Final Report

Online Food Ordering Platform for Small Restaurants

AKI TEAM

12/06/2020

Alexandre Geraldo

Iffatun Nessa Mahi

Kirby Liu

Introduction

The year 2020 has been a remarkable year that has brought about many changes to the restaurant industry. Through the lockdown, many restaurants have had to become dependent on food delivery service to survive while indoor dining has either been forced to close or become a much less popular dining option. Before the pandemic, food delivery might have compromised 10%-15% of revenue however as Covid-19 hotspots are popping up throughout the country, many restaurants are reporting 90% plus delivery-based sales. Several business owners have had to reinvent their businesses to survive the changes caused by COVID-19. Small restaurant owners are particularly hit hard and are still struggling to survive in this environment. To keep businesses open, many small restaurants are forced to move to an online ordering platform such as UberEats or GrubHub. These online platforms provide all the tools necessary to put a local restaurant into the online ordering food business. That sounds fantastic but there is an issue here: These platforms charge the restaurant a fee for each transaction. These fees are incredibly unpopular, but many restaurant owners feel they have no other options as a Washington Post article, "Restaurants are barely surviving. Delivery apps will kill them ," recently highlighted the high delivery fees of popular delivery apps like Grubhub, DoorDash, and Uber eats charges both restaurant owners for each transaction. Many restaurant owners voiced concerns as these fees barely allow restaurant owners to survive an already difficult conditions brought about by the pandemic.

Our aim for this project is to provide an alternative/sustainable platform for restaurants to participate in the food delivery/gig worker economy that benefits restaurant owners, drivers, and customers. For customers the application will offer feature-rich functionalities that tracks previous order preferences, recommends new restaurants/menu, offer promotions, and provide reviews of nearby restaurants. For restaurants the application will offer simple commonly used features that include employee management, menu management, app sales notification, and promotion offerings. Our goal is to keep fees low, so that the money collected will be majority for the gig-workers that deliver the food.

Our metric for success is to -1) create a working prototype that 2) once presented to restaurant owners, potential customers, and from gig-workers receives positive feedback, and 3) lastly the payment model is sustainable from an operational/business standpoint for all stakeholders including application developers, restaurant owners, gig-workers, and customers alike.

Requirement Analysis

User Requirements

- User Requirement 1: The system will allow customers to add orders, verify orders status, cancel orders.
- User Requirement 2: The system will allow drivers to pick orders from a queue and delivering to customers.
- User Requirement 3: The system will allow customers to create favorite orders.
- User Requirement 4: The system will provide a global address catalog to avoid duplicate records added to the system. The address catalog will also provide latitude and longitude coordinates to be used in data visualizations.
- User Requirement 5: The system will allow the restaurants to specify the operation hours of the business, select a local manager, and implement the use of the global address catalog. A restaurant must also have a feature to enable or disable the ordering system as a whole.
- User Requirement 6: The system will track total sales for reporting and tax purposes for restaurants.
- User Requirement 7: The system will allow customers to evaluate orders and deliveries.
- User Requirement 8: This system will allow to store customer information such as name, delivery address, contact information.
- User Requirement 9: This system will allow to store employee information such as their name, address, working hours, productivity.
- User Requirement 10: This system is for promoting lucky day discounts for customers.
- User Requirement 11: This system will identify potential customers to give them discounts based on their bill amount.
- User Requirement 12: The system will allow restaurants will be able to build their menu offering while listing calories, prices, size, promotions, and food category of their food and drink offering.
- User Requirement 13: The system will allow restaurants will have the ability to list the ingredients of each food and tag with special dietary and/or common food allergens.
- User Requirement 14: The system will allow customers to be able to save preferred payment options of credit/debit cards (with billing address), PayPal, and/or google pay.
- User Requirement 15: The system will allow drivers to input and store their vehicle, driver's license, driver history, insurance, bank, and tax information upon enrollment.
- User Requirement 16: The system will allow customers/restaurants/drivers to input and store their contact information including email, phone number, and mailing address.
- User Requirement 17: The system will allow customers/restaurants/drivers to view their past orders and transaction receipts.

Data Requirements

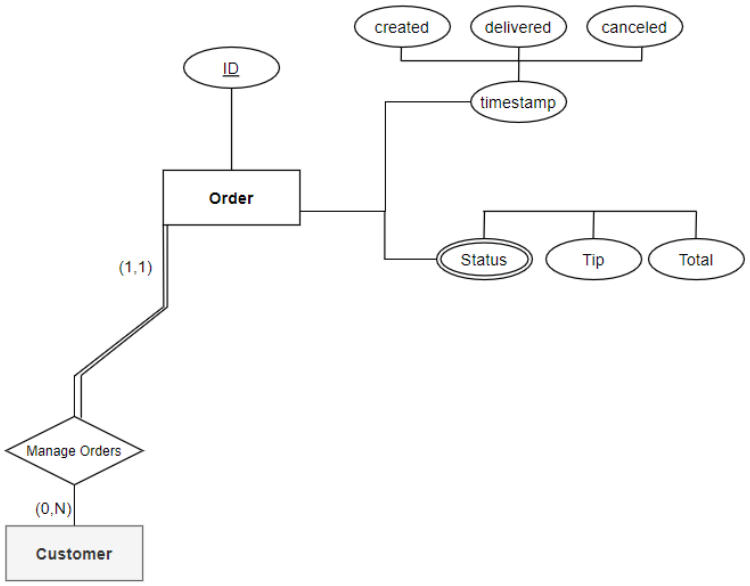
- Data Requirement 1: Restaurant data - ID, type, name, phone, website, status, open, close, day of week; Restaurant has relationship with Employee and Address.
- Data Requirement 2: Address data - ID, street name, city, state, zip code, latitude, longitude.
- Data Requirement 3: Order data - ID, created, delivered canceled, status, tip, total; Order has relationship with Order Details, Address, Customer, Payments, and Drivers.
- Data Requirement 4: Order Detail - Menu ID, Order ID, quantity, total; Order Details has relationship with Order, Meals.

- Data Requirement 5: Order Review - OrderID, rate, comments, timestamp; Order Review has relationship with Order.
- Data Requirement 6: Delivery Review - OrderID, rate, comments, timestamp; Delivery Review has relationship with Order.
- Data Requirement 7: Favorite Order - ID, nickname; Favorite Order has a relationship with Customer and Favorite Items.
- Data Requirement 8: Favorite Detail - OrderID, MenuID; Favorite Detail has a relationship with Favorite Order and Menu.
- Data Requirement 9: Customer Data - customerID, firstName, lastName, email, phone. Customer has relationship with Address, Credentials, Orders and Favorite Orders entities.
- Data Requirement 10: Employee Data - firstName, lastName, ssn, employeeID, holidayStatus {on, off}, phone. There is ISA relation among Employee, Part-time, Full-time, and Seasonal type of employee. Employee has also relationship to Address, Employee Productivity, and Restaurant entities.
    - Part-Time Data - startDate, endDate, hourlyWage, weeklyHours. This has hierarchical relation with Employee.
    - Full-time Data - startDate, endDate, hourlyWage, weeklyHours. This has hierarchical relation with Employee.
    - Seasonal Data - startDate, endDate, hourlyWage, weeklyHours. This has hierarchical relation with Employee.
- Data Requirement 11: Daily Employee Productivity Data - clockedIn. clockedOut, assignedHours, hoursEarned, goal {not reached, reached, overworked}, pID, date. Employee Productivity has relationship with Employee entity.
- Data Requirement 12: Potential Customer – order_ID, total, afterDiscount (5%). This has relationship with Order and Order_item (analytical).
- Data Requirement 13: Lucky Day Data – order_ID, total, afterDiscount, luckyDayDiscount (4%). This is related to Order and Order_item .
- Data Requirement 14: Menu data- MenuID; has a relationship with Item, Restaurant, and Order entity.
- Data Requirement 15: Item data- ItemID, Item Name, Item Category, Price, Size, Calories, and Promotion Tag; Item data has relationship with Menu and Ingredients.
- Data Requirement 16: Ingredients data- IngredientsID, Ingredients Name, Diet Adherence Tag, and Allergen; ingredients have relationship with Menu.
- Data Requirement 17: Payment data- PaymentID, Card Payment, PayPay, and Google pay; Payment data has relationship with User and Order.
- Data Requirement 18: Driver data- DriverID, Phone Number, Email, Birthday, First/Last Name, Date Start; Driver Data has relationship with Mailing Address, Credentials, Orders, Ratings, Car vehicle, Driver License, Driver Hsitory, Bank Account, 1099 Form.
- Data requirements 19: Car Vehicle data- VIN, Make, Model, Year, License Plate; Car Vehicle has relationship with Driver and Insurance.
- Data requirements 20: Insurance data- InsuranceID, Insurance company, policy number, coverage start, coverage end; Insurance data has relationship with Car Vehicle.
- Data requirements 21: Driver License data- Driver LicenseID Driver License Number, First/Last Name, Expiration, State Issue, Address, and Birthday; Driver License has relationship with Driver.
- Data Requirements 22: Driver History data- DriverID, Accidents, Tickets; Driver History has relationship with Driver.

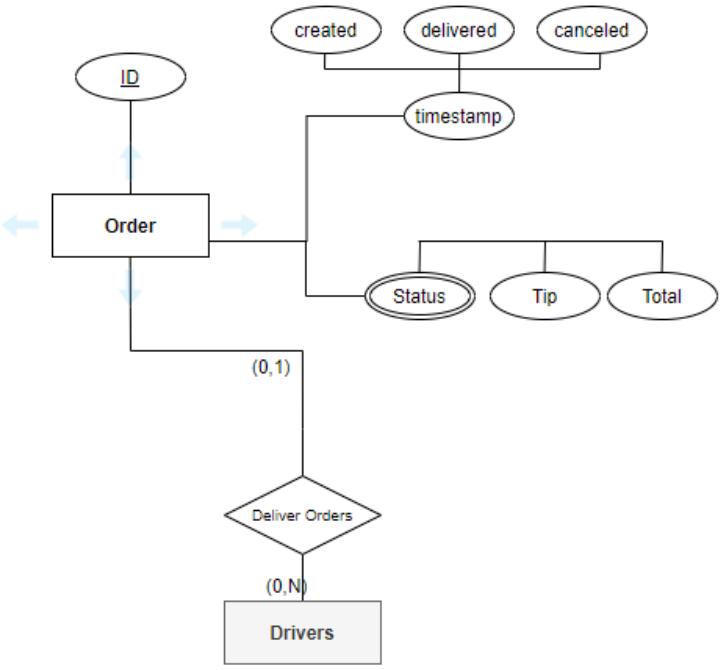
- Data Requirements 23: 1099 Form data- SSN/EIN, Name, Application Date, Address, Signature Present; 1099 Form has relationship with Driver and Restaurant.
- Data Requirements 24: Bank Account data- BankID, Account Number, and Routing Number; Bank Account has relationship with Driver and Restaurant.
- Data Requirements 25: Favorite Restaurant Detail- RestaurantID and UserID; Favorite Restaurant Detail has a relationship with Restaurant and User.

Functional Requirements

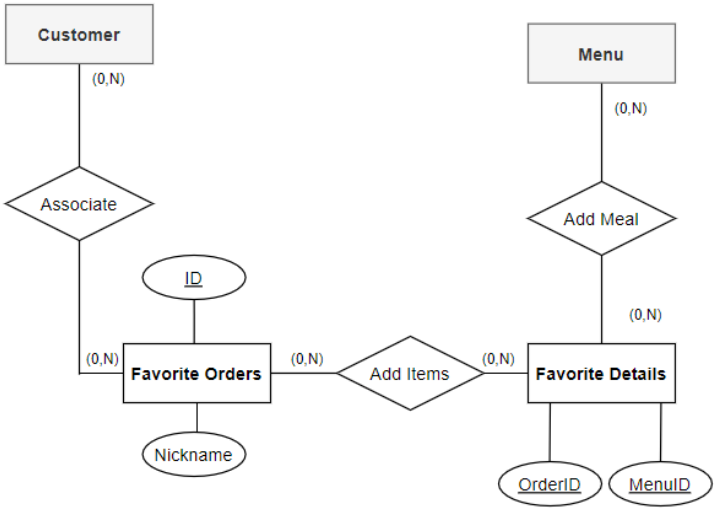
- Functional Requirement 1: Customers managing their orders with input Order ID.



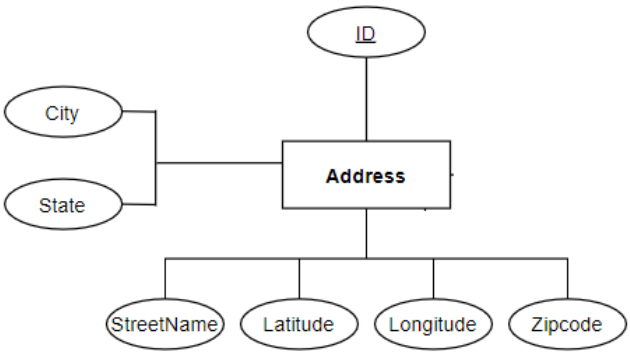
- Functional Requirement 2: Drivers picking order from Order queue with Order Status = 1 (ready to deliver).



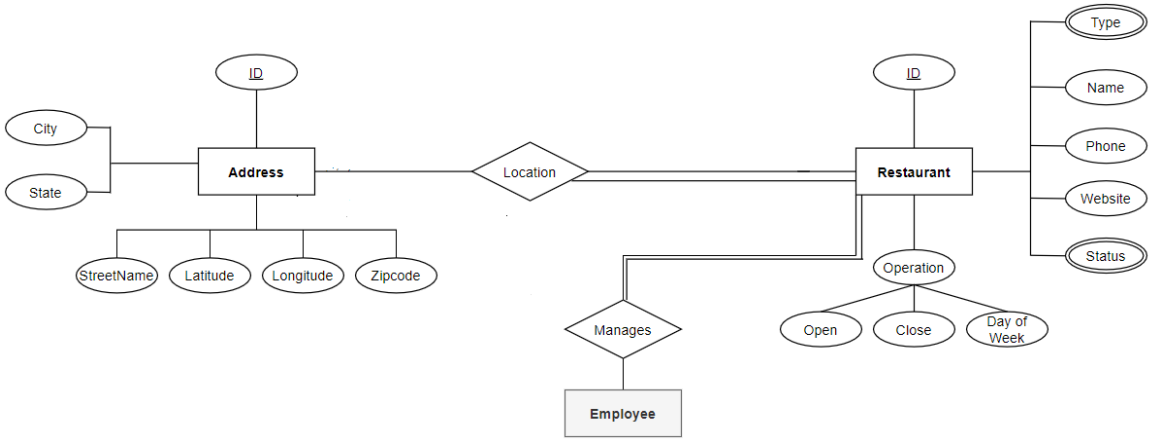
- Functional Requirement 3: Customers managing favorite orders.



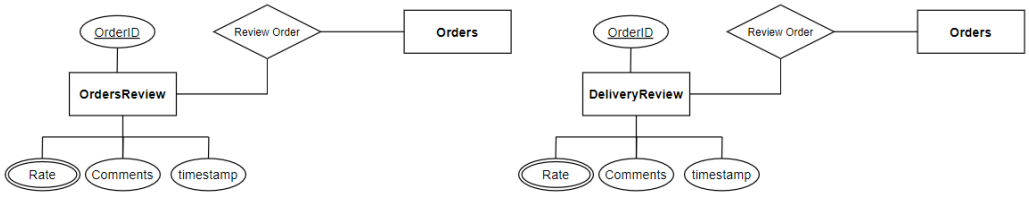
- Functional Requirement 4: Global address catalog used to associate order delivery address and restaurant address.



