relationships**: Relationships are described, but the draft lacks major clarity in some areas, like the composite primary key in the Transaction entity and the absence of relationships for the Location entity.

Summary: It provides a foundational structure but needs more detailed descriptions of each entity's purpose, clarification of attribute usage, and relationship definitions to fully meet requirements.

5. Relationship Formulation and ERD Logic

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

1:M Relationships:

- **Customer to Transaction**: Correctly Identified as a 1:M relationship, where one customer can have multiple transactions.
- **Car to Transaction**: Identified as a 1:M relationship, implying that each car can be involved in multiple transactions. However, this may need reconsideration depending on business rules (e.g., if a car can be imported multiple times).

M:M Relationships:

- Draft does not explicitly identify any M:M relationships. If, for example, a car can be associated with multiple transactions and a transaction can involve multiple cars, an M:M relationship would be needed, potentially requiring an intersection table.

ERD Logic:

- ERD will have to drastically change after the above changes are made. Also, the naming is inconsistent between the draft of the outline and the diagram itself. Capitalization is also incorrect.

6. Naming Consistency and Conventions

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

a) Naming Consistency:

- The names used in the overview align well with those in the entity definitions (e.g., Customer, Transaction, Cars, Location).

b) Pluralization:

- Entities are inconsistently pluralized. For instance, Cars is plural, while Customer and Transaction are singular. Maintaining consistent pluralization for entity names would help with clarity.

c) Capitalization:

- Attribute names use a mix of camelCase and snake_case (e.g., customerID, brandCar). Using a consistent naming convention, such as snake_case or camelCase uniformly, would enhance readability and maintain professionalism.

Summary: There is a general alignment between the overview and entity names; however, inconsistencies in pluralization and capitalization should be addressed to ensure uniformity and clarity throughout the database design.

Suggestions for Improvement

- **Clarify Relationships**:
  - Re-evaluate the 1:M relationships to ensure they align with the business logic.
  - Identify and define any necessary M:M relationships, potentially introducing intersection tables where appropriate.
- **Standardize Naming Conventions**:
  - Decide on a consistent pluralization approach for entity names.
  - Use uniform capitalization (e.g., all attributes in snake_case or camelCase).
- **Refine Attribute Definitions**:
  - Remove redundant attributes (e.g., carPrice in both Transaction and Cars).
  - Ensure each attribute serves a distinct purpose and is appropriately constrained (e.g., setting unique constraints where necessary).
- **Expand Entity Purposes**:
  - Provide more detailed explanations of each entity's role within the system to enhance understanding and ensure comprehensive coverage of all necessary data aspects.
- **Address Missing Relationships**:
  - Define how the Location entity interacts with other entities if applicable, or clarify why it stands independently.
- **Update ERD Diagram**

Eva Christin Griffin

I really like your project idea for a database that tracks rare car imports.

Your overview describes that this system will keep track of imported vehicles from different countries to help customers order and import a car. You also state that the database will keep track of Names (I'd call this Customers), Prices, Dates, and Transactions. I suggest adding a little bit more information to your overview to answer the following questions a stakeholder might have:

- Who is this system for? For example, is this a system for a car dealership or a special in-person or online store?
- What is the approximate scale of the database we can expect (including numbers and metrics)? For example, how many different countries do we expect to offer vehicles from? How large is the customer base? How many different types of vehicles are offered for importing? What are the average financial values the business must handle?

Your database outline is detailed and provides a good overview of the different entities. There are 4 total entities: Customer, Transaction, Cars, and Location. Here are some suggestions to make the outline more detailed and clear:

- Add a short description for the purpose of each entity, e.g. **Customers**: records the details of *Customers* the shop/dealership sells *Cars* to

Each entity has a list of attributes of different data types and constraints. Here are a few suggestions for improvement:

- Split *Customers.name* into two attributes *Customers.firstName* and *Customers.lastName*
- We will likely need an *emailAddress* attribute for *Customers*
- Rename *Customers.contactNumber* to *Customers.phoneNumber* since "contactNumber" is a little ambiguous
- Do we deliver shipments directly to the *Customers*? If so, we will need a *deliveryAddress* or *mailingAddress* attribute for *Customers*
- Make sure to mark foreign keys correctly as FK

- The *carPrice* attribute doesn't need to appear in two different entities. I would suggest making it only part of the *Cars* entity. For *Transactions,* you could consider different attributes, such as *taxAmount, shippingCost, totalCost* instead.

