



UNIVERSITI MALAYA

WIA2001 DATABASE DATABASE DESIGN (FINAL REPORT)

Tutorial Group: 7

Lecturer : Profesor Madya Dr. Norjihan Binti Abdul Ghani

No	Name	Matric No
1	TEO WEN THING	22004856
2	PHUA WEN TENG	22052286
3	HON YAO ZHI	22004845
4	AMEERA BINTI OMAR	U2101228
5	NADHEA BINTI ISMAIL	U2100792

Table of Content

1.0 Introduction	2
2.0 Business Rules	3
3.0 System Objectives and Scope	4
4.0 Issues in Managing Data	5
5.0 User Requirements	5
6.0 Entities and Attributes Before Normalisation	6
7.0 Conceptual Entity Relationship Diagram	8
8.0 Normalisation process	9
8.1 Unnormalised Form (UNF)	9
8.1 First Normal Form (1NF)	10
8.2 Second Normal Form (2NF)	10
8.3 Third Normal Form (3NF)	13
9.0 Implementable Entity Relationship Diagram	17
10.0 Entities and Attributes After Normalisation	18
11.0 Sample SQL Statements	21
11.1 DDL Query	21
11.2 DML Query	25
11.3 Best SQL Statement	29
12.0 Screenshots of Tables from the Developed System	34
12.0 Discussion	37
12.1 Challenges Faced by Group	37
12.2 Lessons Learnt	38
13.0 Future Improvement	39
Appendix	40
i. Contributions by Each Member	40
ii. Meeting with Collaborator	42
iii. Letter to Company	43
iv. Organisation Verification	44
v. Peer Work Group Evaluation Form	45

1.0 Introduction

Presotea, a well-known Taiwanese brand has grown globally since its first shop in 2006. Now with over 370 franchises worldwide, including Malaysia since 2020. The name "Presotea" is a combination of "pressure" and "tea," highlighting the distinctive brewing method employed by the brand known as "pressure-steaming". Presotea offers various beverages including milk tea, fruit tea, lattes, and special drinks.

Our group has selected Presotea located in SS2, Petaling Jaya as our project collaborator. Its operation hours are from 11 am to 12 am daily. This shop is managed by a store manager, full-time workers and part-time workers. Each position has distinct roles like store manager is the one who manages the store to ensure the smooth functioning business. Full-time workers are responsible for opening and closing the store while part-time workers contribute to the daily activities by assisting the full-time workers. The employees are assigned to a different shift which is the morning shift (10:00 am to 5:30 pm) and night shift (5:00 pm to 1:00 am). The shift on Fridays and weekends is different as there is an additional middle shift to accommodate the high traffic of customers.

The problems currently exist with Presotea are they do not have a proper database that records customer purchases based on demographics, race, and age to predict customers' preferences. Other than that, they still use manual stock documentation which may lead to miscalculation and is a tedious process to handle. In addition, they don't have a proper system to record employee information for future branch openings, storing their basic details. Furthermore, they are planning to open up a new branch in the upcoming time, so that they can reach more customers to expand their business. This is one of the reasons they need a better database system to help them manage all the data.

The solution to the above-stated problem is to develop a proper database system. It can help them access information quickly and precisely. Hence, our group decided to develop a database management system for Presotea located in SS2.

2.0 Business Rules

An order can include one or many beverages. (1, m)

A beverage can be included in zero or many orders. (0, m)

An order can be handled by one and only one employee. (1, 1)

An employee can handle zero or many orders. (0, m)

An order belongs to one and only one customer. (1, 1)

A customer can make one or many orders. (1, m)

Each order can have zero or one associated food delivery. (0, 1)

Each food delivery is associated with one and only one order. (1, 1)

A beverage is made with one or many ingredients. (1, m)

An ingredient can be found in zero or many beverages. (0, m)

An ingredient is supplied by one and only one supplier. (1, 1)

One supplier can supply one or more ingredients. (1, m)

Each ingredient can have one or many inventory records. (1, m)

Each inventory record belongs to one and only one ingredient. (1, 1)

An employee can have zero or many attendance records. (0, m)

Each attendance record belongs to one and only one employee. (1, 1)

An employee can be associated with one and only one employee position. (1, 1)

Each employee position can be held by zero or many employees. (0, m)

An employee may have one or many shift schedules. (1, m)

Each shift schedule belongs to one and only one employee. (1, 1)

3.0 System Objectives and Scope

Scope:

The scope of the designed database system aims to enhance operational efficiency and customer service for the beverage shop based on insights gained from the interview dialogue. The system will comprehensively capture and manage customer data, including drink preferences, purchase history, and demographic information, enabling personalised customer interactions and targeted marketing efforts. Additionally, the database will track sales records and inventory information, facilitating real-time analysis, prediction, and proactive inventory management. Including employee information in the database system is anticipated for future expansion, supporting effective human resource management as the company grows. The designed system will play a central role in streamlining daily operations, improving customer experiences, and contributing to the overall success and growth of the beverage shop.

Objectives:

- To ease the updating and checking of inventory data by using query statements and tables.
- To track customer's purchase history.
- To help reduce redundancies and unnecessary data.
- To ease the customer data collection and storing process.
- To help organise inventory data by arranging them accordingly into tables.
- To ease the process of tracking sales to enhance overall business efficiency and decision-making process

4.0 Issues in Managing Data

1. The beverage shop does not have a proper database system to record employee information.
2. Using spreadsheets like Excel to keep track of the inventory left, where the worker will need to count the ingredients every week and update it in the spreadsheet may lead to data anomalies.
3. The workers are having a hard time recording the customers' orders.