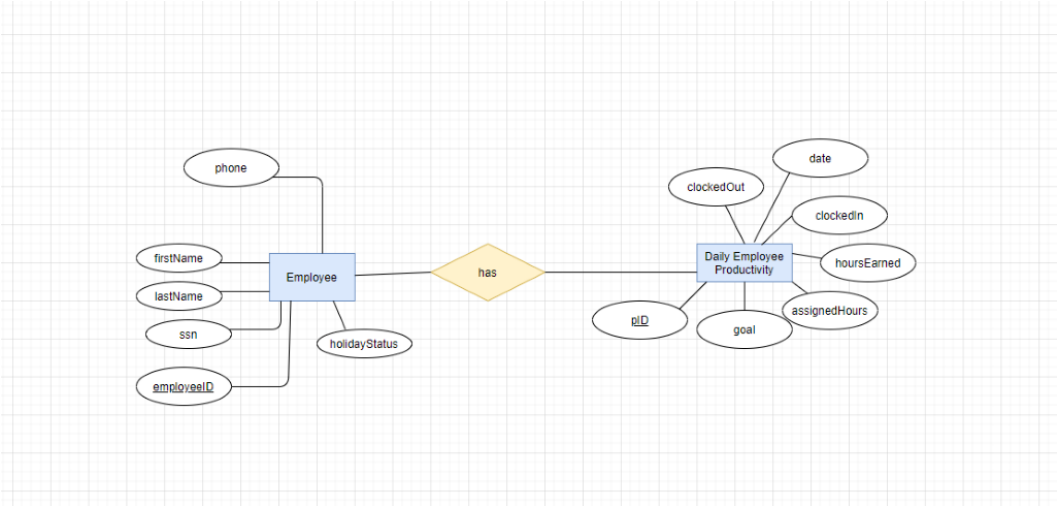
- Functional Requirement 5: Restaurant management system and relationships with Employee and Address entities.



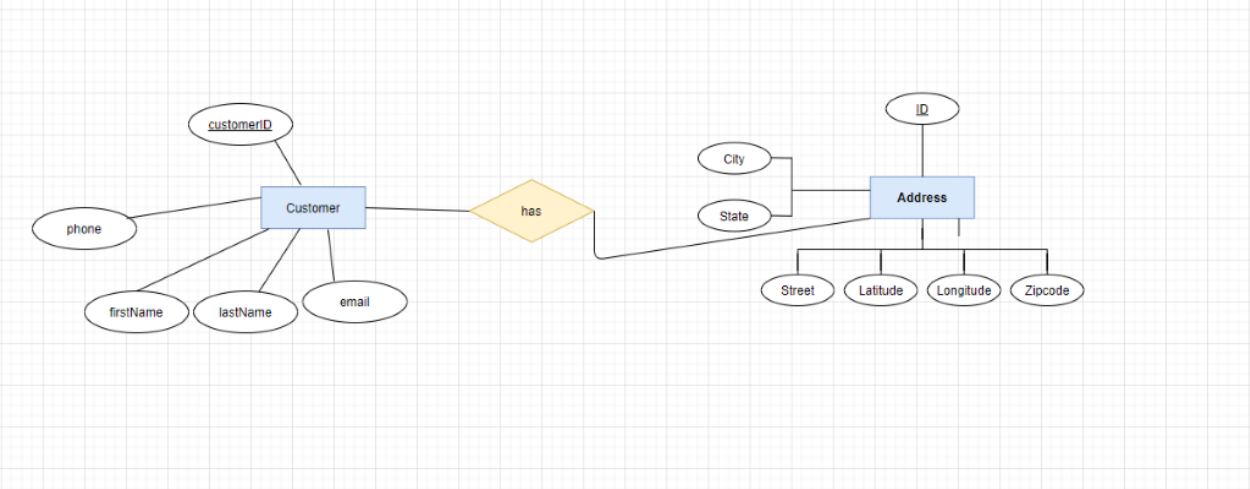
- Functional Requirement 6: Customers evaluating orders and deliveries input Order ID.



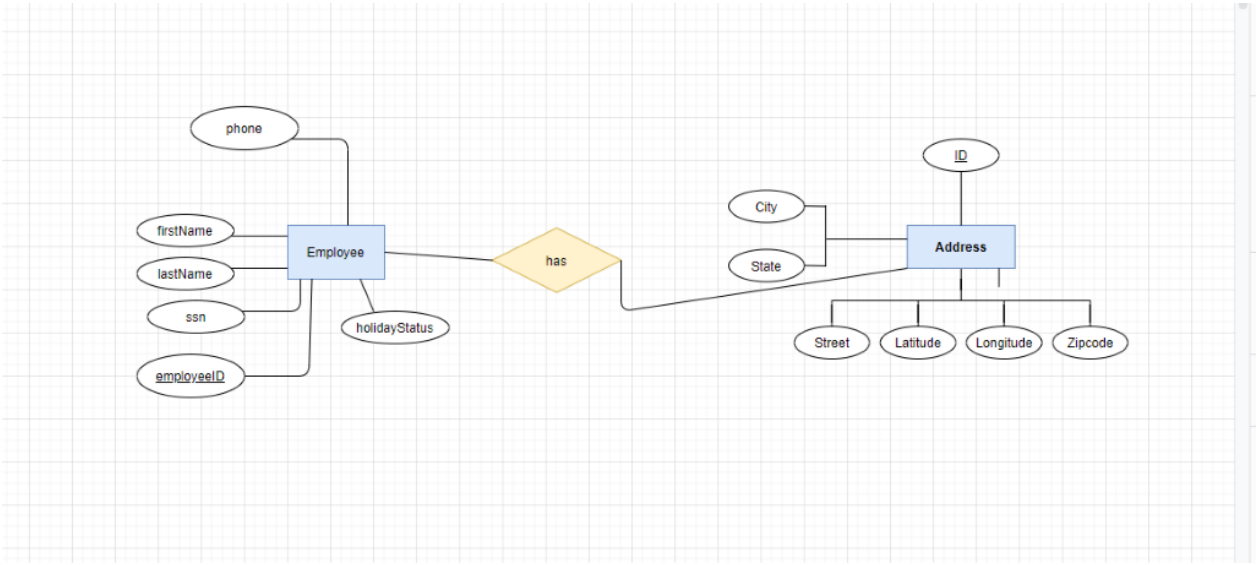
- Functional Requirement 7: Check for Daily Employee Productivity with the input of Employee ID.



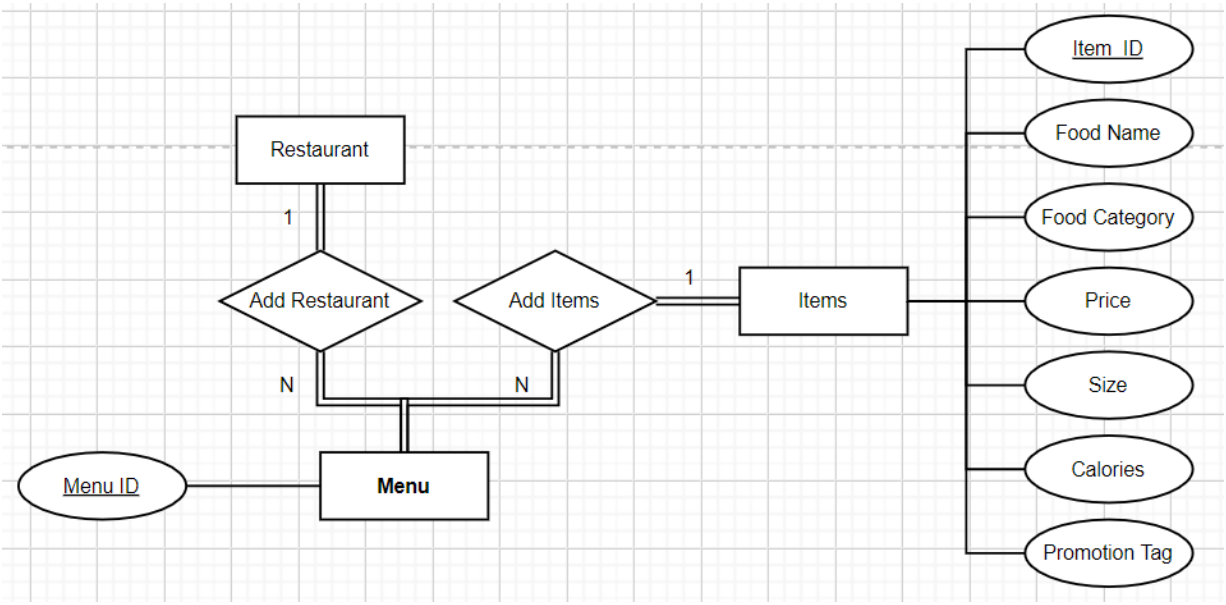
- Functional Requirement 8: From the customer ID, we can find out their street, city, state, and zip.



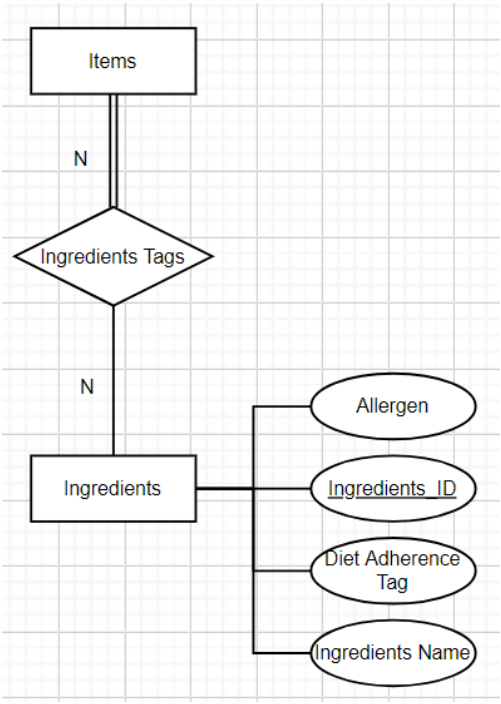
- Functional Requirement: From the employee SSN, we can find out their street, city state, and zip.



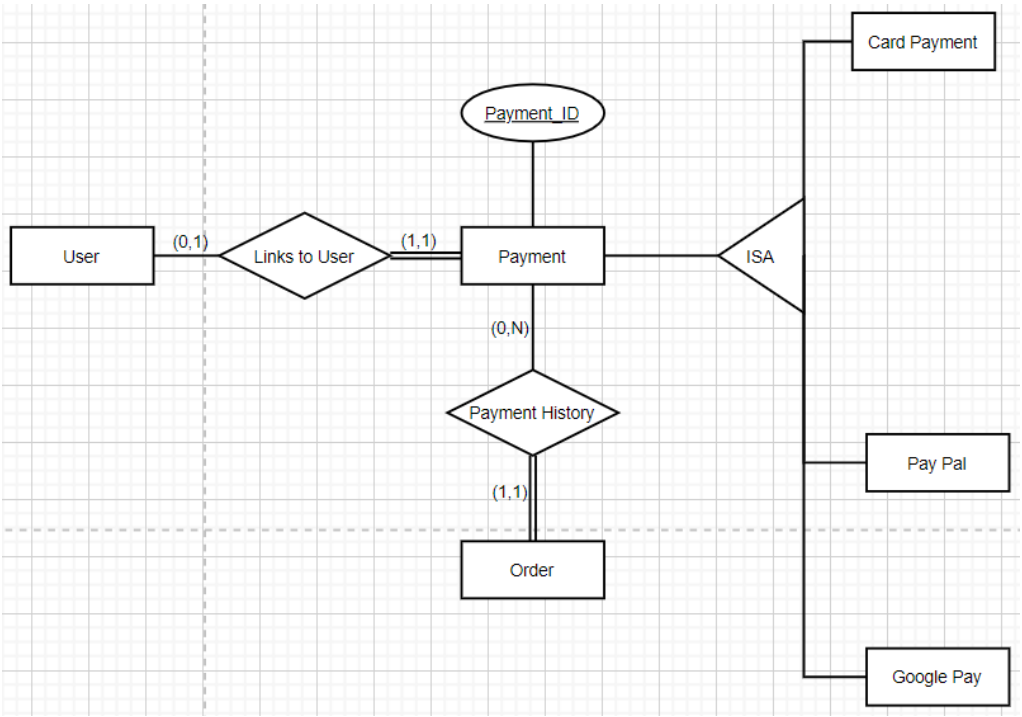
- Functional Requirement 12: Restaurants will be able to build their menu offering while listing calories, prices, size, promotions, and food category of their food and drink offering.



- Functional Requirement 13: Restaurants will have the ability to list the ingredients of each food and tag with special dietary and/or common food allergens.

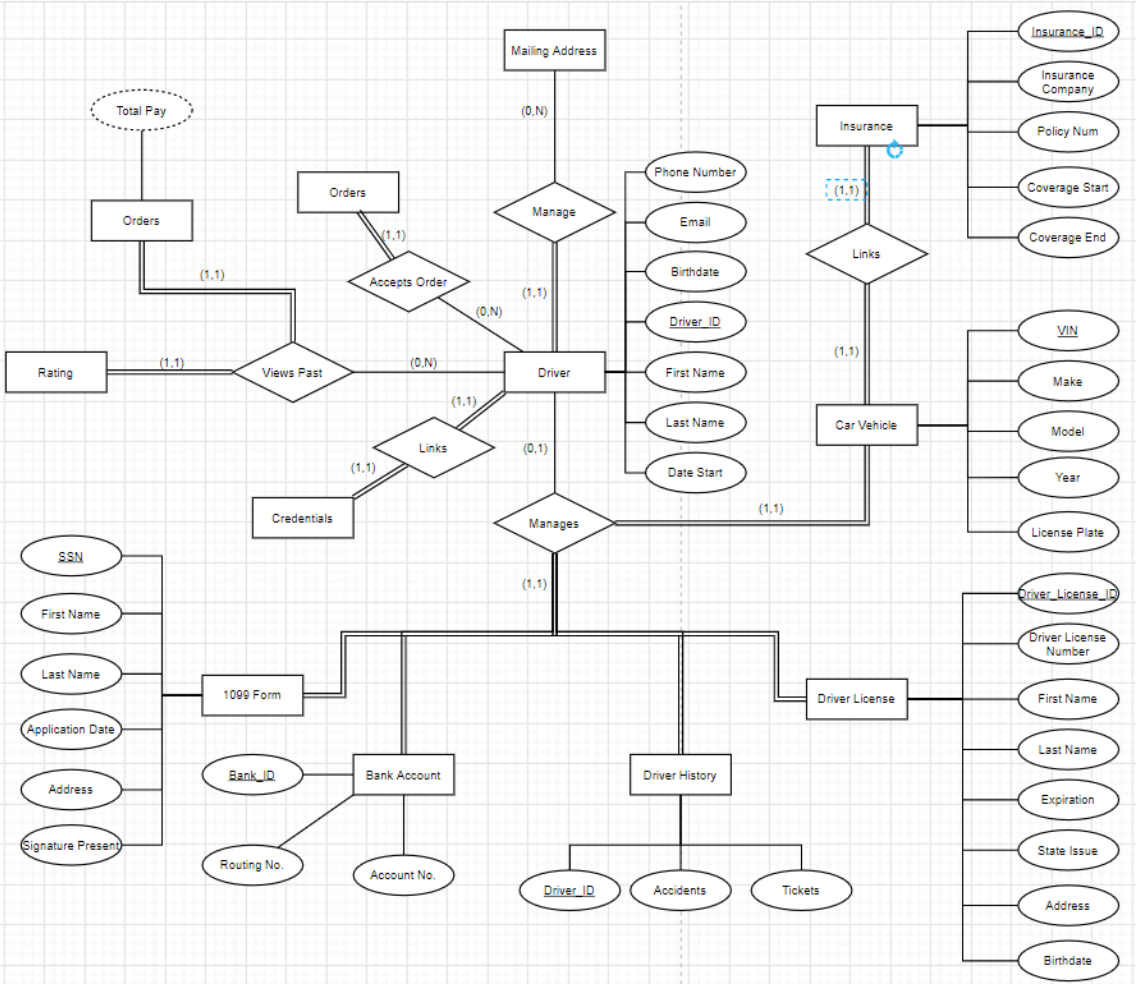


- Functional Requirement 14: Customers will be able to save preferred payment options of credit/debit cards (with billing address), paypal, and/or google pay.