The database contains several 1:M and 1:1 relationships. The outline does not reflect a M:N relationship. Here are my suggestions:

- Make the relationship between *Cars* and *Transactions* 1:M. There is likely a possibility that a customer may want to order more than one vehicle at a time. Although this may be rare, the database should be prepared to handle this situation if it comes up.
- *Location* should have a 1:M relationship with *Cars*. A *Location* can have many *Cars* but a *Car* can only be stored at one *Location*.
- The *Locations* entity should only keep track of information specific to a location. I would remove the *locationFrom* and *locationTo* attributes. Since *Cars* will be related to *Locations*, we will know where a particular car is located. The *Customers* entity should store the customer's location. So we will automatically know where a car came from and where it was delivered to.
- The project must contain at least one M:N relationship, so make sure you find a way to include at least one.

The ERD will change after implementing the changes mentioned above. Here are a few suggestions when revising the ERD:

- Make sure all entity and attribute names are identical between your ERD and database outline.
- Use the correct symbols for the relationships. For example, right now your outline states that there is a 1:1 relationship between *Cars* and *Transactions*, but the ERD shows a 1:M relationship instead. (I think it *should* be 1:M but make sure it's consistent.)

The naming convention follows camelCase consistently, and all attributes are provided in singular. However, the naming is inconsistent between the database outline and ERD. Not all entity names are capitalized in the ERD. Suggestions:

- Give each entity a capitalized plural name, e.g. *Customers*, *Transactions*, and ensure that the ERD reflects the exact same names are your database outline
- Make it clear in your ERD which attributes are the foreign keys

I hope these suggestions are helpful. Overall, I think this is a great project idea and will make for an interesting database!

P3-Feedback:

Jade Zelaya

Does the UI show where it will utilize a SELECT for every table in the schema? In other words, a data table for each table in the schema should be displayed on the UI (which are not required to be populated with sample data, but should at least have column names). If yes, which tables from the schema do you see fully represented in the UI with a SELECT? If not, which tables and/or attributes are missing? Note: it is generally not acceptable for just a single query to join all tables and display them.

Every table is present in the UI and have their own represented UI.

Does at least one form utilize a search/filter/dropdown with a dynamically populated list of properties? If yes, which form(s) have which features incorporated? For which attribute(s)? If not, where are a couple places this could be implemented?

I could not find a form of search, filter or drop down with a dynamically populated list of properties but I'd recommend this feature in Transaction Details so you can filter it by year to see the total profit for a specific year.

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. If yes, which tables from their schema do you see a complete INSERT for? If not, which tables either do not have INSERTs, or have attributes missing from their INSERT? In general, do you have any suggestions for their INSERT forms?

Yes there is an insert for every table in the schema. All the insert function have a complete attribute to fill in that matches with the sample data.

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. If yes, describe all the DELETEs you see. If not, what's missing in terms of this requirement?

There is a delete function for each table's data.

Is there at least one UPDATE for any one entity, with fields for the corresponding attributes for that entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record? If yes, describe all the UPDATEs you see. If not, where would you suggest adding an UPDATE, or what's missing from the UPDATE(s) you see?

There is an update for all entities!

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. If yes, which NULLable relationship(s) do you see, and does it seem to make sense? If not, where would you suggest adding a NULLable relationship and why?

I don't see any NULL relationships.

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 like the color scheme! I recommend remaining consistent with it.

Arstanbek Bulanbekov

Does the UI show where it will utilize a SELECT for every table in the schema? In other words, a data table for each table in the schema should be displayed on the UI (which are not required to be populated with sample data, but should at least have column names). If yes, which tables from the schema do you see fully represented in the UI with a SELECT? If not, which tables and/or attributes are missing? Note: it is generally not acceptable for just a single query to join all tables and display them.

