POINT OF SALE SYSTEM

1) Question
   To streamline the sales and operations at a Bakery by implementing a Point of Sale System.

2) Working with data/model
   I created my own Dataset involving Tables: Customer, Employee, Orders, Product, Product Type, Order Details and Billing and inserting random values to these tables through ***MySQL Workbench.***

3) Conclusion
   Point of Sale system can be implemented at a bakery replacing the pen and paper working methodologies to better organize the customer and sales data. This will help the management of the bakery to keep track of all the sales, employee performance, consumer behavior, product performance and plan their marketing better.
   Management of data will help to fetch all records which can be analyzed further to take wiser business decisions for the management.

FINAL REPORT

Topic: **Point of Sale (POS) System at a Bakery**

Date: **15th December 2017**

Course: **Business Data Management**

Tool Used: **MySQL Workbench**

Submitted to:

Rutgers University, Newark, NJ

PROJECT DESCRIPTION

What is a Point of Sale (POS) System?

- Ask any retailer, "What is a point of sale system?" and they'll tell you—it's the central component of their business; the hub where everything merges.
- A point of sale system is a combination of software and hardware that allows merchants to take transactions and simplify key day-to-day business operations.
- Modern POS systems do more than just offer flexibility when processing daily transactions. "They improve a merchant's chances of success by providing them with tools to streamline business processes.
- Many of these vital processes would be overly tedious and resource exhaustive to complete without the support of a modern POS system.
- Even with all the benefits, analysis have shown that 84 percent of single-store bakery retailers aren't using a POS system. These merchants are using a combination of manual methods, cash registers and, in some cases, Excel for bookkeeping.

What are the problems in the existing Bakery Setup?

- In present time, owners spend most of the time managing orders through conventional ways of pen or paper or a semi conventional way of cash register receipts which in turn calls for too much of a hassle in compiling these pieces of paper.
- In the business terminology, it is said that "Customer is King" and Kings need to be addressed in the best way possible. The non-POS enabled bakery outlets lose a huge chunk of business without maintaining records of their repeating customers.
- The sales in a store are not monitored, therefore, the performance of the employees is hard to administer.

What is the Business Scope of implementing POS at Bakery?

- Sales records at a bakery are better handled through POS and this allows the owners or management to increase the inventory of the better selling cakes and donuts while look into the matter of why the other products are having lesser sales.

- ❖ Customer retention is very important for any business. By capturing customer details will allow a bakery to offer them special discounts or gift vouchers to increase customer loyalty.

- ❖ Having a record of the employees who are taking order form customers will enable the Bakery to establish healthy competition amongst the salesman/employees, thus, increasing bakery sales.

DATABASE STRUCTURE

The database contains 7 tables which includes:

Main Tables:

- ➢ Customer
- ➢ Billing
- ➢ Order
- ➢ Employee
- ➢ Product

Transactional Tables:

- ➢ Order Details
- ➢ Prod Type

DATABASE TABLE	DESCRIPTION
Main Table	
Customer	Records all the information about the customer who will place the order.
Billing	Records all the information about invoice, customer, order, Net amount of the order and employee who will take the order.
Order	Records all the information about order, customer, employee, Date and time of placing the order.
Employee	Records all the information about employee who will take the order and Personal details of the employee.
Product	Records all the information about the type of Product selling in the bakery.
Transactional Table	
Order_Details	Records all the information about the order details which includes quantity of product, order Description and product which will be included in the order.
Prod_Type	Records all the information about the product type description, Product code, measurement of the product and price of the product.

