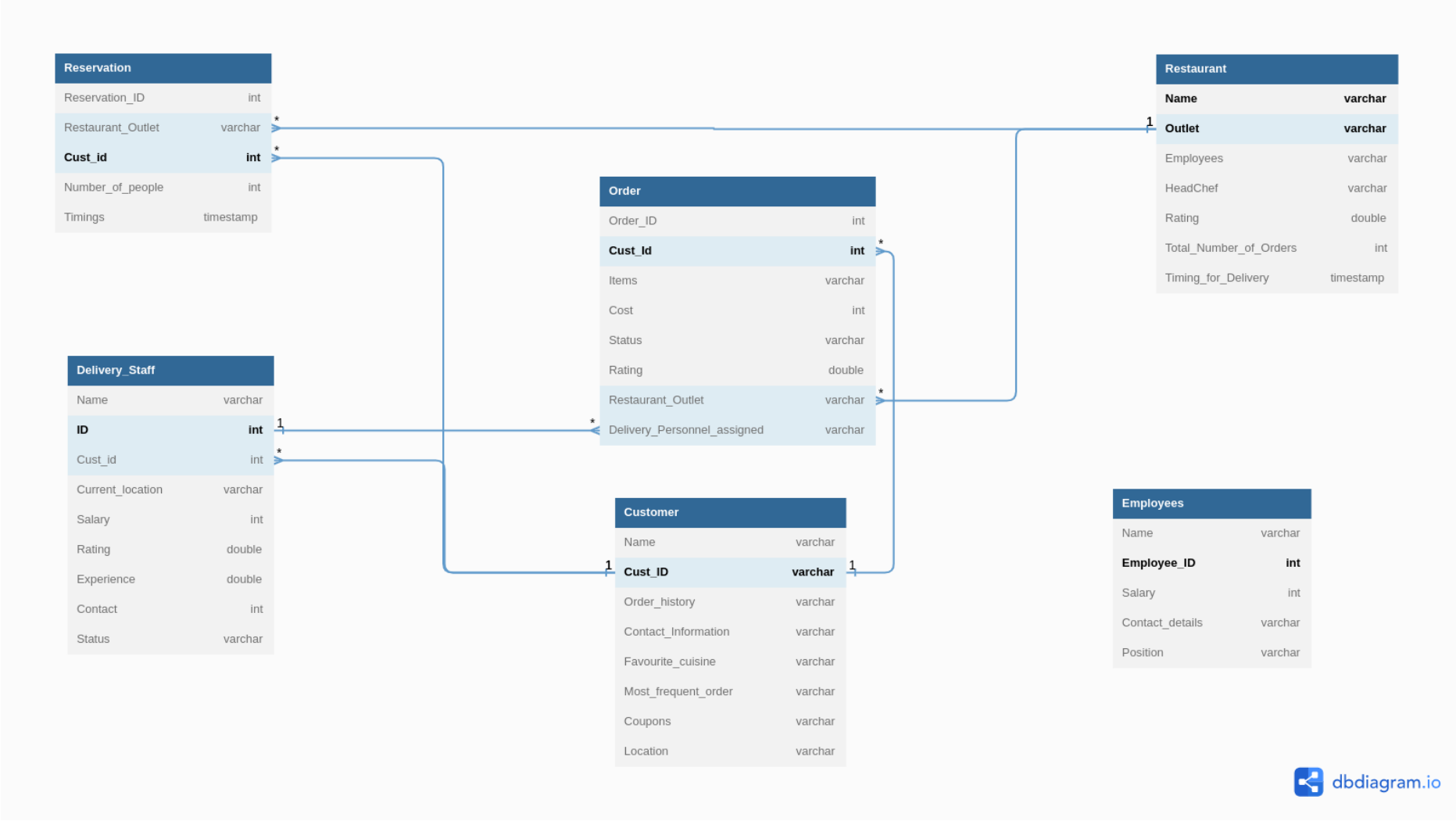
DNA PROJECT PHASE 3

TEAM 34

Talib Siddiqui: 2021101078
Vanshita Mahajan: 2021101102
Khushi Wadhwa: 2021101104