Yes all 5 Tables are displayed and have a separate page in the UI

Does at least one form utilize a search/filter/dropdown with a dynamically populated list of properties? If yes, which form(s) have which features incorporated? For which attribute(s)? If not, where are a couple places this could be implemented?

No search/filter/dropdown options were found on the UI

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. If yes, which tables from their schema do you see a complete INSERT for? If not, which tables either do not have INSERTs, or have attributes missing from their INSERT? In general, do you have any suggestions for their INSERT forms?

Yes, every INSERT option has input field for each attribute for the corresponding 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). Or, alternatively, there should be an INSERT for INSERTing into the intersection table(s) directly. If yes, list all the table INSERTs that correctly add their corresponding FK attributes, and describe the group's implementation for INSERTing into the intersection of their M:M relationship. If not, which INSERTs need to be altered, and in what way?

There are no INSERT options that will insert multiple items into different tables. All INSERT options in UI insert only a single item. All INSERTs that have a NOT NULL foreign key need the option to also add the item it refers to.

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. If yes, describe all the DELETEs you see. If not, what's missing in terms of this requirement?

No, all DELETE options delete only singular items without deleting any items from other tables they refer to. Tables with mandatory relationships might need the option to delete the item it's referring to.

Is there at least one UPDATE for any one entity, with fields for the corresponding attributes for that entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record? If yes, describe all the UPDATEs you see. If not, where would you suggest adding an UPDATE, or what's missing from the UPDATE(s) you see?

Every UPDATE option (edit in UI) asks for every attribute of an item, except its id(PK).

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. If yes, which NULLable relationship(s) do you see, and does it seem to make sense? If not, where would you suggest adding a NULLable relationship and why?

There is no mention of a NULLable or optional relationship in the schema or the document.

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 making the ID (Primary key) of each element visible. It might help working with bigger sample data since Foreign keys only show IDs of the item they refer to

Jonathan Guzman

Does the UI show where it will utilize a SELECT for every table in the schema? In other words, a data table for each table in the schema should be displayed on the UI (which are not required to be populated with sample data, but should at least have column names). If yes, which tables from the schema do you see fully represented in the UI with a SELECT? If not, which tables and/or attributes are missing? Note: it is generally not acceptable for just a single query to join all tables and display them.

Yes every table, home, cars, customers, locations, transactions, translation details has their town tab/window.

Does at least one form utilize a search/filter/dropdown with a dynamically populated list of properties? If yes, which form(s) have which features incorporated? For which attribute(s)? If not, where are a couple places this could be implemented?

No there's not, there's not a way to filter any of the data in any of the tables. Maybe start with a simple one like customers/locations name?

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. If yes, which tables from their schema do you see a complete INSERT for? If not, which tables either do not have INSERTs, or have attributes missing from their INSERT? In general, do you have any suggestions for their INSERT forms?

Yes, every table/tab has a field to insert some data into its respective 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). Or, alternatively, there should be an INSERT for INSERTing into the intersection table(s) directly. If yes, list all the table INSERTs that correctly add their corresponding FK attributes, and describe the group's implementation for INSERTing into the intersection of their M:M relationship. If not, which INSERTs need to be altered, and in what way?

There is an option to insert for data including cars and transactions (which is the M:M relationship according to the 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. If yes, describe all the DELETEs you see. If not, what's missing in terms of this requirement?

There's a button to delete data on each table by pressing delete on every row, though the SQL has a restriction on the delete of "no action"

Is there at least one UPDATE for any one entity, with fields for the corresponding attributes for that entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record? If yes, describe all the UPDATEs you see. If not, where would you suggest adding an UPDATE, or what's missing from the UPDATE(s) you see?

There's a way to edit/update information on every table by editing one row, thought the SQL has a restriction on update of "no action"

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. If yes, which NULLable relationship(s) do you see, and does it seem to make sense? If not, where would you suggest adding a NULLable relationship and why?

I don't think there's a nullable relation since the data is independent of each other. The only place that i think it could be doable could be the car price as if it the pierce hasn't assigned yet and if the car was in a transition already casade delete it maybe?