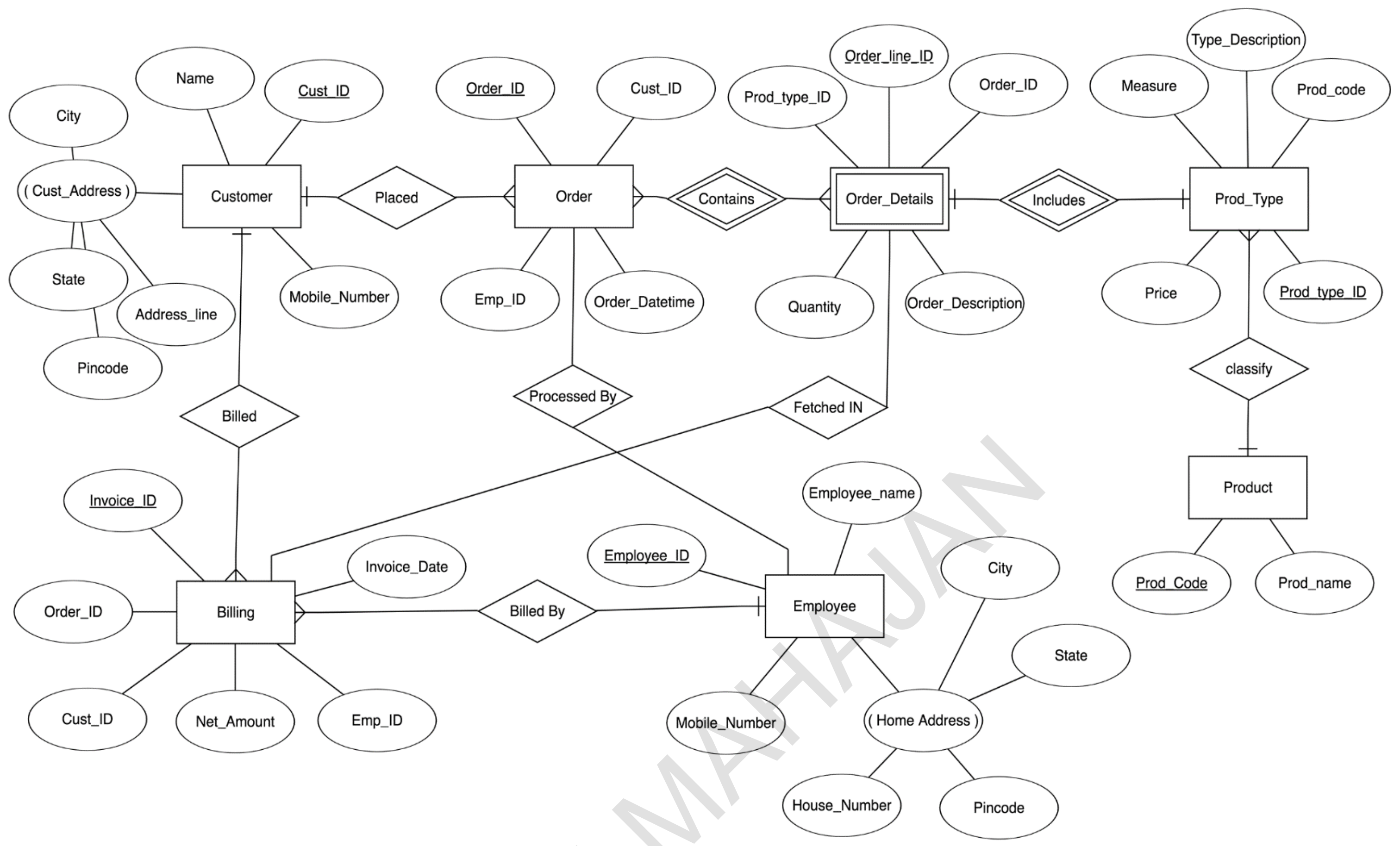
DATABASE DESIGN PROCESS

Our approach for the database project was to take a real-time problem and build a solution for it. We visited a few eating joints and bakeries and witnessed that the processes are totally messed up and the productivity of their stores is in a bad condition. The customers are decreasing, employees are increasing in number and order management is in a bad state. We talked to the owner of one of the bakeries to discuss the working process in the bakery and the problems they were facing currently.

We created seven tables which mainly constructed the entire work flow inside the bakery starting from placing the order to billing the customer for that order. We also tried to avoid any kind of redundancy that may arise in the system. Also, we introduced employee table in this system to include security and transparency features in the system which is usually missing in the non-POS systems. The scope for further tabular inclusions has also been kept in the design for example 'Delivery' table is not included in the design right now but may be included in the advanced designs at later stages.

APPENDICES

Appendix A: E-R DIAGRAM



Explanation of E-R Diagram

Customer will enter the bakery Shop and order the product from the various products available in the shop. The employee of the bakery will ask for the information about the customer which will be recorded in Customer Table. The information about the products will be present in the Product Table like cakes, cupcakes and Donuts. The information about the measurement of the product such as size and price is available in the Prod Type Table. The employee of the bakery who will be placing the order already has the information about him/her in the Bakery system. New order will be created in the system whose information will be in the Order Table and the details of the order will be recorded in the Order Details Table which additionally includes type of product, quantity of the product and order description. Finally, the employee who took the order will send all the order details to the Billing which includes order id, customer id, Net amount, employee id and generate new invoice id for the order. In this way, the whole system will work after the implementation.