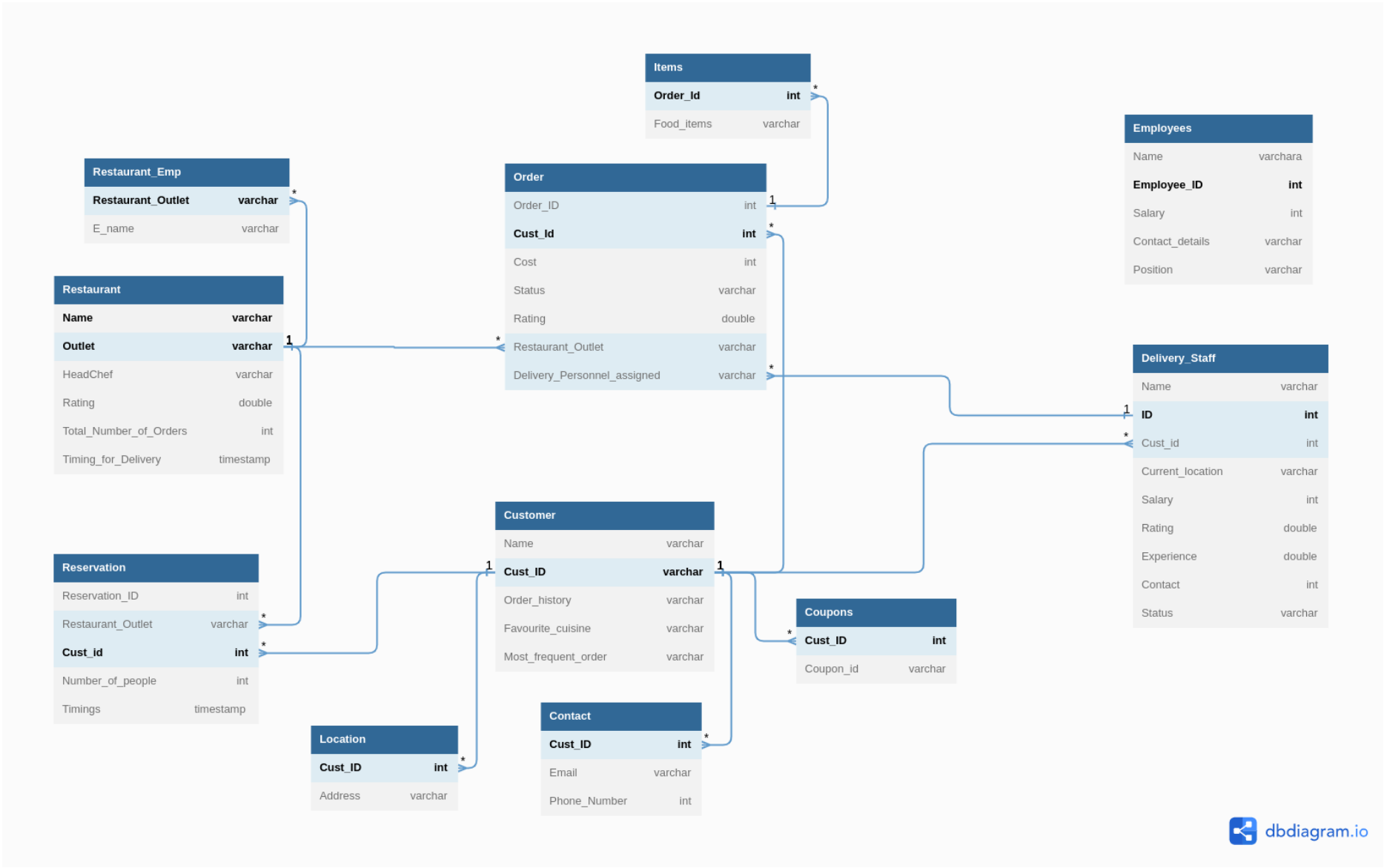
Relational Model:



Clarifications:

1. In the Relation **Restaurant,** both **Name** and **Outlet** make a composite Primary key but all references are made to Outlet only. This is due to limitation of the tool, kindly consider this as a reference to Name and Outlet combined.