5.0 User Requirements

- **Maintaining and recording employees' basic information**
 - To maintain (insert, update, and delete) data on staff.
 - To perform searches on employee detail
- **Maintaining and recording inventory data**
 - To maintain (insert, update, and delete) data on inventory.
 - To perform an inventory search
- **Maintaining and recording product information**
 - To maintain (insert, update, and delete) data on products such as ingredients, prices and product names.
- **Tracking data and record**
 - To record customer purchase history
 - To record supplier information
 - To record sales made
- **Generating report**
 - To retrieve top-selling drinks for specified months and years
 - To retrieve ingredients' details including unit cost and stocks with the suppliers' details respectively.
 - To retrieve customer information, including customer ID, name, contact, and email, along with the count of orders placed by each customer.
 - To retrieve total sales of beverages by category

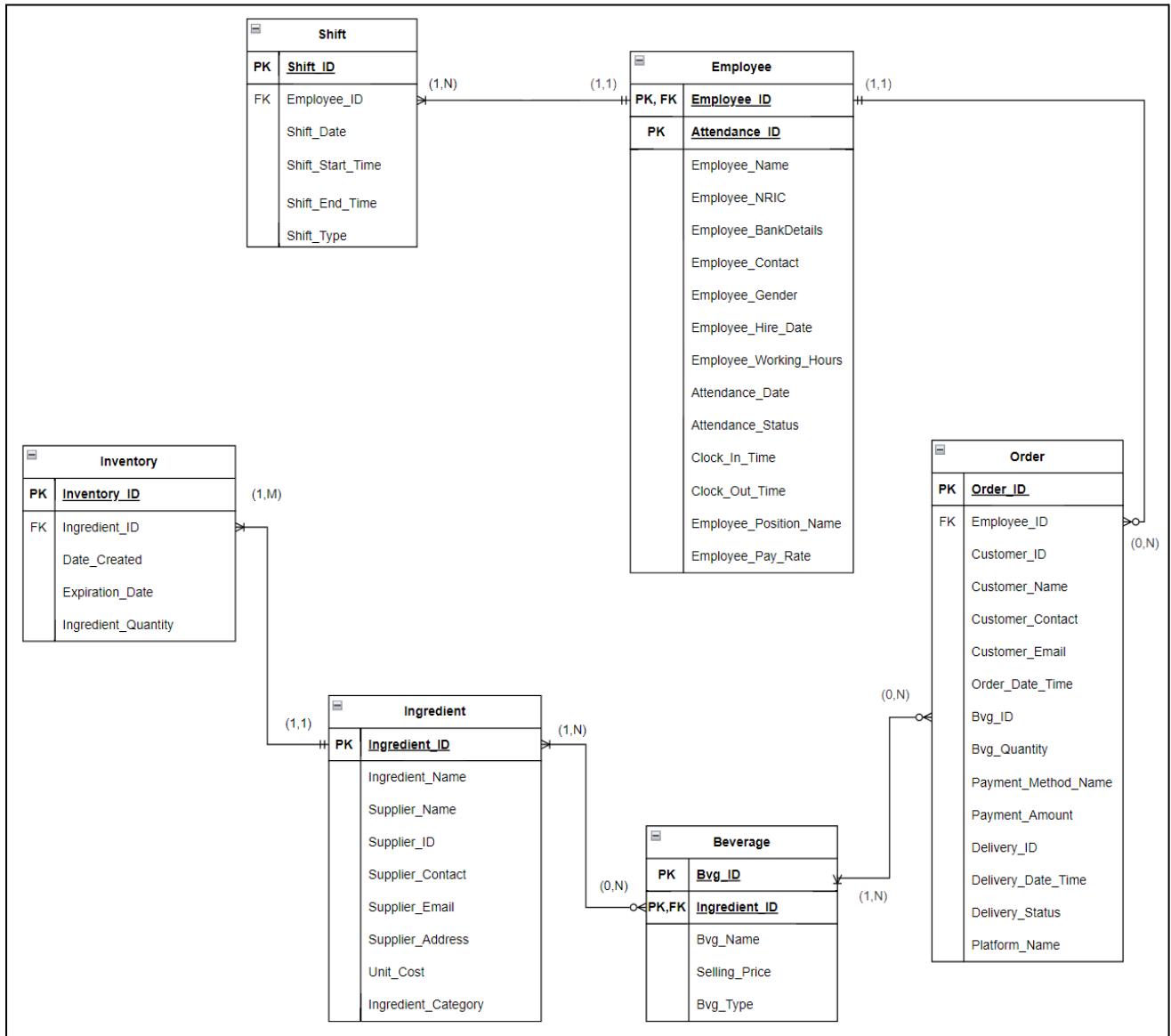
6.0 Entities and Attributes Before Normalisation