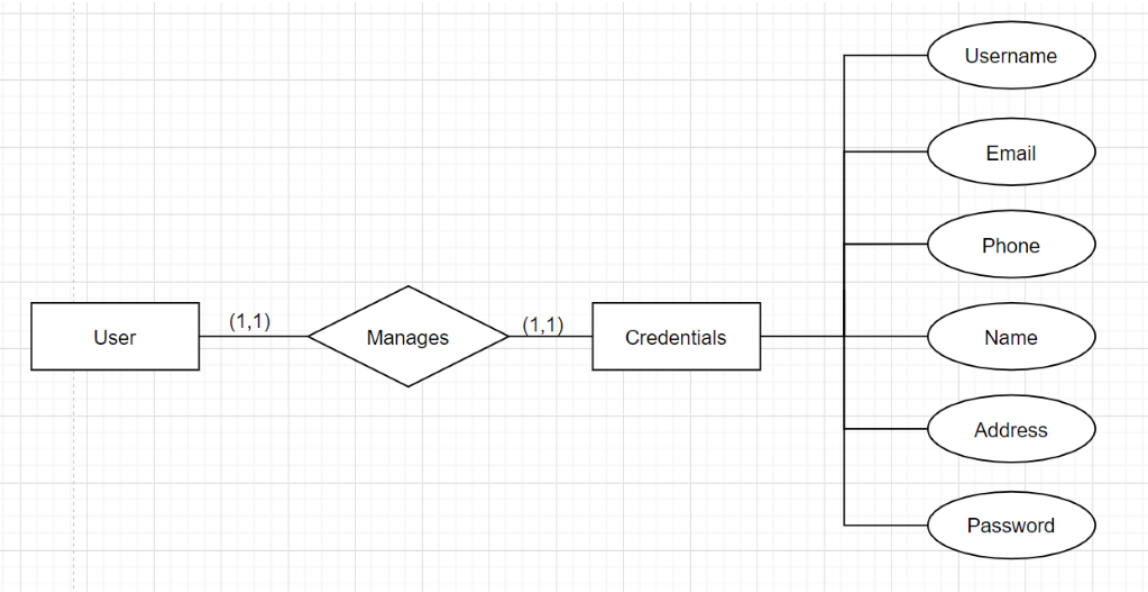


- Functional Requirement 15: Drivers to input and store their vehicle, driver's license, driver history,

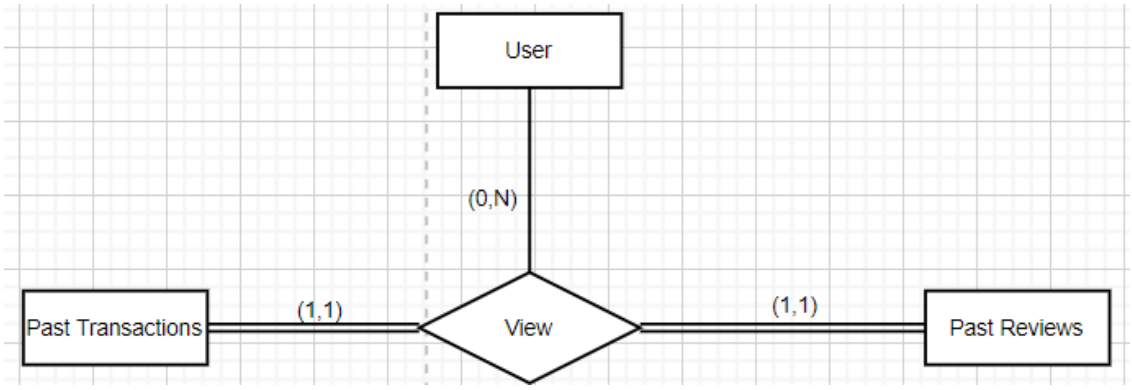
insurance, bank, and tax information upon enrollment.



- Functional Requirement 16: The system will allow customers/restaurants/drivers to input and store their credentials, contact information, including email, phone number, and mailing address.



- Functional Requirement 17: The system will allow customers/restaurants/drivers to view their past orders and transaction receipts.



Conceptual Design

Entities and Attributes

Entity 1: Restaurant

Entity: An entity used to store restaurant records.

Attributes

- ID: non-negative numeric field to encode a distinct restaurant (autogenerated).
- Name: Name of the restaurant.
- Type: Type of food provided by the restaurant (Italian, Greek, Japanese, Chinese, etc).
- Phone: Restaurant's contact phone number.
- Website: Restaurant's website URL.
- Status: Current restaurant status (1 – opened, 0 – closed)
- Open At: Start time (hh:mm:ss) of restaurant operations.
- Closed At: End time (hh:mm:ss) of restaurant operations.
- Day of Week: Operational days of week (Sun-Sat).

Relationships

- Set Restaurant Location: relationship between Restaurant and Address entities.
- Set Restaurant Manager: relationship between Restaurant and Employee entities.

Primary Key

- It is identified by the ID attribute because each restaurant will be encoded to a unique non-negative integer value.

Entity 2: Address

Entity: An entity used to store the global address catalog.

Attributes

- ID: non-negative numeric field to encode a distinct address (autogenerated).
- Street: Name of the street including the residence number.
- City: Name of the city.
- State: Name of the state.
- Zipcode: Zip code of the address.
- Latitude: Latitude coordinate of the address.
- Longitude: Longitude coordinate of the address.

Relationships

Primary Key

- It is identified by the ID attribute because each address will be encoded to a unique non-negative integer value.

Entity 3: Order

Entity: An entity used to store customer orders.

Attributes

- ID: non-negative numeric field to encode a distinct address (autogenerated).
- Created: Timestamp when the order was created.
- Delivered: Timestamp when the order was delivered.
- Canceled: Timestamp when the order was canceled.
- Status: Status of the order (1 – ready for deliver, 2 – deliver in progress, 3- deliver completed, 4 – order canceled).
- Tip: Tip amount of the order.
- Total: Total amount of the order.

Relationships

- Deliver To: relationship between Order deliver address and Address entities.
- Manage Orders: relationship between Order and Customer entities.
- Select Payment: relationship between Order and Customer's Payments entities.
- Deliver Orders: relationship between Order and Drivers entities.

Primary Key

- It is identified by the ID attribute because each order will be encoded to a unique non-negative integer value.

Entity 4: Order Detail

Entity: An entity used to store the items of an order.

Attributes

- OrderID: non-negative numeric field that relates items and order.
- MenuID: non-negative numeric field that relates items and meals.
- Quantity: Quantity of an order item.
- Total: Total amount of the order item.

Relationships

- Add Order Meal: relationship between Order Detail and Menu entities.
- Add Items: relationship between Order Detail and Order entities.

Primary Key

- It is identified by the OrderID and MenuID attributes because each order can have multiple meals (MenuID). Using these keys as primary key guarantees non-duplicate meals for the same order.

Entity 5: Order Review

Entity: An entity used to store customer order reviews.

Attributes

- OrderID: non-negative numeric field that relates Order Review and Order.
- Rate: Rate scale (from 0 to 5) of the order.
- Comments: Customer's comments about the order.
- Timestamp: Timestamp when the review record was created.

Relationships

- Review Order: relationship between Order Review and Order entities.

Primary Key

- It is identified by the OrderID attribute because it matches the relationship between Order and Order Review. An order can have only one customer review.

Entity 6: Delivery Review

Entity: An entity used to store customer delivery reviews.

Attributes

- OrderID: non-negative numeric field that relates Delivery Review and Order.
- Rate: Rate scale (from 0 to 5) of the order.
- Comments: Customer's comments about the delivery service.
- Timestamp: Timestamp when the review record was created.

Relationships

- Review Delivery: relationship between Delivery Review and Order entities.

Primary Key

- It is identified by the OrderID attribute because it matches the relationship between Order and Delivery Review. An order can have only one customer delivery review.

Entity 7: Favorite Order

Entity: An entity used to store customer favorite orders.

Attributes

- ID: non-negative numeric field to encode a distinct address (autogenerated).
- Nickname: Nickname for the favorite order.

Relationships

- Link Customer: relationship between Favorite Order and Customer entities.

Primary Key

- It is identified by the ID attribute because each favorite order will be encoded to a unique non-negative integer value.

Entity 8: Favorite Detail

Entity: An entity used to store customer favorite orders.

Attributes

- FavoritedID: non-negative numeric field that relates Favorite Order and Favorite Detail entities.
- MenuID: non-negative numeric field that relates Favorite Detail and Menu entities.

Relationships

- Add Favorite Detail: relationship between Favorite Detail and Favorite Order entities.
- Add Favorite Meal: relationship between Menu and Favorite Detail entities.

Primary Key

- It is identified by the FavoriteID and MenuID attributes because each detail of a favorite order can have multiple meals (MenuID). Using these keys as primary key guarantees non-duplicate meals for the same favorite order.

Entity 9: Customer

Entity: An entity to store customer information.

Attributes

- customerID : int type
- firstName : character type
- lastName : character type
- email : combination of char and int type
- phone : int type

Relationships

- gets : relationship between Customer and Marketing.

- isEligibleFor : relationship between Customer and Loyalty Program.
- has : relationship between Customer and Address

Primary Key

- It is identified by customerID because this id will help to separate customers and will generate unique combination (char and int) of number.

Entity 10: Loyalty Program

Entity: An entity to check if a customer is eligible to get a specific discount depending on their total amount.

Attributes

- bill : float type
- ID : int type
- discount : float type

Relationships

- isEligibleFor : relationship between Loyalty Program and Customer.

Primary Key

- ID is the primary key to identify the different types of discount.

Entity 11: Marketing

Entity: An entity to check if a customer is eligible to get a specific discount depending on their total amount.

Attributes

- oddMonthWeeklyDiscount : float type
- ID: int type
- startDate : char type
- endDate : char type
- holiday discount: float type

Relationships

- gets: relationship between Customer and Marketing.

Primary Key

- ID, startDate and endDate is the primary key to differentiate promotional period.

Entity 12: Employee

Entity: An entity to store employee's information.

Sub Entities:

Part-Time: store information for part time employees

Attributes:

- endDate : char type
- startDate : char type
- hourlyWage : float type
- weeklyHours : int type

Full-Time: store information for full time employees

Attributes:

- endDate : char type
- startDate : char type
- hourlyWage : float type
- weeklyHours : int type

Seasonal: store information for seasonal employees

Attributes:

- endDate : char type
- startDate : char type
- hourlyWage : float type
- weeklyHours : int type

Attributes

- firstName : char type
- lastName : char type
- ssn : int type
- phone : int type
- employeeID : int type
- holidayStatus : char type

Relationships

- ISA : hierarchical relationship among Employee, Part-Time, Full-Time, and Seasonal.
- has : relation between Daily Employee Productivity and Employee.
- supervision : recursive relation.

Primary Key

- employeeID is the primary key as everyone has unique id.

Entity 13: Daily Employee Productivity

Entity: An entity that keeps track of daily productivity of employees.

Attributes

- date : char type
- clockedIn : char type
- clockedOut : char type
- assignedHours : int type
- hoursEarned : int type
- pID : int type
- goal : char type

Relationships

- has : relationship between Daily Employee Productivity and Employee.

Primary Key

- pID is the primary key that will generate unique id for each clocked in employee.

Entity 14: Menu

Entity: An entity that is used to store menu records.

Attributes

- MenuID: non-negative numeric field that relates items and meals.

Relationships

- Add items: relationship between Menu and Item.
- Add Restaurant: relationship between Restaurant and Menu.

Primary Key

- It is defined by MenuID because each Menu will have a unique non-negative integer value.

Entity 15: Item

Entity: An entity that is used to store food records.

Attributes

- ItemID: non-negative numeric field that relates Items and Menu.
- Item Name: String Characters that describes items name.
- Item Category: String Characters that describes item category.

- Price: non-negative numeric field that denotes price.
- Size: String character that describes the size.
- Calories: non-negative numeric field that denotes calories.
- Promotion Tag: non-negative numeric that describes promotion or coupon.

Relationships

- Add items: relationship between Menu and Item.
- Ingredients tag: relationship between Ingredients and Menu.

Primary Key

- It is defined by ItemID because each item will be encoded with a non-negative integer value.

Entity 16: Ingredients

Entity: An entity that is used to store ingredients records.

Attributes

- IngredientsID: non-negative numeric field that relates Ingredients and Item.
- Ingredients Name: String Characters that describes items name.
- Diet Adherence Tag: String Characters that denotes the exclusion of dietary restrictions.
- Allergen: String Character that describes allergen association.

Relationships

- Ingredients tag: relationship between Ingredients and Menu.

Primary Key

- It is defined by IngredientsID because each ingredient will be encoded with a unique non-negative integer value.

Entity 17: Payment

Entity: An entity that is used to store payment records.

Attributes

- PaymentID: non-negative numeric field that relates Payments and Order.
- Card Payment: String Characters that describes if payment is card payment.
- Paypal: String Characters that describes if item is paypal.
- Google pay: String Characters that describes if item is Google pay.

Relationships

- Links to User: relationship between Payment and User.
- Payment History: relationship between Payment and Order.

Primary Key

- It is defined by PaymentID because each Payment will be encoded with a unique non-negative

integer value.

Entity 18: Driver

Entity: An entity that is used to store driver records.

Attributes

- DriverID: non-negative numeric field that relates Driver and Order.
- Phone Number: non-negative numeric field that describes telephone number.
- Email: String Characters that records email address.
- Birthday: String Characters that records Birthday.
- First/Last Name: String Characters that records First and Last Name,
- Date Start: non-negative numeric field that captures the start date of Driver.

Relationships

- Manages: relationship between Driver and Bank Account.
- Manages: relationship between Driver and Location.
- Manages: relationship between Driver and1099 Form.
- Manages: relationship between Driver and Orders.
- Manages: relationship between Driver and Driver History.
- Manages: relationship between Driver and Car Vehicle.
- Manages: relationship between Driver and Driver License.
- View Past: relationship between Driver and Order and Ratings.

Primary Key

- It is defined by DriverID because each Driver will be encoded with a unique non-negative integer value.

Entity 19: Insurance

Entity: An entity that is used to store insurance records

Attributes

- InsuranceID: non-negative numeric field that relates Driver and Insurance.
- Insurance Company: String Characters that captures the Insurance Company.
- Policy Number: non-negative numeric field that captures the policy number.
- Coverage Start: non-negative numeric field that captures the coverage start.
- Coverage End : non-negative numeric field that captures the coverage end.

Relationships

- Links to: relationship between Car Vehicle and Insurance.

Primary Key

- It is defined by InsuranceID because each Insurance will be encoded with a unique non-negative integer value.

Entity 20: Driver License

Entity: An entity that is used to store Driver License records

Attributes

- Driver_LicenseID: non-negative numeric field that relates Driver and Driver License.
- Driver License Number: non-negative numeric field that capture the driver license number.
- First/Last Name: String Characters that captures the first and last name of Driver.
- Expiration: non-negative numeric field that captures the expiration date.
- State Issue: String Characters that captures the state the issued the driver's license.
- Address: String Characters that captures the address reflected on the driver's license.
- Birthdate: non-negative numeric field that captures the birthdate.

Relationships

- Manages: relationship between Driver and Driver License

Primary Key

- It is defined by DriverID because each Driver will be encoded with a unique non-negative integer value.

Entity 21: Driver History

Entity: An entity that is used to store Driver History records.

Attributes

- DriverHistoryID: non-negative numeric field that relates Driver and Driver History.
- Accidents: non-negative numeric field that describes date and type of accidents.
- Tickets: non-negative numeric field that describes the data and type of tickets.

Relationships

- Manages: relationship between Driver and Driver History

Primary Key

- It is defined by DriverHistoryID because each Driver History will be encoded with a unique non-negative integer value.

Entity 22: 1099 Form Data

Entity: An entity that is used to store 1099 Form records.

Attributes

- SSN/EIN: non-negative numeric field that relates 1099 Form to Driver and Restaurants.
- Name: non-negative numeric field that describes date and type of accidents.
- Application Date: non-negative numeric field that describes the data and type of ticket
- Address:
- Signature Present:

Relationships

- Inputs: relationship between 1099 Form and Driver.
- Inputs: relationship between 1099 Form and Restaurants.
-

Primary Key

- It is defined by SSN/EIN because each 1099 Form will be encoded with a unique non-negative integer value that is either the SSN number for Driver or EIN number for Restaurants.

Entity 23: Bank Account Data

Entity: An entity that is used to store Bank Accounts records.

Attributes

- BankID: non-negative numeric field that relates Bank Account Data to Driver and Restaurants.
- Account Number: non-negative numeric field that describes Bank Account Number.
- Routing Number: non-negative numeric field that describes Bank Routing Number.

Relationships

- Inputs: relationship between Bank Account and Driver.
- Inputs: relationship between Bank Account and Restaurants.

Primary Key

- It is defined by BankID because each Bank Aaccount will be encoded with a unique non-negative integer value.

Entity 24: Favorite Restaurant Detail

Entity: An entity that is used to store Customer Favorite Restaurants records.

Attributes

- FavoriteRestaurantID: non-negative numeric field that relates Favorite Restaurant entities to

Customer and Restaurant.

- Customer ID: non-negative numeric field that relates Customer to Favorite Restaurant entities.
- Restaurant ID: non-negative numeric field that relates Restaurant to Favorite Restaurant entities.