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 personally found the colors of the UI eye tiring, if they wanted to keep the same colors i would recommend using a different shade. I will say i personally love the home tab but maybe add a developer tab and have the schemas/ERD there and in home have something like a logo or the concept of the UI.

Alexa Baruela

Does the UI show where it will utilize a SELECT for every table in the schema? In other words, a data table for each table in the schema should be displayed on the UI (which are not required to be populated with sample data, but should at least have column names). If yes, which tables from the schema do you see fully

represented in the UI with a SELECT? If not, which tables and/or attributes are missing? Note: it is generally not acceptable for just a single query to join all tables and display them.

The tables that are represented with select are Cars, Customers, Locations, Transactions, and Transaction Details, but there isn't a way to hide or filter by certain columns.

Does at least one form utilize a search/filter/dropdown with a dynamically populated list of properties? If yes, which form(s) have which features incorporated? For which attribute(s)? If not, where are a couple places this could be implemented?

There is a dropdown function available when you go into the DELETE menus for all of the tables. They are in the Transactions Date in Cars, Name in Customers, Location Name in Locations, Transactions Date in Transactions, and Sales ID in Transaction Details.

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. If yes, which tables from their schema do you see a complete INSERT for? If not, which tables either do not have INSERTs, or have attributes missing from their INSERT? In general, do you have any suggestions for their INSERT forms?

The UI does implement INSERT's on their Cars, Customers, Locations, Transactions, and Transaction Details tables.

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). Or, alternatively, there should be an INSERT for INSERTing into the intersection table(s) directly. If yes, list all the table INSERTs that correctly add their corresponding FK attributes, and describe the group's implementation for INSERTing into the intersection of their M:M relationship. If not, which INSERTs need to be altered, and in what way?

There is an insert for every attribute including the M:M relationship, Cars and Transactions. The other tables include Customers, Locations, and Transaction Details.

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. If yes, describe all the DELETEs you see. If not, what's missing in terms of this requirement?

There is a DELETE for every table, and there is a DELETE that removes things from the M:M relationship, Cars and Transactions. The other tables are Customers, Locations, and Transaction Details.

Is there at least one UPDATE for any one entity, with fields for the corresponding attributes for that entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record? If yes, describe all the UPDATEs you see. If not, where would you suggest adding an UPDATE, or what's missing from the UPDATE(s) you see?

There is an update for every attribute for every table. This includes: Cars: Make, Model, Model Year, Car Value Customers: Name, ContactNumber Locations: LocationName Transactions: transactionDate, customerID, fromLocation, toLocation Transactions Details: SalesID, Make, Model, Year, Sale Price

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. If yes, which NULLable relationship(s) do you see, and does it seem to make sense? If not, where would you suggest adding a NULLable relationship and why?

There's no NULLable relationship. For their situation, I don't really see where you could add a nullable/optional relationship because they all carry important information. If they added an optional email table, maybe the customer could choose between giving their email or phone number? But otherwise, there is no need for it from what I 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.

In the Cars table, I think the column names are really clear and understandable. On the contrary, all of the other tables could use AS aliases to make it more readable (not camelCase or 2 words put together.

Actions on Feedback List:

- Separated the Schema diagram from the Home page
- Removed some additional ADD functions where they weren't needed.'
- Edited some of the tables to give better names and labels
- Made sure the tables followed the same font/style within its table group.
-

Changes based on Feedback:

We kept and added more data to our overview giving more details of what our target audience is and a rough estimate of sales and customers we expect. We also made many changes to the attributes. First off, we edited the naming conventions of our database to make it line up with the assignment requirements. The entities are now plural and the attributes are singular. We also removed some redundant attributes and added new ones that were better suited for the project. For example, we removed toLocation and fromLocation from the Locations table, and instead implemented them as foreign keys in the Transactions table. We also added an entirely new entity, because we needed to include a M:M relationship. This new entity table TransactionCars facilitates this M:M relationship between Transactions and Cars, so there can be many Transactions for one Car or many Cars in one Transaction. This intersection table completes our schema so that every table has a relationship to some other entity. Furthermore, we rehauled the ERD with the new attributes, new entity, and new relationships between everything.