Appendix B: DATA DICTIONARY

Customer:

Table Name	CUSTOMER
Description	Stores information about the Customer
Primary Key	CUST_ID

Sr. #	Field Name	Data Type	Constraint	Description
1	CUST_ID	CHAR(35)	Primary key	Customer ID
2	CUST_NAME	TEXT	Not Null	Name of the customer
3	MOBILE_NO	BIGINT(11)	Not Null	Customer's Contact num
4	ADDRESS_LINE	CHAR(50)		Customer's Address
5	CITY	CHAR(20)		Customer's City
6	STATE	CHAR(20)		Customer's State
7	PINCODE	INT(5)		Customer's Pin Code

SQL Query:

```
CREATE TABLE CUSTOMER (CUST_ID CHAR(35) NOT NULL,

CUST_NAME TEXT NOT NULL,

MOBILE_NO BIGINT(11) NOT NULL,

ADDRESS_LINE CHAR(50),

CITY CHAR(20),

STATE CHAR(20),

PINCODE INT(5),

PRIMARY KEY (CUST_ID));
```

Employee:

Table Name	EMPLOYEE
Description	Stores information about the Employee
Primary Key	EMP_ID

Sr. #	Field Name	Data Type	Constraint	Description
1	EMP_ID	CHAR(35)	Primary key	Employee ID
2	EMP_NAME	TEXT	Not Null	Name of the customer
3	MOBILE_NO	BIGINT(11)	Not Null	Employee’s Contact num
4	ADDRESS_LINE	CHAR(50)		Employee’s Address
5	CITY	CHAR(20)		Employee’s City
6	STATE	CHAR(20)		Employee’s State
7	PINCODE	INT(5)		Employee’s Pin Code

SQL Query:

```
CREATE TABLE EMPLOYEE (

EMP_ID CHAR(35) NOT NULL,

EMP_NAME TEXT NOT NULL,

MOBILE_NO BIGINT(11) NOT NULL,

ADDRESS_LINE CHAR(50),

CITY CHAR(20),

STATE CHAR(20),

PINCODE INT(5),

PRIMARY KEY (EMP_ID));
```

Orders:

Table Name	ORDERS
Description	Stores information about the Orders
Primary Key	ORDER_ID
Foreign Key(s)	CUST_ID, EMP_ID

Sr. #	Field Name	Data Type	Constraint	Description
1	ORDER_ID	CHAR(35)	Primary key	Order ID
2	CUST_ID	CHAR(35)	Foreign key	Referring Customer table
3	EMP_ID	CHAR(35)	Foreign key	Referring Employee table
4	ORDER_DT_TIME	DATETIME	Not Null	Order date and time

SQL Query:

```
CREATE TABLE ORDERS (
   ORDER_ID CHAR(35) NOT NULL,
   CUST_ID CHAR(35) NOT NULL,
   EMP_ID CHAR(35) NOT NULL,
   ORDER_DT_TIME DATETIME NOT NULL,
   PRIMARY KEY (ORDER_ID),
   FOREIGN KEY (CUST_ID)
      REFERENCES CUSTOMER (CUST_ID)
      ON DELETE CASCADE ON UPDATE RESTRICT,
   FOREIGN KEY (EMP_ID)
      REFERENCES EMPLOYEE (EMP_ID)
      ON DELETE CASCADE ON UPDATE RESTRICT);
```

Product:

Table Name	ORDERS
Description	Stores information about the Products
Primary Key	PROD_CODE

Sr. #	Field Name	Data Type	Constraint	Description
1	PROD_CODE	CHAR(15)	Primary key	Product Code
2	PROD_NAME	CHAR(25)	Not Null	Product Name

SQL Query:

```
CREATE TABLE PRODUCT (
   PROD_CODE CHAR(15) NOT NULL,
   PROD_NAME CHAR(25) NOT NULL,
   PRIMARY KEY (PROD_CODE));
```

Product Type:

Table Name	PRODUCT_TYPE
Description	Stores information about the Product Type
Primary Key	PROD_TYPE_ID
Foreign Key(s)	PROD_CODE

Sr. #	Field Name	Data Type	Constraint	Description
1	PROD_CODE	CHAR(15)	Foreign key	Referring Product table
2	PROD_TYPE_ID	CHAR(35)	Primary key	Product Type ID
3	TYPE_DESC	CHAR(50)	Not Null	Product Type Description
4	MEASURE	CHAR(5)		Product Measure in Pounc