Relationships

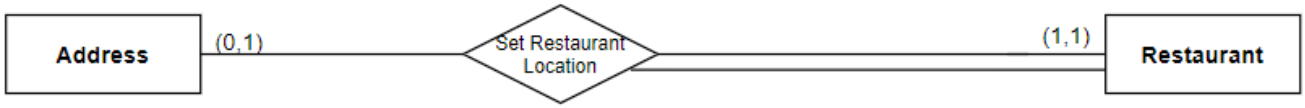
- Add Favorite Restaurant: relationship between Favorite Restaurant and Restaurant.
- Links to Customer ID: relationship between Favorite Restaurant and Customer.

Primary Key

- It is defined by FavoriteRestaurantID because each Favotire Restaurant Detial will be encoded with a unique non-negative integer value.

Relationships

Relationship 1: Set Restaurant Location



Relation: Shows the relation between Address and Restaurant entities. This relation is used to link a restaurant to a specific address location.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Address entity.
- AddressID (Foreign Key from Entity 2): Attribute on Restaurant entity used to establish the relationship with Address entity.

Cardinalities:

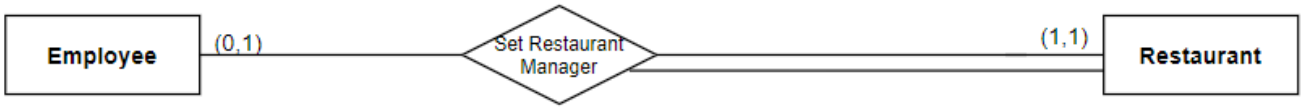
- Address (Address can have at least zero relationships with Restaurant and at most one relationship).

  Participation: 0, Cardinality: 1

- Restaurant (Restaurant can have at least one relationship with Address and at most one relationship. It implies a constraint that does not allow a Restaurant without an address associated).

  Participation: 1, Cardinality: 1

Relationship 2: Set Restaurant Manager



Relation: Shows the relation between Employee and Restaurant entities. This relation is used to link a restaurant to a specific manager (Employee instance).

Attributes

- SSN (Foreign Key from Entity1): Primary key attribute on Employee entity.
- ManagerSSN (Foreign Key from Entity 2): Attribute on Restaurant entity used to establish the relationship with Employee entity.

Cardinalities:

- Employee (Employee can have at least zero relationships with Restaurant and at most one relationship).

  Participation: 0, Cardinality: 1

- Restaurant (Restaurant can have at least one relationship with Employee and at most one relationship. It implies a constraint that does not allow a Restaurant without a manager associated).

  Participation: 1, Cardinality: 1

Relationship 3: Deliver To

Relation: Shows the relation between Address and Order entities. This relation is used to link an order to a deliver address (Address instance).

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Address entity.
- AddressID (Foreign Key from Entity 2): Attribute on Order entity used to establish the relationship with Address entity.

Cardinalities:

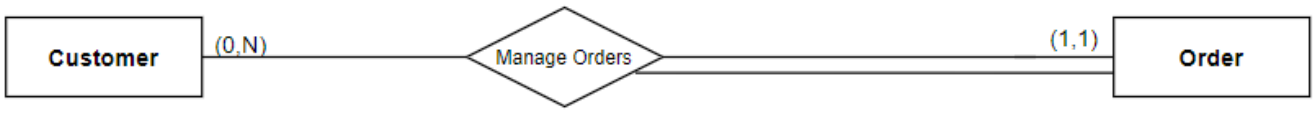
- Address (Address can have at least zero relationships with Order and at most one relationship).

  Participation: 0, Cardinality: N

- Order (Order can have at least one relationship with Address and at most N relationships).

  Participation: 1, Cardinality: 1

Relationship 4: Manage Orders



Relation: Shows the relation between Customer and Order entities. This relation is used to link an order to a customer (Customer instance).

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Customer entity.
- CustomerID (Foreign Key from Entity 2): Attribute on Order entity used to establish the relationship with Customer entity.

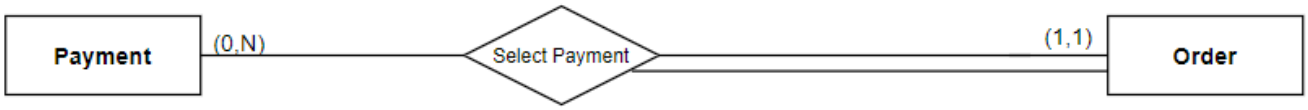
Cardinalities:

- Customer (Customer can have at least zero relationships with Order and at most N relationships).

  Participation: 0, Cardinality: N
- Order (Order can have at least one relationship with Customer and at most one relationship).

  Participation: 1, Cardinality: 1

Relationship 5: Select Payment



Relation: Shows the relation between Payment and Order entities. This relation is used to link an order to a payment method (Payment instance).

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Payment entity.
- PaymentID (Foreign Key from Entity 2): Attribute on Order entity used to establish the relationship with Payment entity.

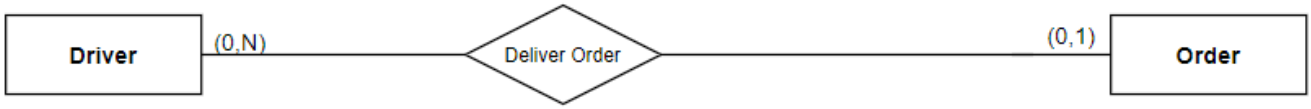
Cardinalities:

- Payment (Payment can have at least zero relationships with Order and at most N relationships).

  Participation: 0, Cardinality: N
- Order (Order can have at least one relationship with Payment and at most one relationship).

  Participation: 1, Cardinality: 1

Relationship 6: Deliver Order



Relation: Shows the relation between Driver and Order entities. This relation is used to link an order to a driver that will deliver the customer order.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Driver entity.
- DriverID (Foreign Key from Entity 2): Attribute on Order entity used to establish the relationship with Driver entity.

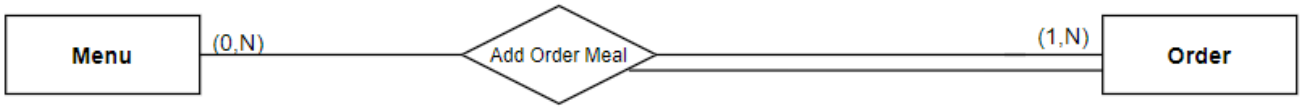
Cardinalities:

- Driver (Driver can have at least zero relationships with Order and at most N relationships).

  Participation: 0, Cardinality: N
- Order (Order can have at least zero relationship with Driver and at most one relationship).

  Participation: 0, Cardinality: 1

Relationship 7: Add Order Meal



Relation: Shows the relation between Menu and Order entities. This relation is used to link an order to a menu item (food or beverage) which is part of customer order.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Menu entity.
- MenuID (Foreign Key from Entity 2): Attribute on Order entity used to establish the relationship with Menu entity.

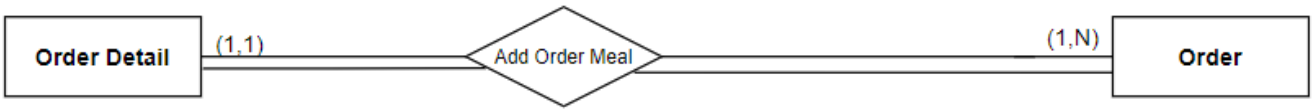
Cardinalities:

- Menu (Menu can have at least zero relationships with Order and at most N relationships).

  Participation: 0, Cardinality: N
- Order (Order can have at least one relationship with Menu and at most N relationship).

  Participation: 1, Cardinality: N

Relationship 8: Add Items



Relation: Shows the relation between Order Detail and Order entities. This relation is used to link an order to order detail.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Order entity.
- OrderID (Foreign Key from Entity 2): Attribute on Order Detail entity used to establish the relationship with Order entity.

Cardinalities:

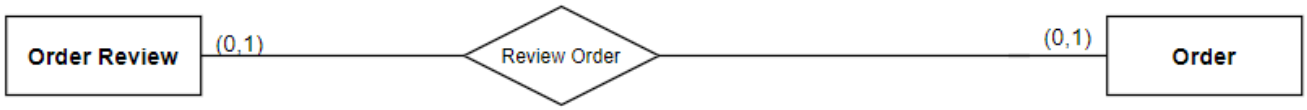
- Order Detail (Order Detail can have at least one relationship with Order and at most one relationship).

  Participation: 1, Cardinality: 1

- Order (Order can have at least one relationship with Order Detail and at most N relationship).

  Participation: 1, Cardinality: N

Relationship 9: Review Order



Relation: Shows the relation between Order Review and Order entities. This relation is used to link an order review to order.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Order entity.
- OrderID (Foreign Key from Entity 2): Attribute on Order Review entity used to establish the relationship with Order entity.

Cardinalities:

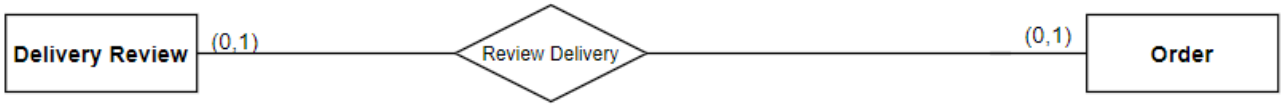
- Order Review (Order Detail can have at least zero relationship with Order and at most one relationship).

  Participation: 0, Cardinality: 1

- Order (Order can have at least zero relationship with Order Review and at most one relationship).

  Participation: 0, Cardinality: 1

Relationship 10: Review Delivery



Relation: Shows the relation between Delivery Review and Order entities. This relation is used to link an delivery review to order.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Order entity.
- OrderID (Foreign Key from Entity 2): Attribute on Delivery Review entity used to establish the relationship with Order entity.

Cardinalities:

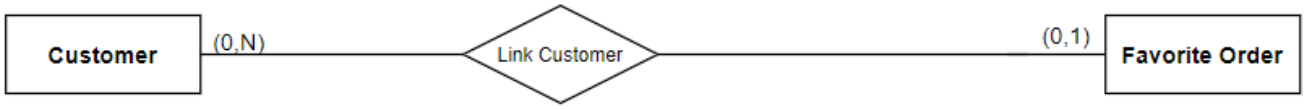
- Delivery Review (Order Detail can have at least zero relationship with Order and at most one relationship).

  Participation: 0, Cardinality: 1

- Order (Order can have at least zero relationship with Order Review and at most one relationship).

  Participation: 0, Cardinality: 1

Relationship 11: Link Customer



Relation: Shows the relation between Customer and Favorite Order entities. This relation is used to link a customer to a favorite order.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Customer entity.
- CustomerID (Foreign Key from Entity 2): Attribute on Favorite Order entity used to establish the relationship with Customer entity.

Cardinalities:

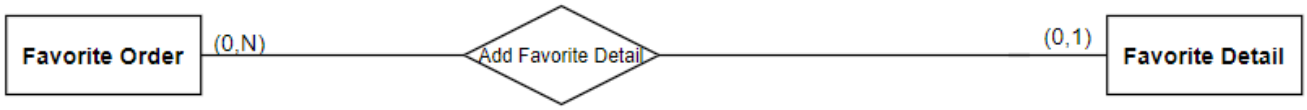
- Customer (Customer can have at least zero relationship with Favorite Order and at most N relationships).

  Participation: 0, Cardinality: N

- Order (Order can have at least zero relationship with Order Review and at most one relationship).

  Participation: 0, Cardinality: 1

Relationship 12: Add Favorite Detail



Relation: Shows the relation between Favorite Order and Favorite Detail entities. This relation is used to link the details of a favorite item to the favorite order.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Favorite Order entity.
- FavoriteID (Foreign Key from Entity 2): Attribute on Favorite Detail entity used to establish the relationship with Favorite Order entity.

Cardinalities:

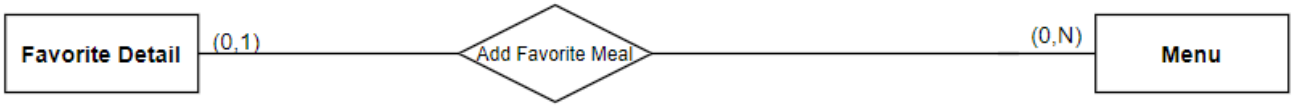
- Favorite Order (Favorite Order can have at least zero relationship with Favorite Detail and at most N relationships).

  Participation: 0, Cardinality: N

- Favorite Detail (Order can have at least zero relationship with Order Review and at most one relationship).

  Participation: 0, Cardinality: 1

Relationship 13: Add Favorite Meal



Relation: Shows the relation between Favorite Detail and Menu entities. This relation is used to link the details of a favorite item to a menu (food, beverage) item.

Attributes

- ID (Foreign Key from Entity1): Primary key attribute on Menu entity.
- MenuID (Foreign Key from Entity 2): Attribute on Favorite Detail entity used to establish the relationship with Menu entity.

Cardinalities:

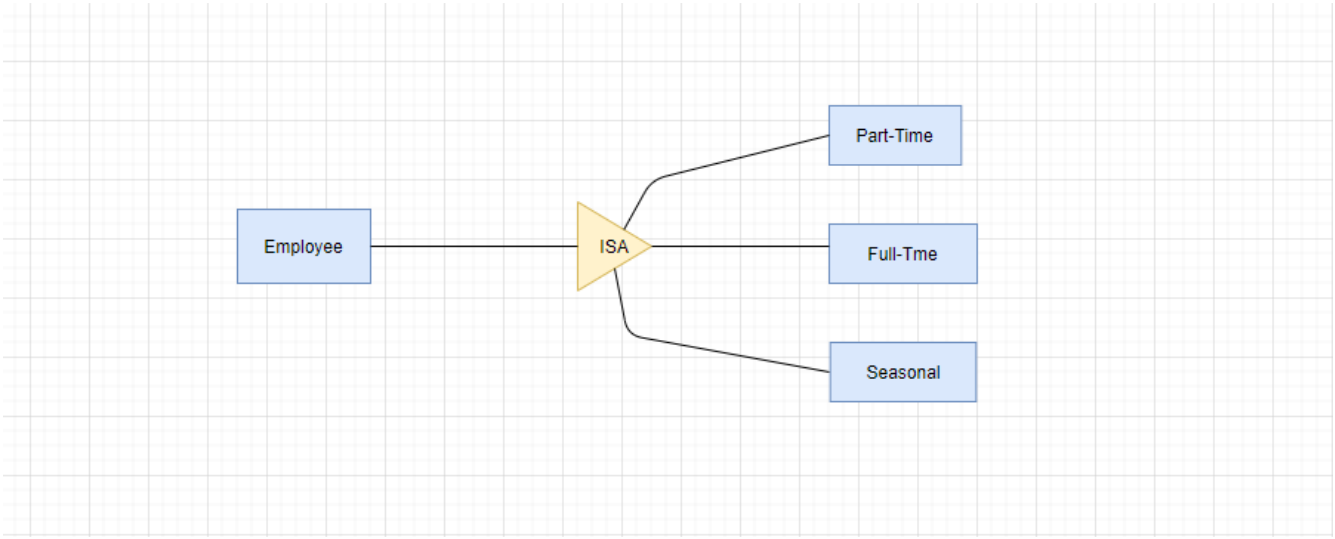
- Favorite Detail (Favorite Detail can have at least zero relationship with Menu and at most one relationship).

  Participation: 0, Cardinality: 1

- Menu (Menu can have at least zero relationship with Favorite Detail and at most N relationships).

  Participation: 0, Cardinality: N

Relationship 14: ISA



Relation: Shows hierarchical relation among Employee, Part-Time, Full-Time and Seasonal entities. This shows that there are three types of Employee.

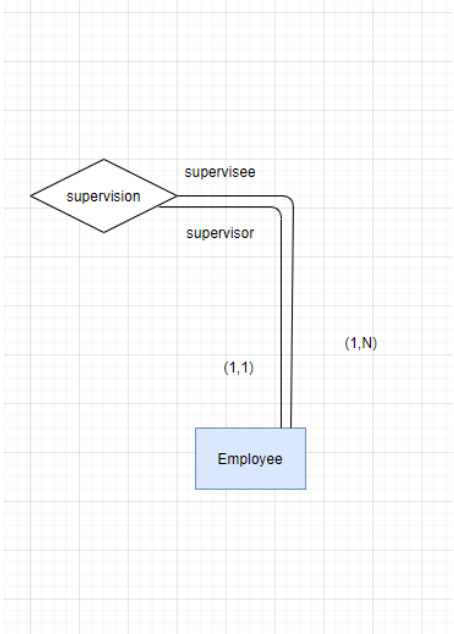
Attributes

- employeeID : Primary key attribute of Employee entity.