For step 3, we made changes to the column names and attributes (represented within the HTML) so they are more user friendly and less confusing. For example, we changed the information in the transaction table to display Customer Name, To, and From instead of customerID, toLocation, and fromLocation which were just integers. Now it is much more readable. Some feedback that we didn't follow was that many of the review insisted that we did not include a dropdown but we did. It is in the DELETE function of each page. They also said there weren't NULLable attributes but they do exist in the SQL. For example, carValue in Cars is optional and therefore NULLable.

Overview:

The main focus that our database is created to solve is keeping track of importation of rare cars from around the world. With more than a 1,000 people getting into importing cars from Japan, Europe, Australia, United States and many other countries. Our focus is to help take in information such as customers, transaction history, cars ID etc, and location. Our target audience are car-dealerships who specialize in dealing with and have knowledge of old collectables and super rare cars. A rough estimate of consumers using our database, would probably be about 30-50% of people around the world. I would probably expect more imports to be from Japan, due to the rise of Japanese Domestic Motor (JDM) cars, such as the Nissan R34's and Silvias, and other Japanese brands, like Toyota, Mazda, and Mitsubishi. We would also see another amount of imports from Europe, with brands such as Audi, Ferrari, and Mercedez. I would say the business would be worth thousands of dollars, and could be millions.

Database Outline (Updated):

This entity is for storing data about each customer that uses our service. It will track name and contact information, and each customer will be linked to the transaction table

- Customers:

    - customerID: int, auto_increment, unique, not NULL, PK
    - name: varchar(255), not NULL
    - contactNumber: varchar(255), not NULL
    - Relationship: There is a 1:M relationship between the customer and transactions. There can only be 1 customer for each transaction, but multiple transactions for 1 customer. The customerID will be a foreign key in the transaction table.

This entity is for tracking details of each transaction made. It saves the location where the cars came from and the location where the cars went to. It also saves date and has a reference to the customer that made the transaction

- Transactions:
    - salesID: int, auto_increment, unique, not NULL, PK
    - fromLocation: int, FK
    - toLocation: int, FK
    - transactionDate: datetime, not NULL
    - customerID: int, not NULL, FK
    - Relationship: There is a M:M relationship between Transactions and Cars because there can be many transactions for one car and one transaction that involves many cars
      There is also a M:1 relationship between Transactions and Locations. There can be many transactions that occur in one location, so the toLocation and fromLocation attributes are foreign keys linking to the Locations table

This entity keeps track of details of each specific car. This means the make model and year are stored, as well as an option to include the car's approximate value

- Cars:
    - carID: int, auto_increment, unique, not NULL, PK
    - make: varchar(255), not NULL
    - model: varchar(255), not NULL
    - modelYear: year, not NULL
    - carValue: decimal(10, 2)
    - Relationship: There is a M:M relationship between Cars and Transactions

This is an intersection table between Transactions and Cars. It allows for the functionality of having multiple cars in one transaction, along with the car's sale price.

- TransactionCars:
    - transactionCarID: int, auto_increment, unique, not NULL, PK
    - salesID: int, not NULL, FK
    - carID: int, not NULL, FK
    - salePrice: decimal(10, 2), not NULL
    - Relationship: this is an intersection table to facilitate the M:M relationship between transactions and cars, so there can be many transactions for 1 car or many cars in 1 transaction

This table keeps track of different locations that cars go to and from. It just stores the name of the location for query purposes.

- Locations:
    - locationID: int, auto_increment, unique, not NULL, PK
    - locationName: varchar(255), unique
    - Relationship: 1:M relationship with the Transactions table. There can be 1 location that has many transactions.