5	PRICE	CHAR(10)	Not Null	Product Price

SQL Query:

```
CREATE TABLE PRODUCT_TYPE (
   PROD_CODE CHAR(15) NOT NULL,
   PROD_TYPE_ID CHAR(35) NOT NULL,
   TYPE_DESC CHAR(50) NOT NULL,
   MEASURE CHAR(5),
   PRICE CHAR(10) NOT NULL,
   PRIMARY KEY (PROD_TYPE_ID),
   FOREIGN KEY (PROD_CODE)
      REFERENCES PRODUCT (PROD_CODE)
      ON DELETE CASCADE ON UPDATE RESTRICT);
```

Order Details:

Table Name	ORDER_DETAILS
Description	Stores information about the Order Deta
Primary Key	ORDER_LINE_ID
Foreign Key(s)	ORDER_ID, PROD_TYPE_ID

Sr. #	Field Name	Data Type	Constraint	Description
1	ORDER_ID	CHAR(35)	Foreign key	Referring Product table
2	ORDER_LINE_ID	CHAR(35)	Primary key	Order Line ID
3	PROD_TYPE_ID	CHAR(35)	Foreign key	Referring Product Type t
4	ORDER_DESC	CHAR(50)	Not Null	Order Description
5	QUANTITY	INT	Not Null	Purchase quantity

SQL Query:

```
CREATE TABLE ORDER_DETAILS (

   ORDER_ID CHAR(35) NOT NULL,

   ORDER_LINE_ID CHAR(35) NOT NULL,

   PROD_TYPE_ID CHAR(35) NOT NULL,

   ORDER_DESC CHAR(50) NOT NULL,

   QUANTITY INT NOT NULL,

   PRIMARY KEY (ORDER_LINE_ID),

   FOREIGN KEY (ORDER_ID)

      REFERENCES ORDERS (ORDER_ID)

      ON DELETE CASCADE ON UPDATE RESTRICT,

   FOREIGN KEY (PROD_TYPE_ID)

      REFERENCES PRODUCT_TYPE (PROD_TYPE_ID)

      ON DELETE CASCADE ON UPDATE RESTRICT);
```

Billing:

Table Name	BILLING
Description	Stores information about the Bi details
Primary Key	INVOICE_ID
Foreign Key(s)	ORDER_ID, CUST_ID, EMP_ID

Sr. #	Field Name	Data Type	Constraint	Description
1	INVOICE_ID	CHAR(35)	Primary ke	Invoice ID
2	ORDER_ID	CHAR(35)	Foreign ke	Referring Orders table
3	CUST_ID	CHAR(35)	Foreign ke	Referring Customer ta

4	EMP_ID	CHAR(35)	Foreign ke	Referring Employee ta
5	NET_AMOUNT	INT	Not Null	Sales total amount

<u>SQL Query:</u>

```
CREATE TABLE BILLING (
   INVOICE_ID CHAR(35) NOT NULL,
   ORDER_ID CHAR(35) NOT NULL,
   CUST_ID CHAR(35) NOT NULL,
   EMP_ID CHAR(35) NOT NULL,
   NET_AMOUNT INT NOT NULL,
   PRIMARY KEY (INVOICE_ID),
   FOREIGN KEY (ORDER_ID)
      REFERENCES ORDERS (ORDER_ID)
      ON DELETE CASCADE ON UPDATE RESTRICT,
   FOREIGN KEY (CUST_ID)
      REFERENCES CUSTOMER (CUST_ID)
      ON DELETE CASCADE ON UPDATE RESTRICT,
   FOREIGN KEY (EMP_ID)
      REFERENCES EMPLOYEE (EMP_ID)
      ON DELETE CASCADE ON UPDATE RESTRICT);
```

Appendix C: QUERIES GENERATED

Query 1:

Find the names of the employees who took the order before 8 May 2017.

```
SELECT * FROM ORDERS;
SELECT DISTINCT EMP_NAME FROM EMPLOYEE E, ORDERS O
WHERE E.EMP_ID=O.EMP_ID
AND O.ORDER_DT_TIME < ('2017-05-08');
```

Purpose:
This query can help the owner to track the employees who took the orders on, before or after any particular date.