1NF


Items
Order_Id int
Food_items varchar
Employees
Name varchara
Employee_ID int
Salary int
Contact_details varchar
Position varchar
Restaurant_Emp
Restaurant_Outlet varchar
E_name varchar
Order
Order_ID int
Cust_Id int
Cost int
Status varchar
Rating double
Restaurant_Outlet varchar
Delivery_Personnel_assigned varchar
Restaurant
Name varchar
Outlet varchar
HeadChef varchar
Rating double
Total_Number_of_Orders int
Timing_for_Delivery timestamp
Delivery_Staff
Name varchar
ID int
Cust_id int
Current_location varchar
Salary int
Rating double
Experience double
Contact int
Status varchar
Customer
Name varchar
Cust_ID varchar
Order_history varchar
Favourite_cuisine varchar
Most_frequent_order varchar
Reservation
Reservation_ID int
Restaurant_Outlet varchar
Cust_id int
Number_of_people int
Timings timestamp
Coupons
Cust_ID int
Coupon_id varchar
Location
Cust_ID int
Address varchar
Contact
Cust_ID int
Email varchar
Phone_Number int
dbdiagram.io

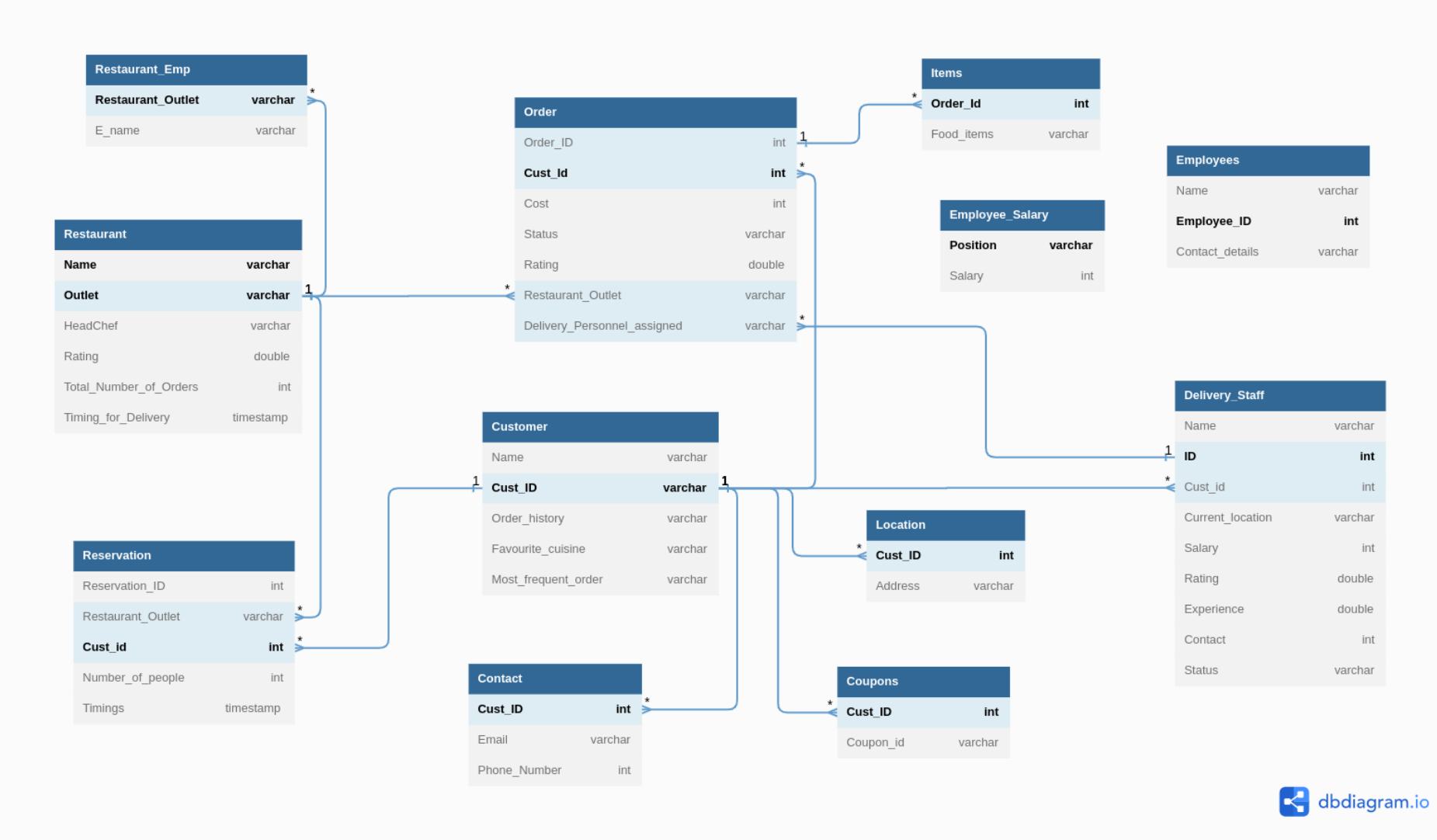
1. In 1NF we decomposed the attribute **"Contact_Information"** in the relation **Customer** into 2 attributes – **Email** and **Phone_Number** in a separate table **Contact** so that 1NF constraints are followed (1NF does not allow for composite attributes)