Cardinalities:

- No cardinalities.

Relationship 14: supervision



Relation: Shows recursive relation of Employee entity. Meaning, supervisor and supervisee are also in

the employee entity.

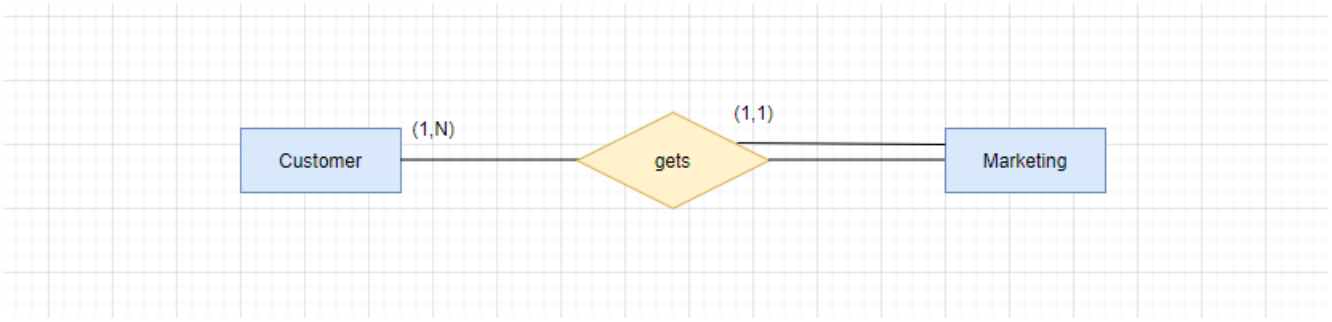
Attributes

- superssn : Foreign key from Employee Entity. Only supervisor will have the value of superssn. All the rows of superssn will be NULL.

Cardinalities:

- supervisor has cardinalities (1,1), only one of the Employee can be supervisor.
- Supervisee has cardinalities (1,N), 1 or more people can be supervised from the Employee entity.

Relationship 15: gets



Relation: relationship between Customer and Marketing.

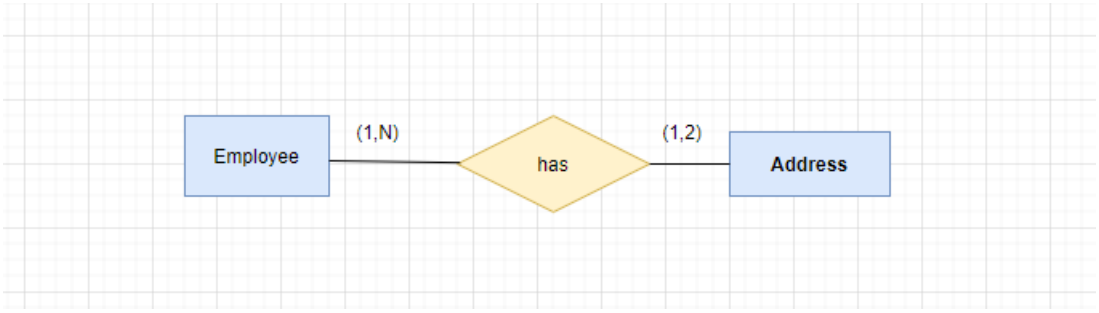
Attributes

- customerID (foreign key of OrderID) : Primary key of Customer.
- ID+startDate+endDate : Primary key of Marketing.

Cardinalities:

- Customer (1,N) : 1 to N customer can get Marketing discount or not.
- Marketing (1,1) : 1 marketing discount can have 1 customer, no more, no less.

Relationship 16: has



Relation: relationship between Address and Employee.

Attributes

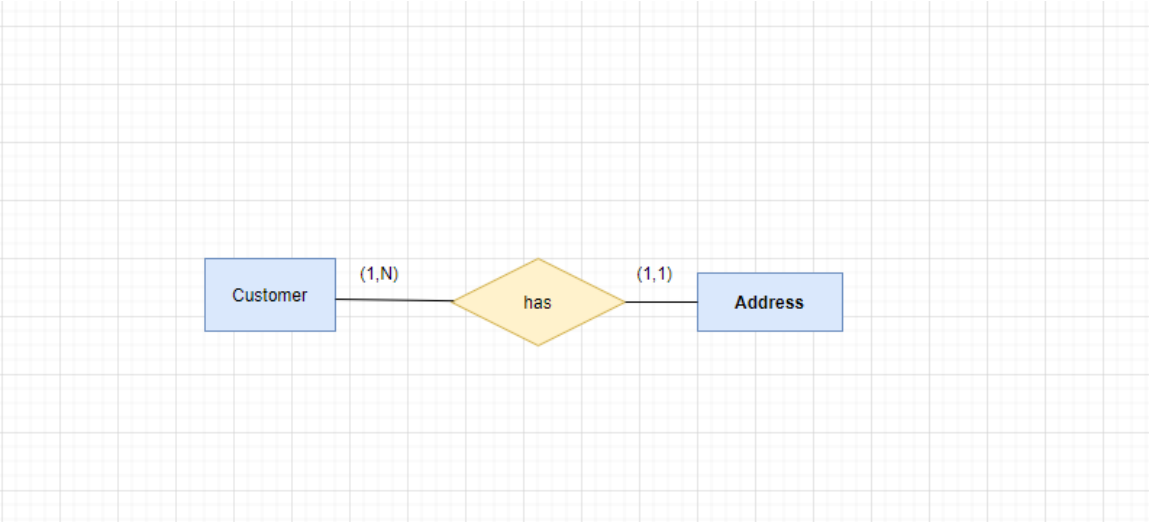
- employeeID : primary key

- ID (foreign key from Address entity) :also, from Address entity, primary key

Cardinalities:

- Employee (1,N) : 1 to N employees can have addresses.
- Address (1,2) : Employee can have minimum 1 address or maximum 2 addresses.

Relationship 17: has



Relation: relation between Customer and Address.

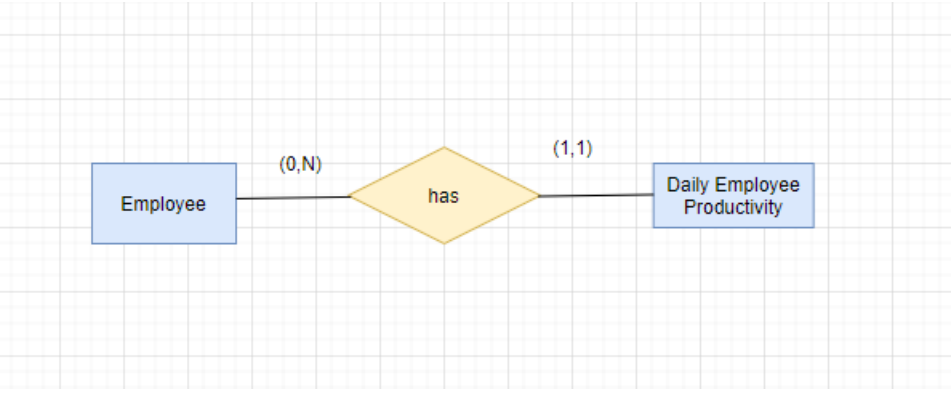
Attributes

- ID (foreign key from Address entity) :also, from Address entity, primary key
- customerID (foreign key of OrderID) : Primary key of Customer

Cardinalities:

- Customer (1,N) : 1 to N customers can have addresses.
- Address (1,1) : Customer can have only 1 address.

Relationship 18: has



Relation: relation between Employee and Daily Employee Productivity.

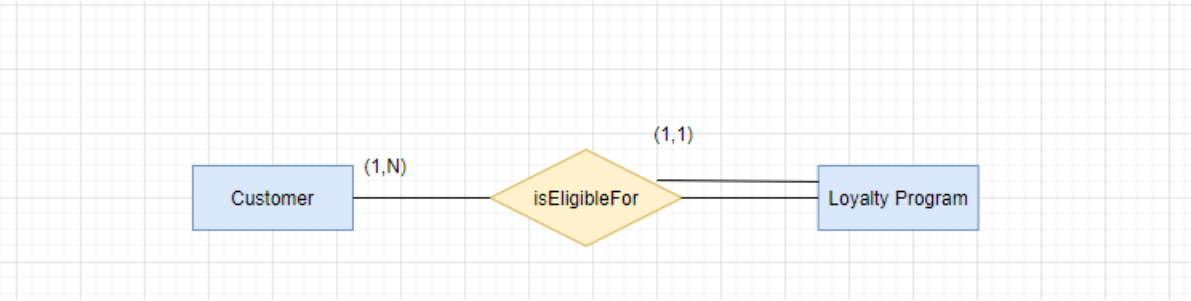
Attributes

- employeeID : primary key
- pID (foreign key of Employee entity's employeeID) : primary key of Daily Employee Productivity.

Cardinalities:

- Employee (0,N) : none to many employees can have daily productivity.
- Daily Employee Productivity (1,1) : per employee can have 1 daily productivity.

Relationship 19: isEligibleFor



Relation: relationship between Customer and Loyalty Program to detect who can be eligible for this type of program.

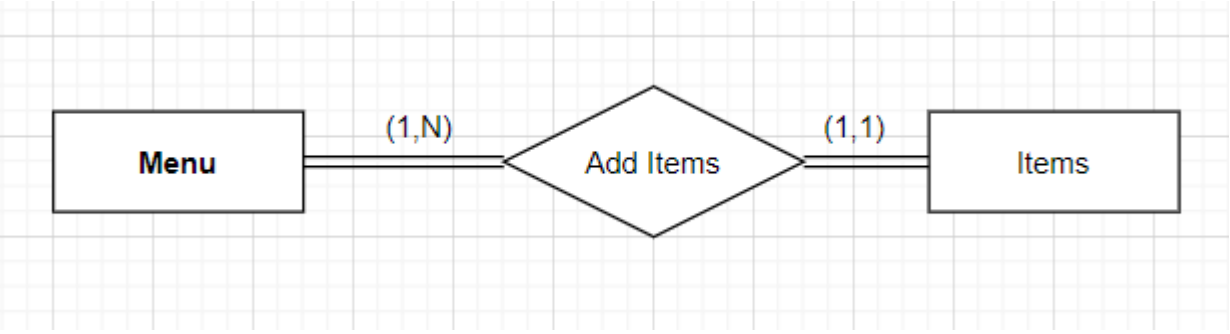
Attributes

- customerID (foreign key of OrderID) : Primary key of Customer.
- ID (foreign key from Orders entity of OrderID) : primary key of Loyalty Program.

Cardinalities:

- Customer (1,N) : 1 to N customers can be eligible for this program.
- Loyalty Program (1,1) : only 1 customer can have 1 loyalty program discount.

Relationship 20: Add Items



Relation: Shows the relation between Items entities and Menu entities. This relation is used to link the add items for restaurants to menu.

Attributes

ItemsID (Foreign Key from Entity1): Primary key attribute on Items entity.

MenuID (Foreign Key from Entity 2): Primary key attribute on Menu entity.

Cardinalities:

Item (Items can have at least 1 relationship with Menu and at most one relationship).

Participation: 1, Cardinality: 1

Menu (Menu can have at least 1 relationship with Item and at most N relationships).

Participation: 1, Cardinality: N

Relationship 21: Ingredients Tag

Relation: Shows the relation between Items entities and Ingredients entities. This relation is used to link the add ingredients for restaurants to menu items.

Attributes

ItemsID (Foreign Key from Entity1): Primary key attribute on Items entity.

IngredientsID (Foreign Key from Entity 2): Primary key attribute on Ingredients.

Cardinalities:

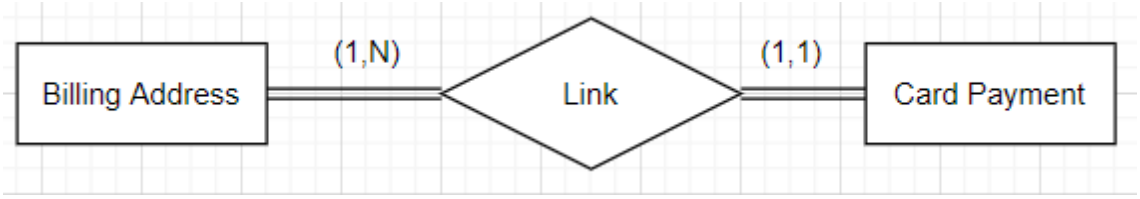
Item (Items can have at least 1 relationship with Ingredients and at most N relationships).

Participation: 1, Cardinality: N

Ingredients (Menu can have at least 0 relationship with Item and at most N relationships).

Participation: 0, Cardinality: N

Relationship 22: Link



Relation: Shows the relation between Billing Address entities and Card Payment entities. This relation is used to link the Card Payment for Customers to Billing Address.

Attributes

AddressID (Foreign Key from Entity1): Primary key attribute on Billing Address entity.

Card_PaymentID (Foreign Key from Entity 2): Primary key attribute on Card Payment entity.

Cardinalities:

Billing Address (Items can have at least 1 relationship with Card Payment and at most N relationships).

Participation: 1, Cardinality: N

Card Payment (Card Payment can have at least 1 relationship with Card Payment and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 23: Payment History

Relation: Shows the relation between Payment entities and Orders entities. This relation is for Customers to view the Payment for past orders.

Attributes

PaymentID (Foreign Key from Entity1): Primary key attribute on Payment entity.

OrderID (Foreign Key from Entity 2): Primary key attribute on Order entity.

Cardinalities:

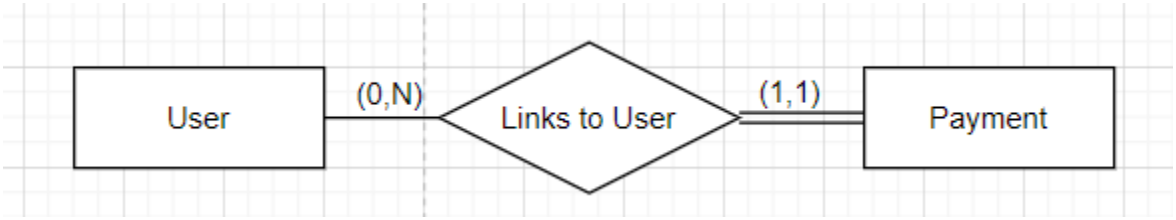
Payment (Payment can have at least 0 relationship with Order and at most N relationships).

Participation: 0, Cardinality: N

Order (Order can have at least 1 relationship with Payment and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 24: Payment



Relation: Shows the relation between User entities and Payment entities. This relation is for Customers to view their Payment methods.

Attributes

PaymentID (Foreign Key from Entity1): Primary key attribute on Payment entity.

UserID (Foreign Key from Entity 2): Primary key attribute on User entity.

Cardinalities:

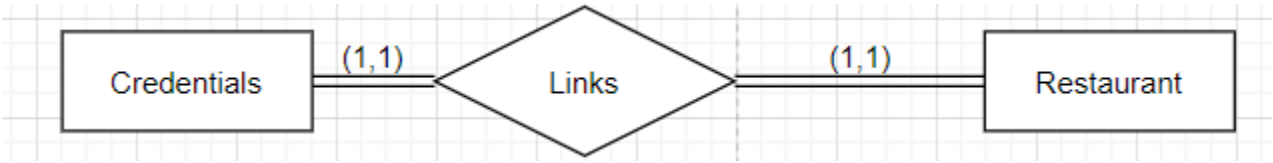
Payment (Payment can have at least 1 relationship with User and at most 1 relationships).

Participation: 1, Cardinality: 1

User (Order can have at least 1 relationship with Payment and at most 1 relationship).

Participation: 0, Cardinality: N

Relationship 25: Links



Relation: Shows the relation between Credentials entities and Restaurant entities. This relation is for Restaurants to view/change their Credentials.

Attributes

UserID (Foreign Key from Entity1): Primary key attribute on Credentials entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

Credentials (Credentials can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Order can have at least 1 relationship with Credentials and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 26: Manages

Relation: Shows the relation between Menu entities and Restaurant entities. This relation is for Restaurants to view/change their Menu.

Attributes

MenuID (Foreign Key from Entity1): Primary key attribute on Menu entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

Menu (Menu can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Restaurant can have at least 1 relationship with Menu and at most 1 relationship).

Participation: 1, Cardinality: 1

1.

Relationship 27: Manages

Relation: Shows the relation between Bank Account entities and Restaurant entities. This relation is for Restaurants to view/change their Bank Account.