Query 2:

Print the names of the customers who ordered at the bakery more than once.

```
SELECT CUST_NAME FROM CUSTOMER C, ORDERS O
WHERE C.CUST_ID=O.CUST_ID
GROUP BY O.CUST_ID
HAVING COUNT(O.CUST_ID)>1;
```

Purpose:

This query can help us to know about the customers who are ordering the bakery products continuously. The owner might give any promotional offers or certain discounts in future to his loyal customers under loyalty program.

Query 3:

Print the names of the customers, employee name who billed them, purchase order description and also the corresponding billing amount who purchased in the month of May.

```
SELECT O.ORDER_ID, C.CUST_NAME, E.EMP_NAME, OD.ORDER_DESC, B.NET_AMOUNT
FROM CUSTOMER C, EMPLOYEE E, ORDER_DETAILS OD, BILLING B, ORDERS O
WHERE B.CUST_ID=C.CUST_ID
AND B.EMP_ID=E.EMP_ID
AND B.ORDER_ID=O.ORDER_ID
AND O.ORDER_ID=OD.ORDER_ID
AND O.ORDER_DT_TIME LIKE '%-05-%';
#SELECT * FROM BILLING;
```

```
#UPDATE
#UPDATE BILLING
#SET NET_AMOUNT='21'
#WHERE INVOICE_ID='INV02';
```

Purpose:

This query helps us in fetching certain information about customers who purchased an item, employees who billed them, the item that was sold and the billing amount in any particular month.

Query 4:

Print the order details for the customer and also his name who paid the highest amount.

```
SELECT C.CUST_NAME, OD.ORDER_ID, OD.ORDER_LINE_ID, OD.ORDER_DESC,
OD.QUANTITY
FROM CUSTOMER C, ORDER_DETAILS OD, ORDERS O, BILLING B
WHERE B.CUST_ID=C.CUST_ID AND B.ORDER_ID=O.ORDER_ID AND
O.ORDER_ID=OD.ORDER_ID
AND B.NET_AMOUNT=(SELECT MAX(NET_AMOUNT) FROM BILLING);
```

Purpose:

This could help the owner to track the customers who spent the most in the bakery. The owner can use this information to see what products are in higher demand that customers are ready to pay highest amount for.

Query 5:

Print the vanilla flavored items with the bakery along with their unit price. Sort by item price from low to high.

```
#Sort by item price from low to high.
SELECT TYPE_DESC AS VANILLA_ITEMS, PRICE FROM PRODUCT_TYPE
WHERE TYPE_DESC LIKE 'VANILLA%'
ORDER BY PRICE ASC;
```

Purpose:

This query gives the information about certain flavored item in the bakery along with its price. The owner can help the customers in knowing the price of particular flavor if asked.

Query 6:

Compute the price as per the ordered quantity for any particular product type.