2. To account for multi-valued attributes 4 new tables have been made after removing them from respective relations:
   Coupons (from Customer)
   Location (from Customer)
   Restaurant_Emp (from Restaurant)
   Items (from Order)
   because these are multi-valued attributes and 1NF does not allow multi-valued attributes.

2NF

The above schema for 1NF also holds for 2NF.
There is only one composite primary key in the database – for the table **Restaurant** (i.e. **Name** + **Outlet**).
Here, the fully functionally dependent constraint is satisfied since neither Name nor Outlet can independently determine any of the other attributes in the relation **Restaurant.**

3NF


Restaurant_Emp
Restaurant_Outlet varchar
E_name varchar
Restaurant
Name varchar
Outlet varchar
HeadChef varchar
Rating double
Total_Number_of_Orders int
Timing_for_Delivery timestamp
Order
Order_ID int
Cust_Id int
Cost int
Status varchar
Rating double
Restaurant_Outlet varchar
Delivery_Personnel_assigned varchar
Items
Order_Id int
Food_items varchar
Employee_Salary
Position varchar
Salary int
Employees
Name varchar
Employee_ID int
Contact_details varchar
Delivery_Staff
Name varchar
ID int
Cust_id int
Current_location varchar
Salary int
Rating double
Experience double
Contact int
Status varchar
Customer
Name varchar
Cust_ID varchar
Order_history varchar
Favourite_cuisine varchar
Most_frequent_order varchar
Location
Cust_ID int
Address varchar
Reservation
Reservation_ID int
Restaurant_Outlet varchar
Cust_id int
Number_of_people int
Timings timestamp
Contact
Cust_ID int
Email varchar
Phone_Number int
Coupons
Cust_ID int
Coupon_id varchar
dbdiagram.io

In the Table **Employees**:

Attribute **Position** depends on Primary Key **ID** and Attribute **Salary** depends on **Position.**

Here X is ID , Y is Salary and Z is Position such that X->Z and Z->Y holds.

We have removed this transitive dependency by creating another Table Employee_Salary to account for this