Attributes

Bank_AccountID(Foreign Key from Entity1): Primary key attribute on Bank Account entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

Bank Account (Bank Account can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Restaurant can have at least 1 relationship with Bank Account and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 28: Manages

Relation: Shows the relation between 1099 Form entities and Restaurant entities. This relation is for Restaurants to view/change their 1099 Form.

Attributes

EINID(Foreign Key from Entity1): Primary key attribute on 1099 Form entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

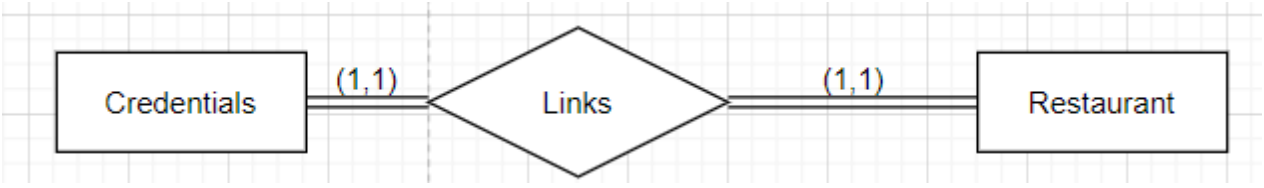
1099 Form (1099 Form can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Restaurant can have at least 1 relationship with 1099 Form and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 29: Links



Relation: Shows the relation between Credentials entities and Restaurant entities. This relation is for Restaurants to view/change their Credentials.

Attributes

UsernameID(Foreign Key from Entity1): Primary key attribute on Credentials entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

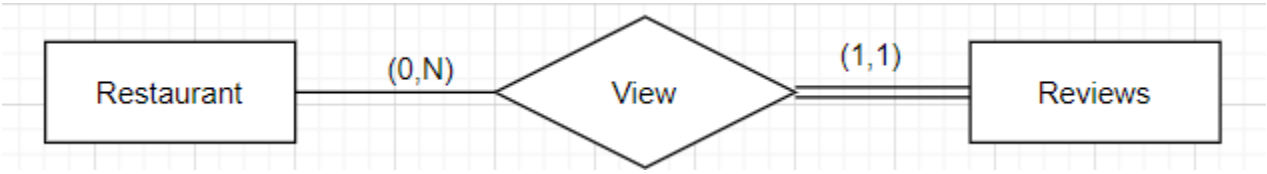
Credentials (Credentials can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Restaurant can have at least 1 relationship with Credentials and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 30: View



Relation: Shows the relation between Review entities and Restaurant entities. This relation is for Restaurants to view their Reviews.

Attributes

ReviewsID(Foreign Key from Entity1): Primary key attribute on Reviews entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

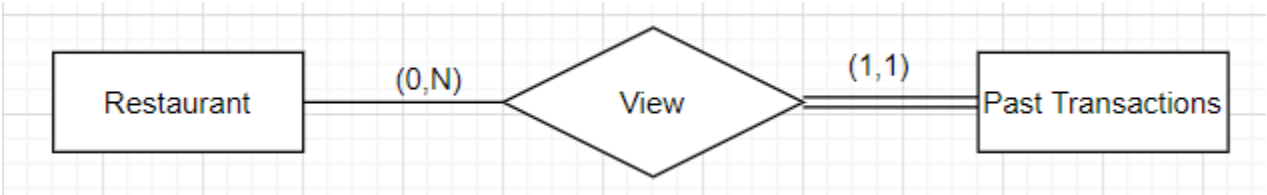
Reviews (Reviews can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Restaurant can have at least 0 relationship with Reviews and at most N relationship).

Participation: 0, Cardinality: N

Relationship 31: View



Relation: Shows the relation between Past Transaction entities and Restaurant entities. This relation is for Restaurants to view their Past Transactions.

Attributes

TransactionID (Foreign Key from Entity1): Primary key attribute on Past Transactions entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

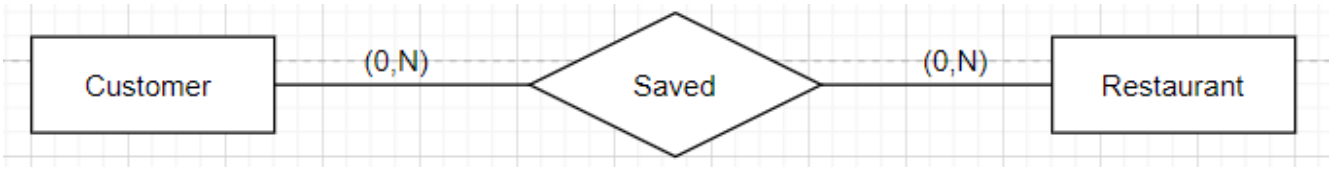
Past Transactions (Past Transaction can have at least 1 relationship with Restaurant and at most 1 relationships).

Participation: 1, Cardinality: 1

Restaurant (Restaurant can have at least 0 relationship with Past Transactions and at most N relationship).

Participation: 0, Cardinality: N

Relationship 32: Saved



Relation: Shows the relation between Customer entities and Restaurant entities. This relation is for Customers to save their preferred Restaurants.

Attributes

CustomerID (Foreign Key from Entity1): Primary key attribute on Customer entity.

RestaurantID (Foreign Key from Entity 2): Primary key attribute on Restaurant entity.

Cardinalities:

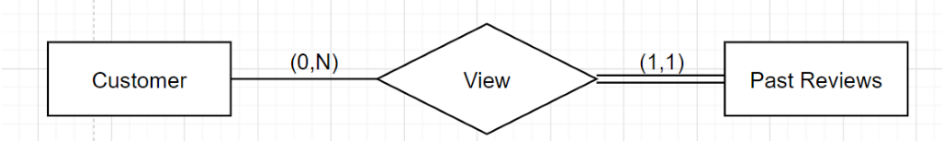
Customer (Customer can have at least 0 relationship with Restaurant and at most N relationships).

Participation: 0, Cardinality: N

Restaurant (Restaurant can have at least 0 relationship with Customer and at most N relationship).

Participation: 0, Cardinality: N

Relationship 33: View



Relation: Shows the relation between Customer entities and Past Reviews entities. This relation is for

Customers to search their Restaurant reviews.

Attributes

CustomerID (Foreign Key from Entity1): Primary key attribute on Customer entity.

ReviewsID (Foreign Key from Entity 2): Primary key attribute on Review entity.

Cardinalities:

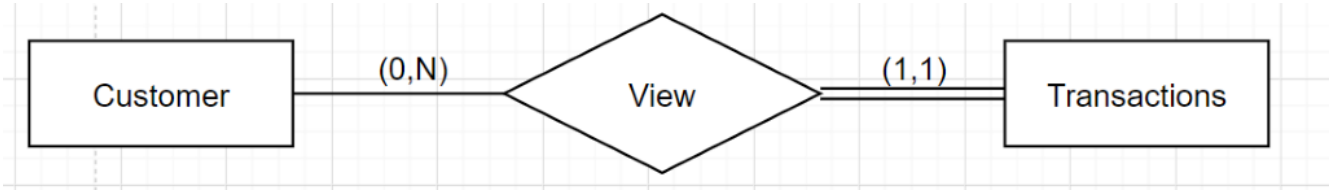
Customer (Customer can have at least 0 relationship with Past Reviews and at most N relationships).

Participation: 0, Cardinality: N

Past Reviews (Past Reviews can have at least 1 relationship with Customer and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 34: View



Relation: Shows the relation between Customer entities and Transactions entities. This relation is for Customers to search their Transactions.

Attributes

CustomerID (Foreign Key from Entity1): Primary key attribute on Customer entity.

TransactionID (Foreign Key from Entity 2): Primary key attribute on Review entity.

Cardinalities:

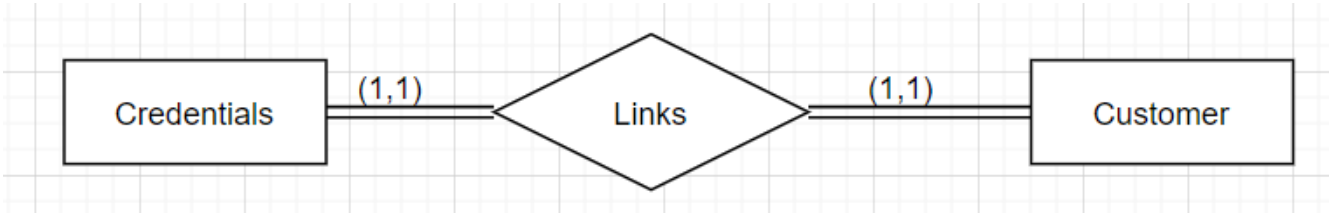
Customer (Customer can have at least 0 relationship with Transactions and at most N relationships).

Participation: 0, Cardinality: N

Transactions (Transactions can have at least 1 relationship with Customer and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 35: Links



Relation: Shows the relation between Customer entities and Credentials entities. This relation is for Customers to save/change their Credentials.

Attributes

CustomerID (Foreign Key from Entity1): Primary key attribute on Customer entity.

UsernameID (Foreign Key from Entity 2): Primary key attribute on Credentials entity.

Cardinalities:

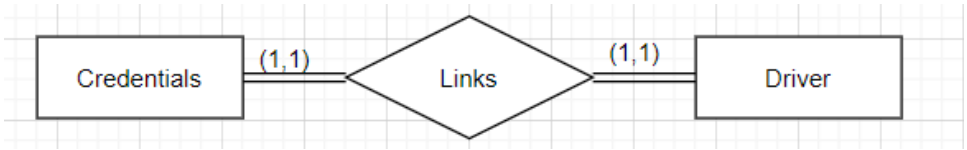
Customer (Customer can have at least 1 relationship with Credentials and at most N relationships).

Participation: 1, Cardinality: 1

Credentials (Credentials can have at least 1 relationship with Customer and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 36: Links



Relation: Shows the relation between Driver entities and Credentials entities. This relation is for Drivers to save/change their Credentials.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

UsernameID (Foreign Key from Entity 2): Primary key attribute on Credentials entity.

Cardinalities:

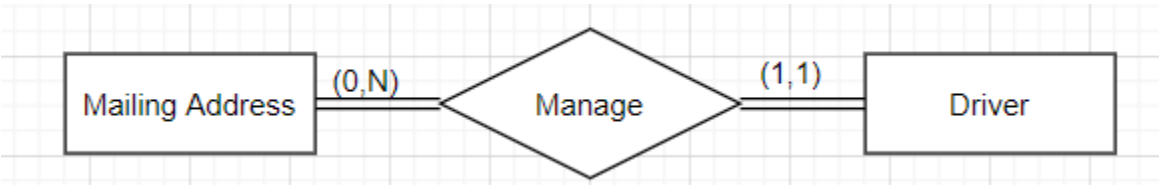
Driver (Driver can have at least 1 relationship with Credentials and at most N relationships).

Participation: 1, Cardinality: 1

Credentials (Credentials can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 37: Manage



Relation: Shows the relation between Driver entities and Mailing Address entities. This relation is for Drivers to save/change their Mailing Address.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

AddressID (Foreign Key from Entity 2): Primary key attribute on Mailing Address entity.

Cardinalities:

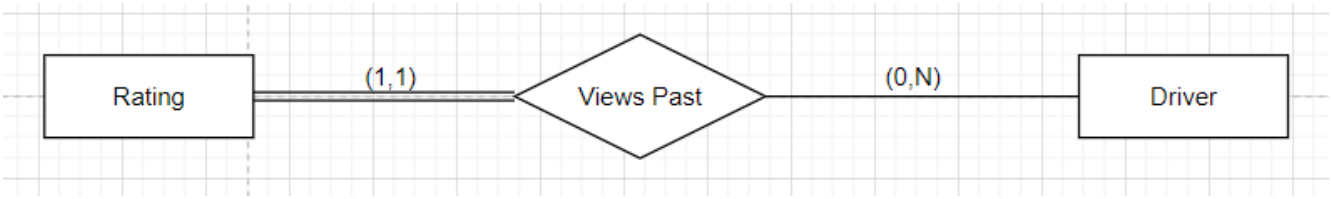
Driver (Driver can have at least 1 relationship with Mailing Address and at most N relationships).

Participation: 1, Cardinality: 1

Mailing Address (Mailing Address can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 0, Cardinality: N

Relationship 38: View Past



Relation: Shows the relation between Driver entities and Rating entities. This relation is for Drivers to search past Rating.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

RatingID (Foreign Key from Entity 2): Primary key attribute on Rating entity.

Cardinalities:

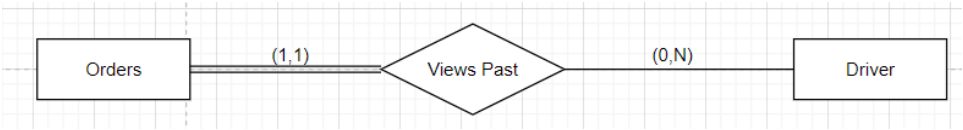
Driver (Driver can have at least 0 relationship with Rating and at most N relationships).

Participation: 0, Cardinality: N

Rating (Rating can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 39: View Past



Relation: Shows the relation between Driver entities and Orders entities. This relation is for Drivers to search past Orders.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

OrderID (Foreign Key from Entity 2): Primary key attribute on Order.

Cardinalities:

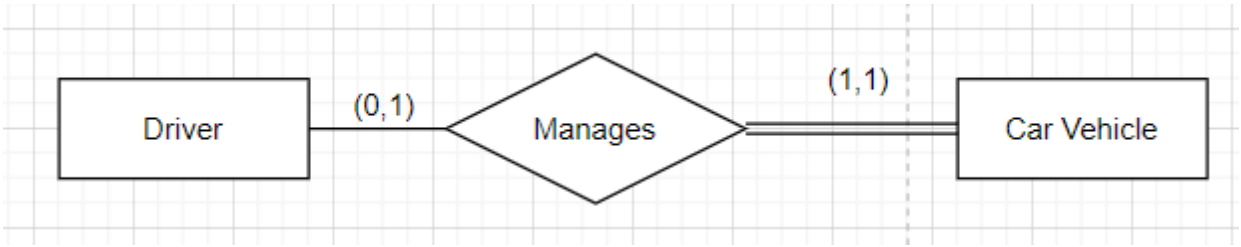
Driver (Driver can have at least 0 relationship with Order and at most N relationships).

Participation: 0, Cardinality: N

Order (Order can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 40: Manages



Relation: Shows the relation between Driver entities and Car Vehicle entities. This relation is for Drivers to Manage their Car Vehicle.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

VIN(Foreign Key from Entity 2): Primary key attribute on Car Vehicle.

Cardinalities:

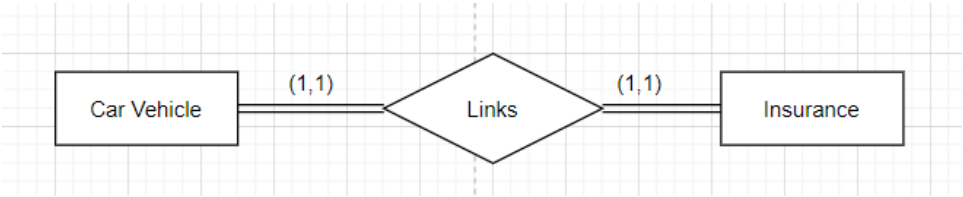
Driver (Driver can have at least 0 relationship with Car Vehicle and at most 1 relationships).

Participation: 0, Cardinality: 1

Car Vehicle (Car Vehicle can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 41: Links



Relation: Shows the relation between Car Vehicle entities and Insurance entities. This relation is for Drivers to Manage their Car Vehicle attached Insurance.

Attributes

InsuranceID (Foreign Key from Entity1): Primary key attribute on Insurance entity.

VIN(Foreign Key from Entity 2): Primary key attribute on Car Vehicle.

Cardinalities:

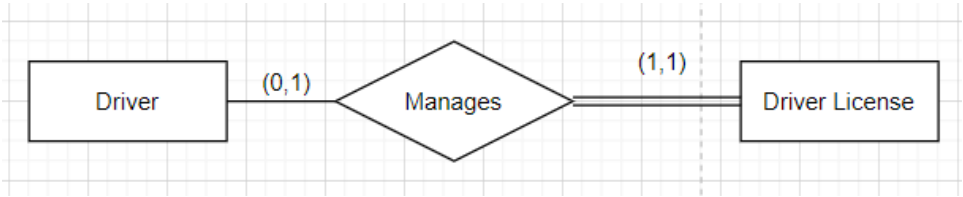
Insurance: (Insurance can have at least 1 relationship with Car Vehicle and at most 1 relationships).