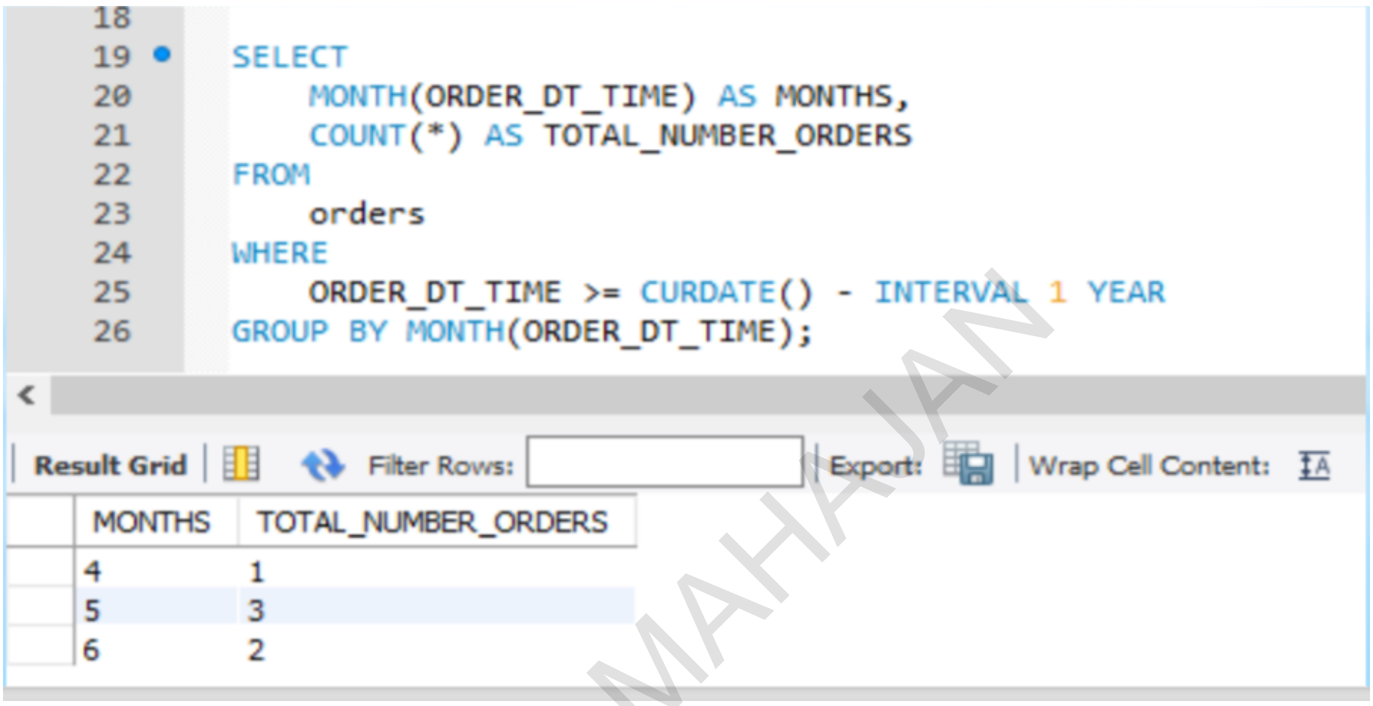
```
SELECT DISTINCT PT.PROD_TYPE_ID, PT.TYPE_DESC, (PT.PRICE*OD.QUANTITY) AS
PRODUCT_PRICE
FROM PRODUCT_TYPE PT, ORDER_DETAILS OD
WHERE PT.PROD_TYPE_ID=OD.PROD_TYPE_ID
AND PT.TYPE_DESC LIKE 'STRAWBERRY%'
ORDER BY PT.PROD_TYPE_ID;
```

Purpose:

This dynamic query helps the owner in calculating the cost of an item as per the quantity ordered.

Appendix D: SPECIAL FEATURE OF THE SYSTEM

- Count the Number of Orders per month to track the monthly sales.
- This allows the management at bakery to calculate the performance of the bakery in a particular month and thus, analyse the performance & profitability.



```
SELECT
    MONTH(ORDER_DT_TIME) AS MONTHS,
    COUNT(*) AS TOTAL_NUMBER_ORDERS
FROM
    orders
WHERE
    ORDER_DT_TIME >= CURDATE() - INTERVAL 1 YEAR
GROUP BY MONTH(ORDER_DT_TIME);
```

MONTHS	TOTAL_NUMBER_ORDERS
4	1
5	3
6	2

From the above screenshot, we can conclude that the maximum sale is in the month of May. Similarly, in future also the owner can track the monthly orders to analyze the monthly sales and increase in the revenue of the shop.

Appendix E: CHALLENGES ENCOUNTERED

❖ Bakery products like Cakes have multiple attributes such as flavor of the cakes and size of the cakes. Each attribute causes change in the final product price. Earlier we tried accommodating these attributes in the *Product* Master Table, but too many attributes proved challenging so, we created a new entity *Prod_Type.*

❖ Initially, we allocated Customer's Mobile_Number as the Primary key for Customer Table only to realize later that a Customer may change their mobile number or use a different mobile number at different times while coming to the store. Therefore, we created a new attribute Cust_ID for uniquely identifying each visiting customer.

❖ We first generated Order Details in the Billing table through the Cust_id. While running the query, when same customer is placing the order second time, we noticed that Billing Table is returning the previous order details as well. So, we joined *Order_Details* and *Billing* entities.

Appendix F: FUTURE SCOPE OF POS SYSTEM

- We can connect multiple stores through the POS system. Thus, it can show us the performance comparison between one bakery outlet to it's sister branch. Also, if we can see the real-time updated inventory at other outlet, this can enable the bakery owner to manage stocks better.
- If POS system is integrated with monitoring hardware at the bakery, it can be used by bakery to display the cake designs, availability of each cake and the variety of products available for the customers.
- We will be adding a Deliveries table too in our schema to handle out of bakery deliveries done delivery persons.
- The system can be further developed into a cloud based mobile application for owners to remotely monitor the sales at the bakery.