Example Data:

Cars - make, model, modelYear, carValue
Ford, Focus, 2010, 5000
Ferrari, 360 Modena, 1999, 500000
Nissan, Skyline, 2024, 40000

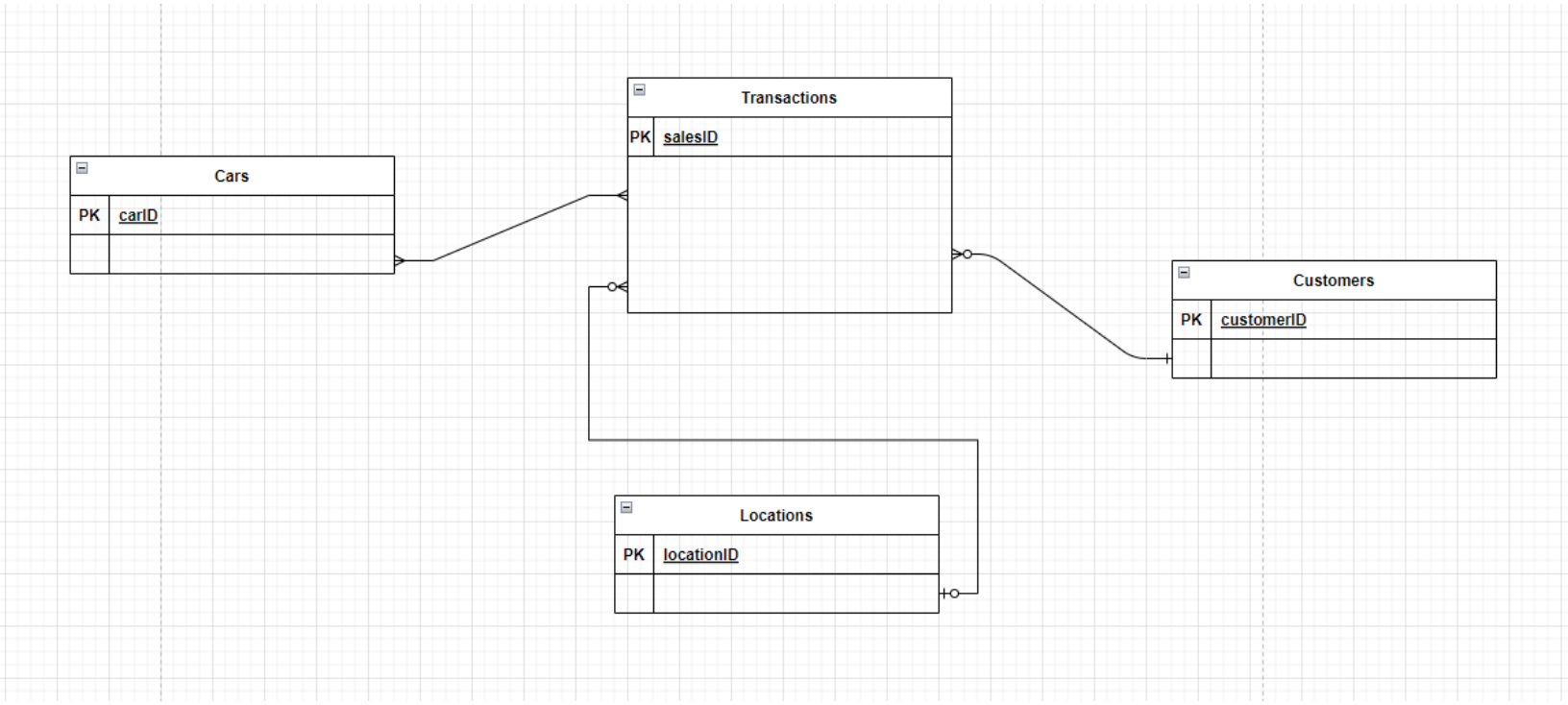
Customers - name, contactNumber
Didi Gregorious, 9079959590
Aaron Judge, 8395746378
Jazz Chisholm, 9394958690

Locations - locationName
Japan
Saudi Arabia
Norway

Transactions - transactionDate, customerID, fromLocation, toLocation
2025-02-04", 2(Aaron Judge), 1(Japan), 3(Norway)
2019-12-25", 3(Jazz Chisholm), 2(Saudi Arabia), 1(Japan)
2023-11-20", 1(Didi Gregorious), 3(Norway), 2(Saudi Arabia)

TransactionCars - salesID, carID, salePrice
1, 1(Ford Focus), 10000
2, 3(Nissan Skyline), 10000
3, 2(Ferrari 360 Modena), 1000000

Entity-Relationship Diagram (Updated):



Schema (Updated):