Participation: 1, Cardinality: 1

Car Vehicle (Car Vehicle can have at least 1 relationship with Insurance and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 42: Manages



Relation: Shows the relation between Driver entities and Driver License entities. This relation is for Drivers to Manage their Driver License.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

Driver_LicenseID (Foreign Key from Entity 2): Primary key attribute on Driver License.

Cardinalities:

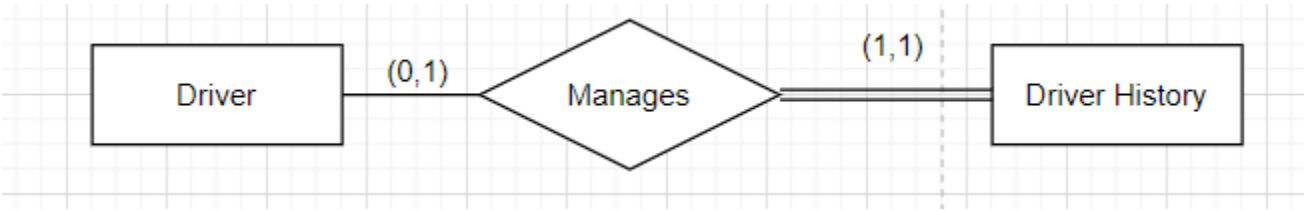
Driver : (Driver can have at least 0 relationship with Driver License and at most 1 relationships).

Participation: 0, Cardinality: 1

Driver License (Driver License can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 43: Manages



Relation: Shows the relation between Driver entities and Driver History entities. This relation is for Drivers to Manage their Driver History.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

Driver_HsitoryID (Foreign Key from Entity 2): Primary key attribute on Driver History.

Cardinalities:

Driver (Driver can have at least 0 relationship with Driver History and at most 1 relationships).

Participation: 0, Cardinality: 1

Driver History (Driver History can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 44: Manages

Relation: Shows the relation between Driver entities and Bank Account entities. This relation is for Drivers to Manage their Bank Account.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

BankAccountID (Foreign Key from Entity 2): Primary key attribute on Bank Account.

Cardinalities:

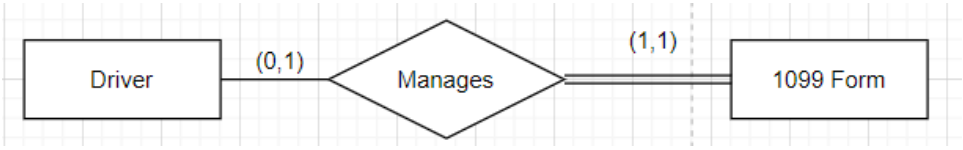
Driver (Driver can have at least 0 relationship with Bank Account and at most 1 relationships).

Participation: 0, Cardinality: 1

Bank Account (Bank Account can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Relationship 45: Manages



Relation: Shows the relation between Driver entities and Bank Account entities. This relation is for Drivers to Manage their 1099 Form.

Attributes

DriverID (Foreign Key from Entity1): Primary key attribute on Driver entity.

SSN (Foreign Key from Entity 2): Primary key attribute on 1099 Form.

Cardinalities:

Driver : (Driver can have at least 0 relationship with 1099 Form and at most 1 relationships).

Participation: 0, Cardinality: 1

1099 Form (1099 Form can have at least 1 relationship with Driver and at most 1 relationship).

Participation: 1, Cardinality: 1

Normalization: 3NF

Restaurant:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.

- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Restaurant_Types:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Address:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Order:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Order_Item:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Order_Review:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Delivery_Review:

- 1NF: Has a primary key: id, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using id (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Menu:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using ID (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Items:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied, only using ID we can get Seasonal employee information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Ingredients:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied, only using ID we can get Full-Time employee information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Payment_Method:

- 1NF: Has a primary key: CreditCard+TimeStamp, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied, only using ID we can get Full-Time employee information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Driver:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using ID (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Credentials:

- 1NF: Has a primary key: Username, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using Username(PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Employee:

- 1NF: Has a primary key: employeeID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using employeeID (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Seasonal:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied, only using ID we can get Seasonal employee information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Part-Time:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied, only using ID we can get Part-Time employee information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Full-Time:

- 1NF: Has a primary key: employeeID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied, only using ID we can get Full-Time employee information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Employee-Productivity:

- 1NF: Has a primary key: employeeID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using employeeID (PK) we can get other attributes information
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Customer:

- 1NF: Has a primary key: ID, all attribute contains atomic value, no multivalued or composite attribute.
- 2NF: 1NF is satisfied; only using ID (PK) we can get other attributes information.
- 3NF: 2NF is satisfied; except PK, no attribute is dependent on another attribute.

Data Dictionary

	B	C	D	E	F
1	Attribute	Data Type	Primary Key	Foreign Key	Constraints
2	id	int	Yes		
3	street	varchar(200)			
4	city	varchar(100)			
5	state	varchar(2)			2 characters
6	zipcode	varchar(20)			5 digits hyphen and 4 digits
7	latitude	decimal(11,8)			
8	longitude	decimal(11,8)			
9	userid	int	Yes		
10	password	varchar(255)			
11	employee_id	int		employee(id)	
12	driver_id	int		driver(id)	
13	customer_id	int		customer(id)	
14	role	enum('manager','employee','customer','driver')			
15	credit_card_id	varchar(255)	Yes		
16	branch	varchar(45)			
17	expiration_month	int			
18	expiration_year	int			
19	id	int	Yes		
20	firstname	varchar(100)			
21	lastname	varchar(100)			
22	email	varchar(45)			
23	phone	varchar(30)			
24	address	int		address(id)	
25	id	int	Yes		
26	order_id	int		order(id)	
27	rate	enum('poor','fair','average','good','excellent')			Poor, Fair, Average, Good, Excellent
28	comments	varchar(500)			
29	timestamp	datetime			Default value = Current Timestamp
30	id	int	Yes		
31	firstname	varchar(100)			
32	lastname	varchar(100)			
33	email	varchar(45)			
34	phone	varchar(30)			
35	id	int	Yes		

	B	C	D	E	F
36	firstname	varchar(100)			
37	lastname	varchar(100)			
38	ssn	varchar(15)			
39	holiday_status	enum('on','off')			
40	phone	varchar(30)			
41	address	int		address(id)	
42	supervisor	int		employee(id)	
43	id	int	Yes		
44	employee_id	int		employee(id)	
45	clocked_in	datetime			
46	clocked_out	datetime			
47	assigned_hours	int			
48	hours_earned	int			
49	goal	int			
50	employee_id	int	Yes	employee(id)	
51	start_date	datetime			
52	end_date	datetime			
53	hourly_wage	decimal(12,4)			
54	weekly_hours	int			
55	id	int	Yes		
56	comments	varchar(45)			
57	uid	varchar(255)			
58	id	int	Yes		
59	allergen	varchar(20)			
60	diet_adherence_tag	varchar(45)			
61	name	varchar(75)			
62	price	decimal(11,8)			
63	size	decimal(11,8)			
64	calories	int			
65	promotion_tag	varchar(45)			
66	unit	enum('kg','oz','g','mg','l')			
67	id	int	Yes		
68	restaurant_id	int		restaurant(id)	
69	title	varchar(100)			
70	content	text			

	B	C	D	E	F
71	enabled	tinyint			
72	id	int	Yes		
73	menu_id	int		menu(id)	
74	ingredient_id	int		ingredient(id)	
75	id	int	Yes		
76	userid	int		credential(id)	
77	driver	int		driver(id)	
78	payment_method	int		payment_method(id)	
79	deliver_to	int		address(id)	
80	created	datetime			Default value = Current Timestamp
81	delivered	datetime			
82	canceled	datetime			
83	status	enum('created','in-progress','delivered','canceled')			created, in-progress, delivered, canceled
84	tip	decimal(12,4)			>=0
85	total	decimal(12,4)			>=0
86	id	int	Yes		
87	order_id	int		order(id)	
88	menu_item	int		menu(id)	
89	quantity	int			>=0
90	total	decimal(12,4)			>=0
91	id	int	Yes		
92	order_id	int		order(id)	
93	rate	enum('poor','fair','average','good','excellent')			Poor, Fair, Average, Good, Excellent
94	comments	varchar(500)			
95	timestamp	datetime			Default value = Current Timestamp
96	employee_id	int	Yes	employee(id)	
97	start_date	datetime			
98	end_date	datetime			
99	hourly_wage	decimal(12,4)			
100	weekly_hours	int			
101	id	int	Yes		
102	userid	int		credential(id)	
103	credit_card_id	varchar(255)		credit_card(credit_card_id)	
104	paypal_id	int		paypal(id)	
105	google_pay_id	int		google_pay(id)	

	B	C	D	E	F
106	id	int	Yes		
107	comments	varchar(45)			
108	uid	varchar(255)			
109	id	int	Yes		
110	type	int		restaurant_type(id)	
111	manager	int		employee(id)	
112	address	int		address(id)	
113	name	varchar(200)			>0
114	phone	varchar(20)			
115	website	varchar(200)			
116	status	enum('open','closed','unavailable')			open, closed, unavailable
117	open	time			hh:mm:ss
118	close	time			hh:mm:ss
119	day_of_week	int			1 = Sunday, 2 = Monday, 4 = Tuesday, 8 = Wednesday, 16 = Thursday, 32 = Friday, 64 = Saturday
120	id	int	Yes		
121	category	enum('Fast food','Fast casual','Casual dining','Premium casual','Family style','Fine dining')			Fast food,Fast casual,Casual dining,Premium casual,Family style,Fine dining
122	description	varchar(200)			
123	employee_id	int	Yes	employee(id)	
124	start_date	datetime			
125	end_date	datetime			
126	hourly_wage	decimal(12,4)			
127	weekly_hours	int			
128					

Implementation

Appendix 1.3

Summary

Teamwork

- Alexandre Geraldo
    - Designed the ER model, Relational model, SQL queries for the following entities: Restaurant, Address, Order, Order Detail, Order Review, Delivery Review, Favorite Order, Favorite Detail, and the respect relationship of the entities.
    - Produced the user requirements related to assigned entities.
    - Worked building the design report.
    - Modified the presentation slides.
    - Proofread and modified the Final report.

- Iffatun Nessa Mahi
    - Designed the ER model Relational model, SQL queries for the following entities: Customer, Loyalty Program, Marketing, Employee, Daily Employee Productivity, and the respect relationship of the entities.
    - Produced the user requirements related to assigned entities.
    - Worked building the design report.
    - Modified the presentation slides
    - Prepared the Final report.

- Kirby Liu
    - Designed the ER model Relational model, SQL queries for the following entities: Driver Management, User Management, Payment Management, and Menu Management, and the respect relationship of the entities.
    - Produced the user requirements related to assigned entities.
    - Worked building the design report.
    - Prepared the presentation slides

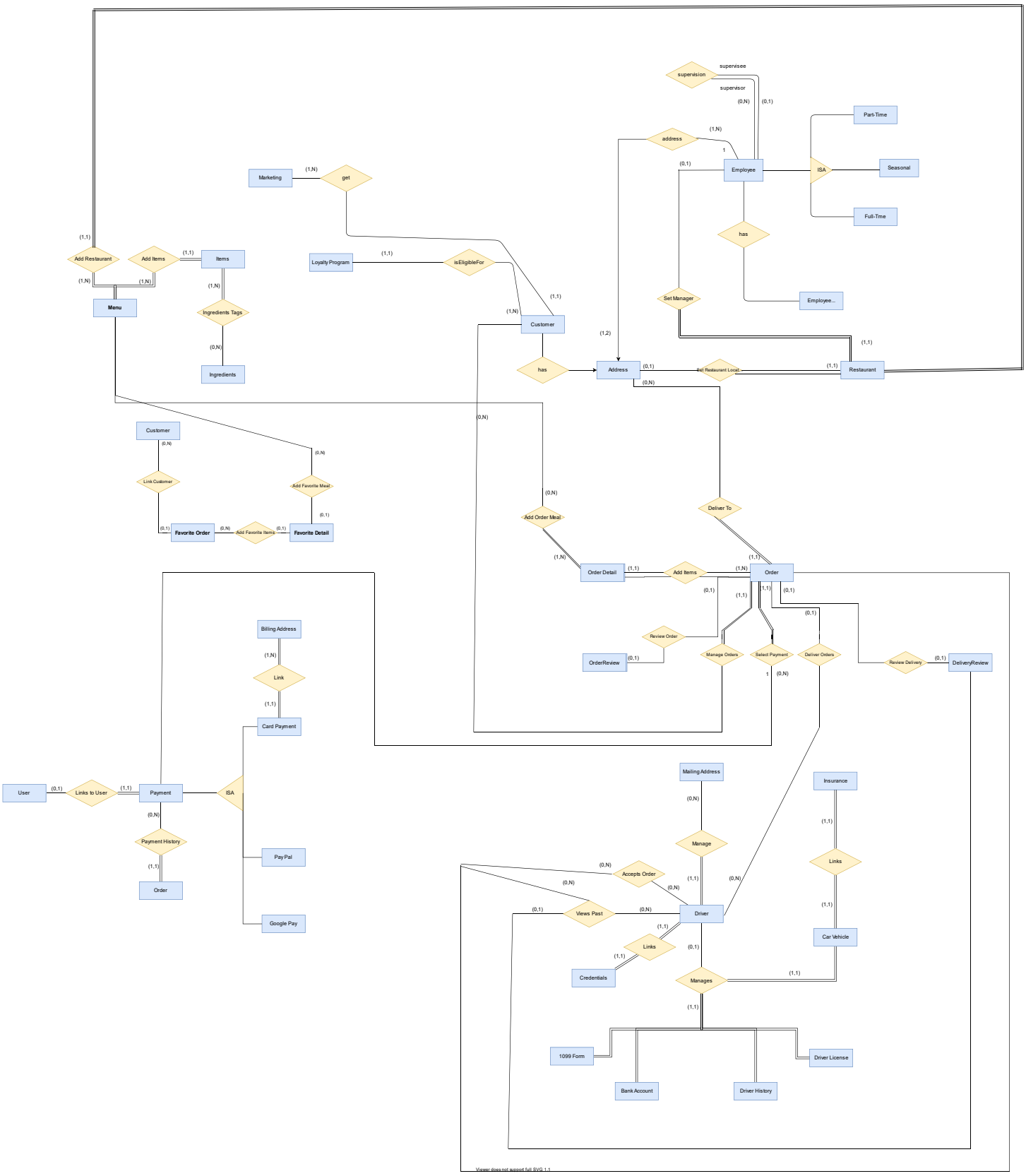
- Proofread and modified the Final report.