No.	Entities	Attributes
1	Order	<ul style="list-style-type: none">- Order_ID- Order_Date_Time- Delivery_ID- Delivery_Date_Time- Delivery_Status- Platform_Name
2	Employee	<ul style="list-style-type: none">- Employee_ID- Employee_Name- Employee_NRIC- Employee_BankDetails- Employee_Contact- Employee_Gender- Employee_Hire_Date- Employee_Position_ID- Employee_Working_Hours- Attendance_ID- Attendance_Date- Attendance_Status- Clock_In_Time- Clock_Out_Time- Employee_Position_Name- Employee_Pay_Rate- Shift_ID- Shift_Date- Shift_Start_Time- Shift_End_Time- Shift_Type
3	Customer	<ul style="list-style-type: none">- Customer_ID- Customer_Name

		<ul style="list-style-type: none"> - Customer_Contact - Customer_Email
4	Payment	<ul style="list-style-type: none"> - Payment_ID - Payment_Method_Name - Payment_Amount - Payment_Date_Time
5	Beverage	<ul style="list-style-type: none"> - Bvg_ID - Bvg_Name - Selling_Price - Bvg_Type - Bvg_Quantity
6	Ingredient	<ul style="list-style-type: none"> - Ingredient_ID - Ingredient_Name - Reorder_ID - Expiration_Date - Ingredient_Category - Ingredient_Quantity_In_Stock - Reorder_Point - Last_Reorder_Date
7	Supplier	<ul style="list-style-type: none"> - Supplier_ID - Supplier_Name - Supplier_Contact - Supplier_Email - Supplier_Address

7.0 Conceptual Entity Relationship Diagram

Initial ERD:



8.0 Normalisation process

The dependency diagrams below illustrate the pre-normalisation stage, where certain tables have already undergone normalisation to the third normal form but others still have transitive relationships and are in the first and second normal forms, respectively.

8.1 Unnormalised Form (UNF)

Order Table								
Order_ID	Employee_ID	Customer_ID	Customer_Name	Customer_Contact	Customer_Email	Order_Date_Time	Bvg_ID	Bvg_Quantity
Payment_Method_Name	Payment_Amount	Delivery_ID	Delivery_Date_Time	Delivery_Status	Platform_Name			
Beverage Table								
Bvg_ID	Bvg_Name	Ingredient_ID	Selling_Price	Bvg_Type				
Ingredient Table								
Ingredient_ID	Ingredient_Name	Supplier_ID	Supplier_Name	Supplier_Contact	Supplier_Email	Supplier_Address	Unit_Cost	Ingredient_Category
Inventory Table								
Inventory_ID	Ingredient_ID	Date_Created	Expiration_Date	Ingredient_Quantity				
Employee Table								
Employee_ID	Employee_Name	Employee_NRIC	Employee_BankDetails	Employee_Contact	Employee_Gender	Employee_Hire_Date	Employee_Working_Hours	
Attendance_ID	Attendance_Date	Attendance_Status	Clock_In_Time	Clock_Out_Time	Employee_Position_Name	Employee_Pay_Rate		
Shift Schedule Table								
Shift_ID	Employee_ID	Shift_Date	Shift_Start_Time	Shift_End_Time	Shift_Type			

8.1 First Normal Form (1NF)

Identify Primary Key.

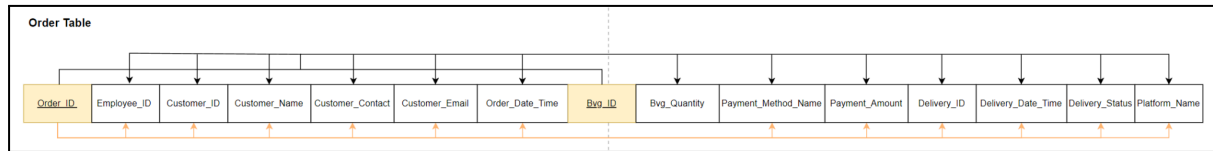
No repeating groups in each cell.

Order Table								
<u>Order_ID</u>	Employee_ID	Customer_ID	Customer_Name	Customer_Contact	Customer_Email	Order_Date_Time	<u>Bvg_ID</u>	Bvg_Quantity
Payment_Method_Name	Payment_Amount	Delivery_ID	Delivery_Date_Time	Delivery_Status	Platform_Name			
Beverage Table								
<u>Bvg_ID</u>	Bvg_Name	<u>Ingredient_ID</u>	Selling_Price	Bvg_Type				
Ingredient Table								
<u>Ingredient_ID</u>	Ingredient_Name	Supplier_ID	Supplier_Name	Supplier_Contact	Supplier_Email	Supplier_Address	Unit_Cost	Ingredient_Category
Inventory Table								
<u>Inventory_ID</u>	Ingredient_ID	Date_Created	Expiration_Date	Ingredient_Quantity				
Employee Table								
Employee_ID	Employee_Name	Employee_NRIC	Employee_BankDetails	Employee_Contact	Employee_Gender	Employee_Hire_Date	Employee_Working_Hours	
<u>Attendance_ID</u>	Attendance_Date	Attendance_Status	Clock_In_Time	Clock_Out_Time	Employee_Position_Name	Employee_Pay_Rate		
Shift Schedule Table								
<u>Shift_ID</u>	Employee_ID	Shift_Date	Shift_Start_Time	Shift_End_Time	Shift_Type			