Access Link

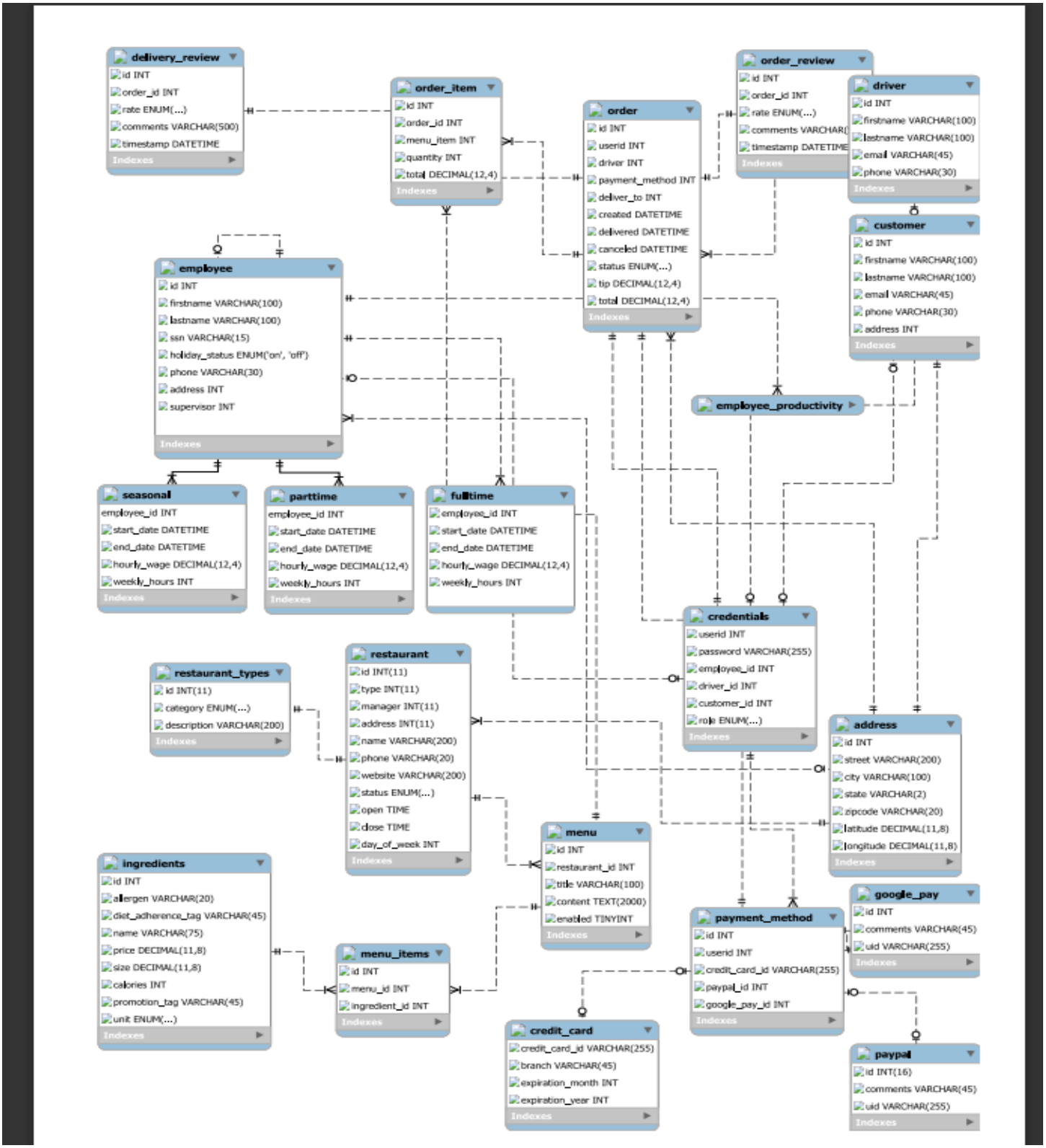
Demo video:
https://transcripts.gotomeeting.com/#/s/a90bf6a9f60491fe5abe1f1bfdca325c9baa60ae5260727e64856e190a4d70b4

Appendix

1.1 ER Model


supervisee
supervision
supervisor
(0,N)
(0,1)
Part-Time
address
(1,N)
1
Employee
(0,1)
ISA
Seasonal
Full-Time
has
Set Manager
Employee...
(1,1)
Restaurant
Set Restaurant Locati...
Address
(0,1)
(0,N)
(1,2)
has
Customer
(1,N)
(1,1)
Loyalty Program
isEligibleFor
Marketing
(1,N)
get
(1,1)
Add Restaurant
Add Items
(1,1)
Items
(1,N)
(1,N)
Menu
(1,N)
Ingredients Tags
(0,N)
Ingredients
Customer
(0,N)
Link Customer
(0,1)
Favorite Order
(0,N)
Add Favorite Items
(0,1)
Favorite Detail
(0,1)
Add Favorite Meal
(0,N)
(0,N)
Add Order Meal
(1,N)
Order Detail
(1,1)
Add Items
(1,N)
Order
(1,1)
Deliver To
(1,1)
(0,1)
(1,1)
(1,1)
(0,1)
(0,1)
Review Order
OrderReview
(0,1)
Manage Orders
Select Payment
Deliver Orders
1
(0,N)
Review Delivery
(0,1)
DeliveryReview
Billing Address
(1,N)
Link
(1,1)
Card Payment
User
(0,1)
Links to User
(1,1)
Payment
ISA
(0,N)
Payment History
(1,1)
Order
Pay Pal
Google Pay
Mailing Address
(0,N)
Manage
(1,1)
Accepts Order
(0,N)
(0,N)
(0,N)
(1,1)
Views Past
(0,1)
(0,N)
Driver
(1,1)
Links
(1,1)
Credentials
(0,1)
Manages
(1,1)
(1,1)
Insurance
(1,1)
Links
(1,1)
Car Vehicle
1099 Form
Bank Account
Driver History
Driver License
Viewer does not support full SVG 1.1

1.2 Relational Model



1.3 SQL code

/* User Requirement 1: The system will allow customers to add orders, verify orders status, cancel orders.*/

```
INSERT INTO `order`(userid, driver, payment_method, deliver_to, created, status) VALUES (userid, driver, payment_method, deliver_to, NOW(), 'created');

 INSERT INTO `order_item`(order_id, menu_item, quantity, total) VALUES (order_id, menu_item, quantity, total);

 -- canceling order

UPDATE `order` SET canceled = NOW(), status= 'canceled' WHERE id = order_id

 -- verify order status

SELECT status FROM `order` WHERE id = :order_id
```

/* User Requirement 2: The system will allow drivers to pick orders from a queue and delivering to customers.*/

```
SELECT o.id, a.street, a.city, a.state, a.zipcode, a.latitude, a.longitude

  FROM `order` o, address a

      WHERE o.status='created'

        AND o.deliver_to = a.id

ORDER BY o.created

LIMIT 1;
```

/* User Requirement 3: The system will provide a global address catalog to avoid duplicate records added to the system. The address catalog will also provide latitude and longitude coordinates to be used in data visualizations. */

```
SELECT a.street, a.city, a.state, a.zipcode, a.latitude, a.longitude FROM address a;
```

/* User Requirement 4: The system will allow the restaurants to specify the operation hours of the business, select a local manager, and implement the use of the global address catalog. A restaurant must also have a feature to enable or disable the ordering system as a whole.*/

```
 -- Operation

INSERT INTO restaurant (type, manager, address, name, phone, website, status, open, close,
```

```
day_of_week)
                VALUES (type, manager, address, name, phone, website, status, open, close,
day_of_week);
```

/*User Requirement 5: The system will allow customers to evaluate orders and deliveries.*/

-- Order review

```
INSERT INTO `order_review`(order_id, rate, comments) VALUES (order_id, rate, comments);
INSERT INTO `delivery_review`(order_id, rate, comments) VALUES (order_id, rate, comments);
```

/* User Requirement 6: Restaurants will be able to build their menu offering while listing calories, prices, size, promotions, and food category of their food and drink offering. */

-- add menu

```
INSERT INTO `menu`(restaurant_id, title, content, enabled) VALUES (restaurant_id, title, content,
enabled);
```

-- adding menu_items

```
INSERT INTO `menu_items`(menu_id, ingredient_id) VALUES (menu_id, ingredient_id);
```

-- adding ingredients

```
INSERT INTO `ingredients`(allergen, diet_adherence_tag, name, price, size, calories,
promotion_tag,unit) VALUES (allergen, diet_adherence_tag, name, price, size, calories,
promotion_tag,unit);
```

-- apply promotion price

```
UPDATE `ingredients` SET price= price where promotion_tag = promotion_tag;
```

/* User Requirement 7: Restaurants will have the ability to list the ingredients of each food and tag with special dietary and/or common food allergens.

--adding ingredients */

```
INSERT INTO `ingredients`(ingredients_id, allergen, ingredients_name, diet_adherence_tag) VALUES
(ingredients_id, allergen, ingredients_name, diet_adherence_tag);
```

-- check allergy of item

```
 select i.name, i.allergen fROM menu_items m, ingredients i Where m.menu_id = menu_id AND
m.allergen = allergen;

 -- search for items that do adhere to user diet

 select i.name, i.diet_adherence_tag fROM menu_items m, ingredients i Where m.menu_id = menu_id
AND   m.allergen != diet_adherence_tag;


/* User Requirement 8: Customers will be able to save preferred payment options of credit/debit cards
(with billing address), paypal, and/or google pay. */

 -- adding payment_method

INSERT INTO `payment_method`(user_id,credit_card_id,paypal_id,google_pay_id) VALUES
(user_id,credit_card_id,paypal_id,google_pay_id);

 -- adding credit_card

INSERT INTO `credit_card`(branch,expiration_month,expiration_year) VALUES
(branch,expiration_month,expiration_year);

 -- adding paypal

INSERT INTO `paypal`(comments,u_id) VALUES (comments,u_id);

 -- adding google_pay

INSERT INTO `google_pay`(comments,u_id) VALUES (comments,u_id);


/* User Requirement 9: The system will allow customers/restaurants/drivers to input and store their
credentials, password, and role. */

 -- adding credentials

INSERT INTO `credentials`(password,employee_id,driver_id,customer_id,role) VALUES
(password,employee_id,driver_id,customer_id,role);


/* User Requirement 10: The system will allow customers/restaurants/drivers to view their past orders
and transaction receipts.

     Driver Dashboard */

 -- Driver Selection

SELECT CONCAT(CONCAT(id,'-'), CONCAT(CONCAT(firstname, ' '),lastname)) name   from driver

 -- Ready to Pickup
```

```
SELECT COUNT(1) FROM `order` WHERE driver='${driver}' AND status='created' AND created
BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)

 -- In Progress

SELECT COUNT(1) FROM `order` WHERE driver='${driver}' AND status='in-progress' AND
created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)

 -- Delivered

SELECT COUNT(1) FROM `order` WHERE driver='${driver}' AND status='delivered' AND created
BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)

 -- Canceled

SELECT COUNT(1) FROM `order` WHERE driver='${driver}' AND status='canceled' AND created
BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)

 -- Pickup Activity Map

SELECT a.latitude, a.longitude,

        CONCAT(CONCAT(cu.firstname, ' '),cu.lastname) name,

        TIMESTAMPDIFF(MINUTE,NOW(), o.created) value

  FROM `order` o, address a, credentials c, customer cu

     WHERE o.driver=:driver

       AND o.status='created'

       AND o.deliver_to = a.id

       AND o.userid = c.userid

       AND c.customer_id = cu.id

       AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)

  -- Poor Service Evaluation

SELECT COUNT(1)

  FROM `delivery_review` dr, `order` o

     WHERE dr.order_id = o.id

       AND dr.rate     = 'poor'

       AND o.driver      = :driver

       AND o.created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
```

```
-- Fair Service Evaluation
SELECT COUNT(1)
  FROM `delivery_review` dr, `order` o
    WHERE dr.order_id = o.id
      AND dr.rate     = 'fair'
      AND o.driver    = :driver
      AND o.created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- Average Service Evaluation
SELECT COUNT(1)
  FROM `delivery_review` dr, `order` o
    WHERE dr.order_id = o.id
      AND dr.rate     = 'average'
      AND o.driver    = :driver
      AND o.created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- Good Service Evaluation
SELECT COUNT(1)
  FROM `delivery_review` dr, `order` o
    WHERE dr.order_id = o.id
      AND dr.rate     = 'good'
      AND o.driver    = :driver
      AND o.created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- Excellent Service Evaluation
SELECT COUNT(1)
  FROM `delivery_review` dr, `order` o
    WHERE dr.order_id = o.id
      AND dr.rate     = 'excellent'
      AND o.driver    = :driver
```

```
        AND o.created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)

/* Manager Dashboard */
-- Order Amount Summary Statistics
 -- minimum
SELECT MIN(total) as minimum
    FROM `order`
        WHERE status = 'delivered'
          AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- average
SELECT AVG(total) as average
    FROM `order`
        WHERE status = 'delivered'
          AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- total
SELECT SUM(total) as total
    FROM `order`
        WHERE status = 'delivered'
          AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- maximum
SELECT MAX(total) as maximum
    FROM `order`
        WHERE status = 'delivered'
          AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- stddev
SELECT STDDEV(total) as stddev
    FROM `order`
        WHERE status = 'delivered'
```

```
        AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- Profit x Loss
SELECT (profit - loss)
FROM (SELECT SUM(total) profit
        FROM `order`
          WHERE status='delivered'
            AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
      ) as a,
      (SELECT SUM(total) loss
        FROM `order`
          WHERE status='canceled'
            AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
      ) as b
 -- Total Sales
SELECT COUNT(1) FROM `order` WHERE status='delivered' AND created BETWEEN
FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- Total Loss
SELECT COUNT(1) FROM `order` WHERE status='canceled' AND created BETWEEN
FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
 -- Preferred Payment Method
SELECT UNIX_TIMESTAMP(NOW()) as 'time_sec', a.*, b.*, c.*
FROM (SELECT COUNT(1) as 'Credit Card'
        FROM `order` o, payment_method p
          WHERE o.payment_method = p.id
            AND p.credit_card_id IS NOT NULL
            AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
      ) as a,
      (SELECT COUNT(1) as 'Google Pay'
        FROM `order` o, payment_method p
```

```
            WHERE o.payment_method = p.id
              AND p.google_pay_id IS NOT NULL
              AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
        ) as b,
        (SELECT COUNT(1) as 'PayPal'
            FROM `order` o, payment_method p
              WHERE o.payment_method = p.id
              AND p.paypal_id IS NOT NULL
              AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
         ) as c
 -- Order Activity
SELECT o.id as `#Order`, CONCAT(CONCAT(cu.firstname, ' '), cu.lastname) as Customer,
          CONCAT(CONCAT(d.firstname, ' '), d.lastname) as Driver, Status, Total
     FROM `order` o, credentials c, customer cu, driver d
         WHERE o.userid = c.userid
           AND c.customer_id = cu.id
           AND o.driver = d.id
            AND created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
ORDER BY 1
 -- Order Frequency
SELECT
   UNIX_TIMESTAMP(created) DIV 3600 * 3600 AS "time",
   count(id) AS "id"
FROM `order`
WHERE
   created BETWEEN FROM_UNIXTIME(:start) AND FROM_UNIXTIME(:end)
GROUP BY 1
ORDER BY UNIX_TIMESTAMP(created) DIV 3600 * 3600
```

/* User requirement 11: Insert, Update and retrieve employee information */

-- Insertion to the Employee table:

```
INSERT INTO `employee` (id, firstname, lastname, ssn, holiday_status, phone, address, supervisor) VALUES (id, firstname, lastname, 'on', phone, address, supervisor);
```

-- Insertion to any (Seasonal/Part-time/Full-time) table:

```
INSERT INTO `fulltime` (employee_id, start_date, end_date, hourly_wage, weekly_hours) VALUES (employee_id, "2020-11-24 00:00:00", "2021-11-24 00:00:00", hourly_wage, weekly_hours);
```

-- Update any table: (employee & fulltime)

```
UPDATE employee, fulltime
SET firstname = firstname, hourly_wage = hourly_wage
WHERE id = employee_id AND employee_id = id ;
```

-- Find information of an Employee using FK

```
SELECT employee_id AS Seasonal_ID, firstname AS FirstName, id AS Employee_ID, holiday_status AS Holiday, supervisor AS Manager_ID, hourly_wage AS Wage
FROM employee, seasonal
WHERE id = employee_id;
```

/* User requirement 12: Insert & retrieve Employee Productivity info */

```
INSERT INTO employee_productivity SET employee_id = 1, clocked_in = time , clocked_out = time , work_date = date, assigned_hours = assigned_hours, hours_earned = TIMESTAMPDIFF(HOUR, clocked_in, clocked_out),
goal = CASE WHEN TIMESTAMPDIFF(HOUR, clocked_in, clocked_out) < assigned_hours THEN 0
WHEN TIMESTAMPDIFF(HOUR, clocked_in, clocked_out) > assigned_hours THEN 2
ELSE 1 END;
```

--Find employees goal using their id

```
SELECT e.firstname AS NameOfEmployee, ep.goal As Weekly_Goal
FROM employee e, employee_productivity ep
WHERE e.id = ep.employee_id ;
```

/* User requirement 13: Insert & retrieve Customer information */

```
INSERT INTO customer SET firstname = firstname, lastname = lastname, email = email, phone = phone, address = address;
```

-- Find the customers who has same address

```
SELECT firstname, lastname, phone
FROM customer
WHERE address = address;
```

/*User requirement 14: Find potential customer from their total amount and add 5% Discount */

```
SELECT o.`id` AS OrderID, oi.total AS BeforeDiscount, oi.total-(oi.total*5/100) As AfterDiscount
FROM order_item oi, `order` o
WHERE oi.total > 100
AND o.id = oi.order_id ;
```

/*User requirement 15: Lucky Day Discount [if the current date is divided by 5 and the result is less than 2 and total >= 110 then the customer gets extra 4% discount] */

```
SELECT o.id AS OrderID, oi.total AS BeforeDiscount, oi.total-(oi.total*5/100) As AfterDiscount,
(oi.total-(oi.total*5/100))-((oi.total-(oi.total*5/100))*4/100) As LuckyDiscount
FROM order_item oi, `order` o
WHERE dayofmonth(curdate())/5 < 2 AND oi.total >= 110 AND o.id = order_id;
```