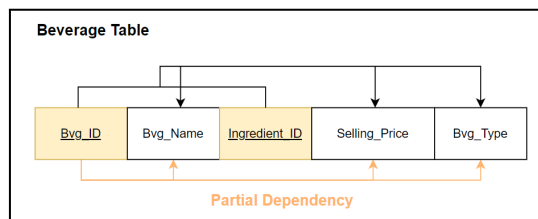
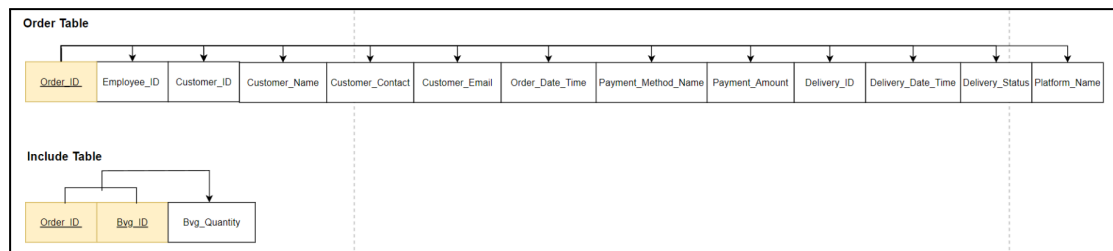
8.2 Second Normal Form (2NF)

Satisfy all 1NF conditions.

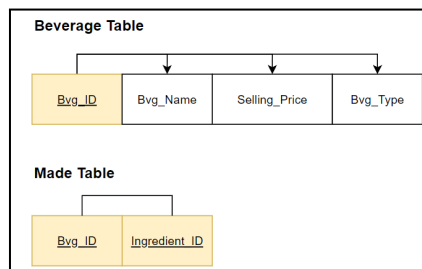
No partial dependency.



1. For the Order Table, all attributes except Bvg_Quantity are partially dependent on Order_ID only. So we create an Include table to separate the partial dependency in the Order table.

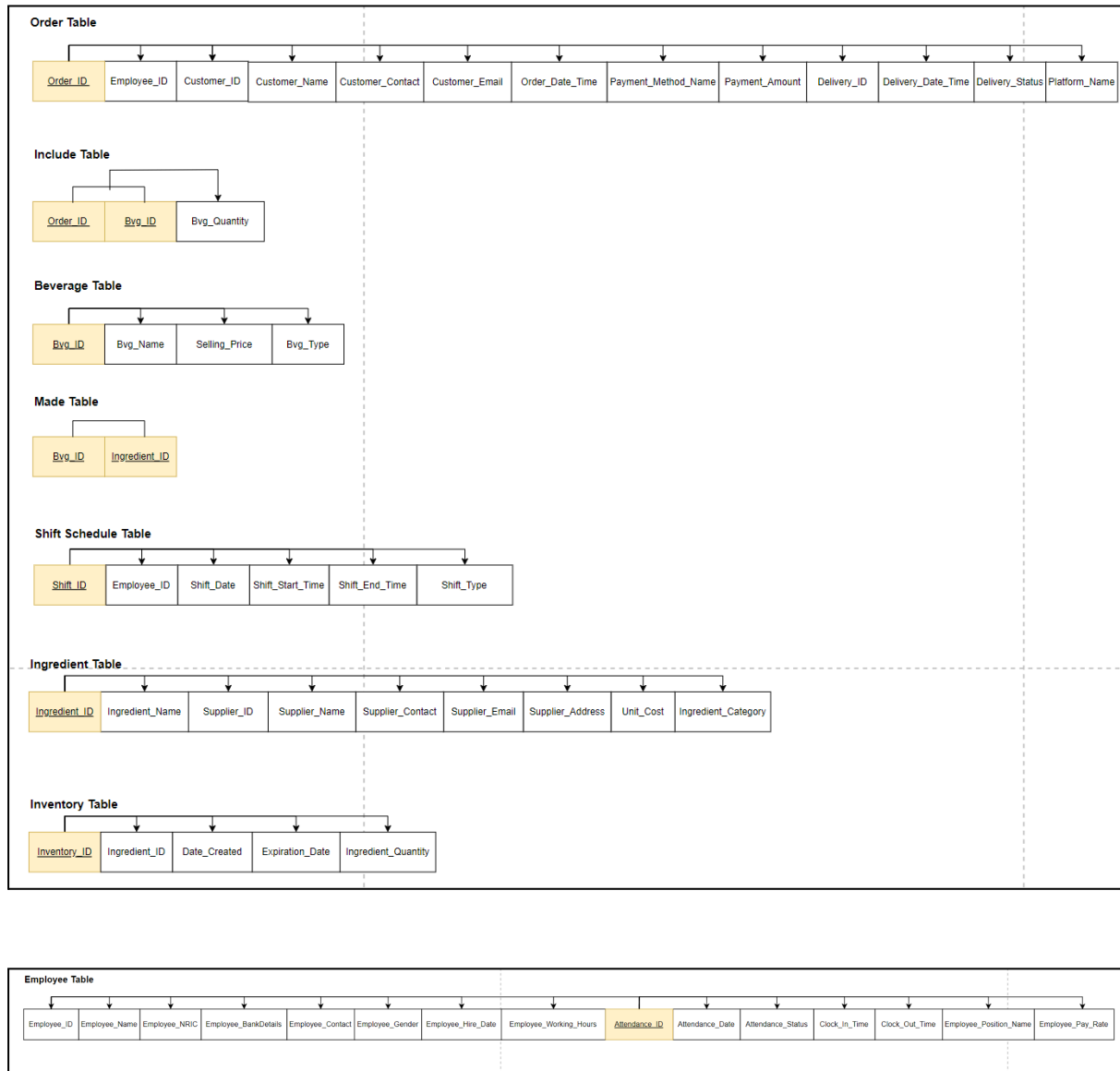


2. For the Beverage table, Bvg_Name, Selling_Price and Bvg_Type are partially dependent on Bvg_ID, so we create a Made table to split the partial dependencies.



3. Employee Table, Shift Schedule Table, Ingredient Table and Inventory Table are in 2NF since they have no partial dependencies.

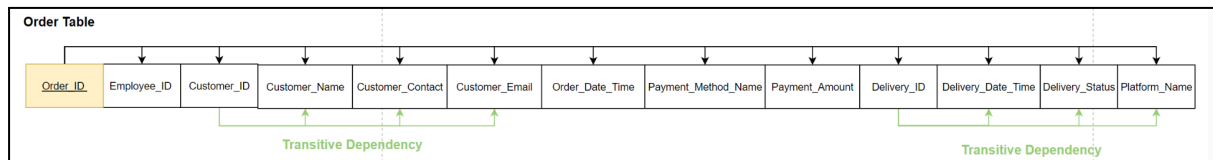
Complete 2NF tables



8.3 Third Normal Form (3NF)

Satisfy all 1NF and 2NF conditions.

No transitive dependency.

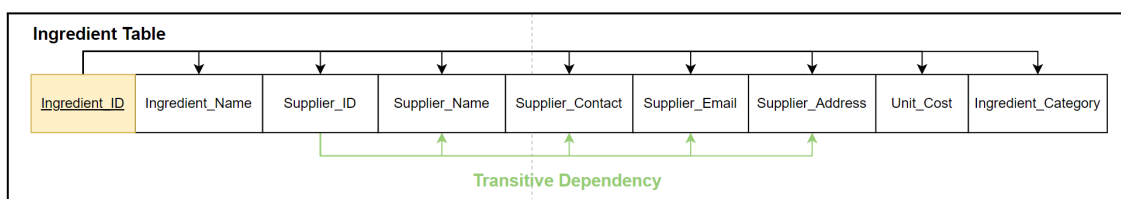
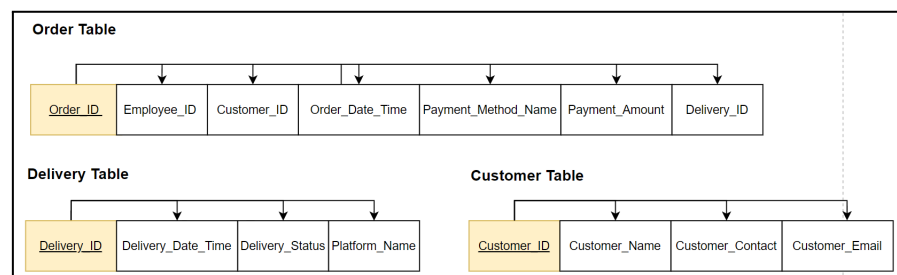


1. For the Order table,

Customer_Name, Customer_Contact and Customer_Email are transitively dependent on Customer_ID while

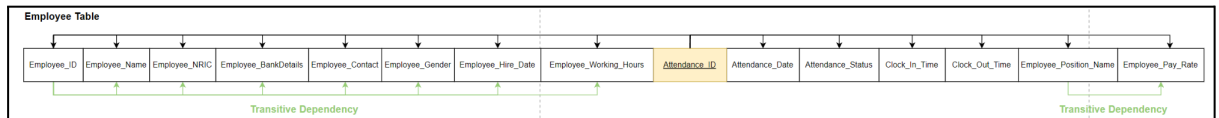
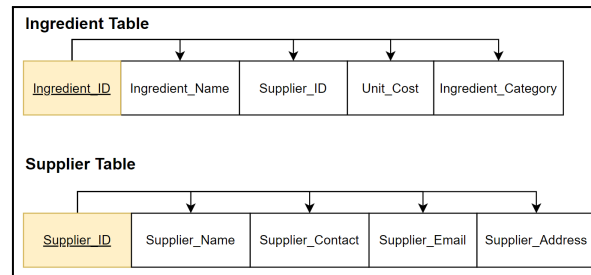
Delivery_Date_Time, Delivery_Status and Platform_Name are transitively dependent on Delivery_ID.

So we create a Delivery table and a Customer table to eliminate those transitive dependencies. Delivery_ID and Customer_ID become the PK for each table.

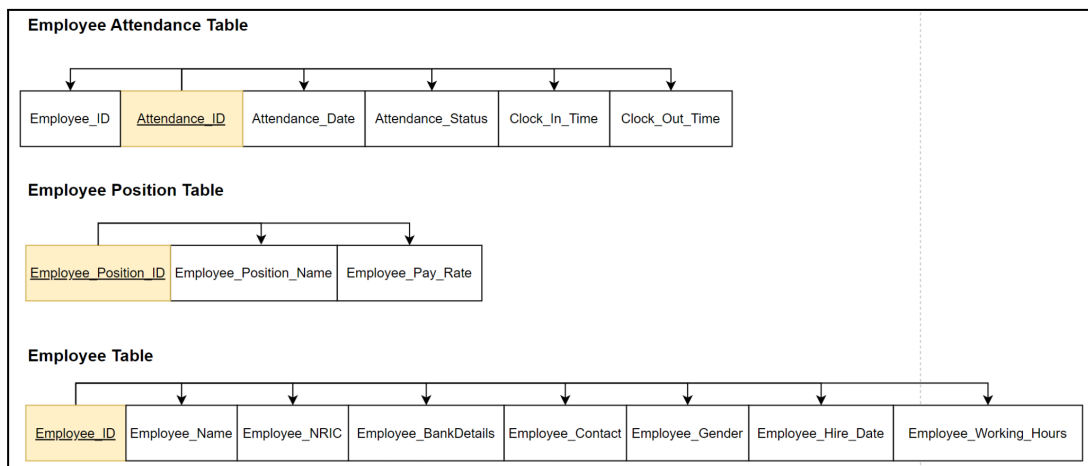


2. For the Ingredient table, Supplier_Name, Supplier_Contact, Supplier_Email and Supplier_Address are transitively dependent on Supplier_ID.

So we create a Supplier table to eliminate those transitive dependencies.



3. It will be the same for the Employee table, we create an Employee Attendance table and Employee Position table to remove the transitive dependencies. But for the Employee Position Table, we create an attribute which is the Employee_Position_ID as the Primary Key of the table.



Complete 3NF tables



Supplier Table

<u>Supplier_ID</u>	Supplier_Name	Supplier_Contact	Supplier_Email	Supplier_Address
--------------------	---------------	------------------	----------------	------------------

Inventory Table

<u>Inventory_ID</u>	Ingredient_ID	Date_Created	Expiration_Date	Ingredient_Quantity
---------------------	---------------	--------------	-----------------	---------------------

Employee Table

<u>Employee_ID</u>	Employee_Name	Employee_NRIC	Employee_BankDetails	Employee_Contact	Employee_Gender	Employee_Hire_Date
Employee_Working_Hours	Employee_Position_ID					

Employee Position Table

<u>Employee_Position_ID</u>	Employee_Position_Name	Employee_Pay_Rate
-----------------------------	------------------------	-------------------

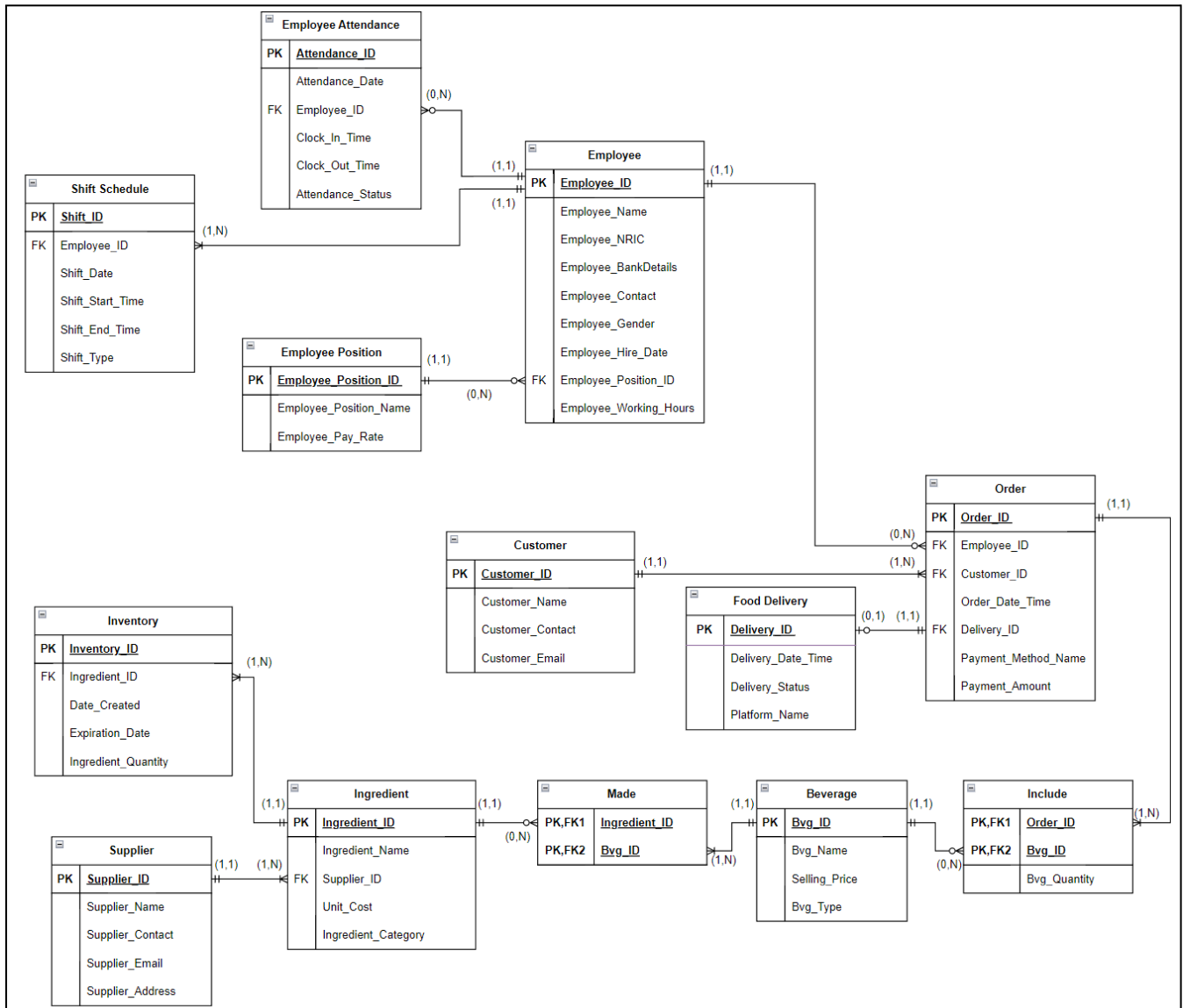
Employee Attendance Table

Employee_ID	<u>Attendance_ID</u>	Attendance_Date	Attendance_Status	Clock_In_Time	Clock_Out_Time
-------------	----------------------	-----------------	-------------------	---------------	----------------

Shift Schedule Table

<u>Shift_ID</u>	Employee_ID	Shift_Date	Shift_Start_Time	Shift_End_Time	Shift_Type
-----------------	-------------	------------	------------------	----------------	------------

9.0 Implementable Entity Relationship Diagram



10.0 Entities and Attributes After Normalisation

No.	Entities	Attributes
1.	Order	<ul style="list-style-type: none">- Order_ID PK- Employee_ID- Customer_ID- Order_Date_Time- Payment_Method_Name- Payment_Amount- Delivery_ID
2	Delivery	<ul style="list-style-type: none">- Delivery_ID PK- Delivery_Date_Time- Delivery_Status- Platform_Name
3	Customer	<ul style="list-style-type: none">- Customer_ID PK- Customer_Name- Customer_Contact- Customer_Email
4	Include	<ul style="list-style-type: none">- Order_ID PK- Bvg_ID PK- Bvg_Quantity
5	Beverage	<ul style="list-style-type: none">- Bvg_ID PK- Bvg_Name- Selling_Price- Bvg_Type
6	Made	<ul style="list-style-type: none">- Bvg_ID PK- Ingredient_ID PK
7	Supplier	<ul style="list-style-type: none">- Supplier_ID PK- Supplier_Name

		<ul style="list-style-type: none"> - Supplier_Contact - Supplier_Email - Supplier_Address
8	Inventory	<ul style="list-style-type: none"> - Inventory_ID PK - Ingredient_ID - Date_Created - Expiration_Date - Ingredient_Quantity
9	Employee	<ul style="list-style-type: none"> - Employee_ID PK - Employee_Name - Employee_NRIC - Employee_BankDetails - Employee_Contact - Employee_Gender - Employee_Hire_Date - Employee_Position_ID - Employee_Working_Hours
10	Employee Position	<ul style="list-style-type: none"> - Employee_Position_ID PK - Employee_Position_Name - Employee_Pay_Rate
11	Employee Attendance	<ul style="list-style-type: none"> - Attendance_ID PK - Employee_ID - Attendance_Date - Attendance_Status - Clock_In_Time - Clock_Out_Time
12	Ingredient	<ul style="list-style-type: none"> - Ingredient_ID PK - Ingredient_Name - Supplier_ID - Unit_Cost

		- Ingredient_Category
13	Shift Schedule	<ul style="list-style-type: none"> - Shift_ID PK - Employee_ID - Shift_Date - Shift_Start_Time - Shift_End_Time - Shift_Type

11.0 Sample SQL Statements

11.1 DDL Query

Query #1 - Creating all the tables needed

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1	DELIVERY_ID	VARCHAR2(10 BYTE)	No	(null)	1	(null)
2	DELIVERY_DATE_TIME	TIMESTAMP(6)	No	(null)	2	(null)
3	DELIVERY_STATUS	VARCHAR2(10 BYTE)	No	(null)	3	(null)
4	PLATFORM_NAME	VARCHAR2(20 BYTE)	No	(null)	4	(null)

The above screenshot shows one of the examples that we have created, which is the **FOOD_DELIVERY** table which contains attributes which are “**DELIVERY_ID**”, “**DELIVERY_DATE_TIME**”, “**DELIVERY_STATUS**” and “**PLATFORM_NAME**”. The type of attributes includes **VARCHAR** to store character string, and **TIMESTAMP** to store the date and time of the delivery. The **NOT NULL** constraint means that this variable must contain a value and it cannot be left empty or null.

```
CREATE TABLE Food_Delivery (  
  Delivery_ID VARCHAR(10) NOT NULL,  
  Delivery_Date_Time TIMESTAMP NOT NULL,  
  Delivery_Status VARCHAR(10) NOT NULL,  
  Platform_Name VARCHAR(20) NOT NULL,  
  PRIMARY KEY(Delivery_ID)  
);
```

CREATE TABLE keyword initiates the creation of a new table named "**FOOD_DELIVERY**". After running this statement, we will have the **FOOD_DELIVERY** table with all the attributes mentioned above along with its **PRIMARY KEY**, the “**DELIVERY_ID**”. Each delivery record will have a unique identifier to prevent duplicates and ensure data integrity.

Query #2 - Alter the table to change the data type.

One of the problems we encountered was that we defined the `Supplier_Contact` attribute as an integer to store a numerical contact value.

```
CREATE TABLE Supplier (  
  Supplier_ID VARCHAR(4) NOT NULL,  
  Supplier_Name VARCHAR(50) NOT NULL,  
  Supplier_Contact INTEGER NOT NULL,  
  Supplier_Email VARCHAR(255) NOT NULL,  
  Supplier_Address VARCHAR(320) NOT NULL,  
  PRIMARY KEY(Supplier_ID)  
);
```

After encountering an error where we cannot insert values into the table, We need to modify the existing table "`Supplier`" by changing the data type of the `Supplier_Contact` column from `INTEGER` to `VARCHAR(15)`. This allows text-based contact information (e.g., phone numbers) in the format of 012-3456789 for instance, instead of just numerical values, as we will not perform any calculations using the contact number.

```
ALTER TABLE Supplier  
MODIFY Supplier_Contact VARCHAR(15);
```

▼

SYS.SUPPLIER		
P	SUPPLIER_ID	VARCHAR2 (4 BYTE)
	SUPPLIER_NAME	VARCHAR2 (50 BYTE)
	SUPPLIER_CONTACT	VARCHAR2 (15 BYTE)
	SUPPLIER_EMAIL	VARCHAR2 (255 BYTE)
	SUPPLIER_ADDRESS	VARCHAR2 (320 BYTE)
SUPPLIER_PK (SUPPLIER_ID)		

Query #3 - Add constraints to the table

For example, below are queries for creating an **Employee** table:

SYS.EMPLOYEE		
P *	EMPLOYEE_ID	VARCHAR2 (4 BYTE)
*	EMPLOYEE_NAME	VARCHAR2 (50 BYTE)
*	EMPLOYEE_NRIC	VARCHAR2 (20 BYTE)
*	EMPLOYEE_BANKDETAILS	VARCHAR2 (20 BYTE)
*	EMPLOYEE_CONTACT	VARCHAR2 (15 BYTE)
*	EMPLOYEE_GENDER	CHAR (6 BYTE)
*	EMPLOYEE_HIRE_DATE	DATE
*	EMPLOYEE_POSITION_ID	VARCHAR2 (4 BYTE)
*	EMPLOYEE_WORKING_HOURS	NUMBER (*,0)
EMPLOYEE_PK(EMPLOYEE_ID)		

SYS.EMPLOYEE		
P *	EMPLOYEE_ID	VARCHAR2 (4 BYTE)
*	EMPLOYEE_NAME	VARCHAR2 (50 BYTE)
*	EMPLOYEE_NRIC	VARCHAR2 (20 BYTE)
*	EMPLOYEE_BANKDETAILS	VARCHAR2 (20 BYTE)
*	EMPLOYEE_CONTACT	VARCHAR2 (15 BYTE)
*	EMPLOYEE_GENDER	CHAR (6 BYTE)
*	EMPLOYEE_HIRE_DATE	DATE
F *	EMPLOYEE_POSITION_ID	VARCHAR2 (4 BYTE)
*	EMPLOYEE_WORKING_HOURS	NUMBER (*,0)
EMPLOYEE_PK(EMPLOYEE_ID)		
FK_EMPLOYEE(EMPLOYEE_POSITION_ID)		

```
ALTER TABLE Employee
ADD CONSTRAINT FK_Employee FOREIGN KEY(Employee_Position_ID)
REFERENCES Employee_Position(Employee_Position_ID);
```

In the given query, we designate **EMPLOYEE_ID** as the primary key for the unique identification of each employee record. The **FOREIGN KEY** constraint (**FK_Employee**) on **EMPLOYEE_POSITION_ID** ensures that the referenced position ID exists in the **EMPLOYEE_POSITION** table, maintaining consistency between employee and position data. This constraint prevents the assignment of non-existent positions to employees. The resulting **EMPLOYEE** table includes all specified attributes, featuring **EMPLOYEE_ID** as the primary key and **EMPLOYEE_POSITION_ID** as the foreign key referencing the **EMPLOYEE_POSITION** table.

Employee table:

	❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1	EMPLOYEE_ID	VARCHAR2(4 BYTE)	No	(null)	1 (null)	
2	EMPLOYEE_NAME	VARCHAR2(50 BYTE)	No	(null)	2 (null)	
3	EMPLOYEE_NRIC	VARCHAR2(20 BYTE)	No	(null)	3 (null)	
4	EMPLOYEE_BANKDETAILS	VARCHAR2(20 BYTE)	No	(null)	4 (null)	
5	EMPLOYEE_CONTACT	VARCHAR2(20 BYTE)	No	(null)	5 (null)	
6	EMPLOYEE_GENDER	CHAR(6 BYTE)	No	(null)	6 (null)	
7	EMPLOYEE_HIRE_DATE	DATE	No	(null)	7 (null)	
8	EMPLOYEE_POSITION_ID	VARCHAR2(4 BYTE)	No	(null)	8 (null)	
9	EMPLOYEE_WORKING_HOURS	NUMBER(38,0)	No	(null)	9 (null)	

The above screenshot shows the **EMPLOYEE** table that we have created.

11.2 DML Query

Query #1 - To insert a new query into the ingredient table

SQL statement below inserts multiple rows of data into the **INGREDIENT** table in a single query.

INSERT ALL: This clause initiates a multi-row insert.

INTO Ingredient(...column names...): Specifies the target table and the columns to receive data.

VALUES (...value sets...): Provides the values to be inserted for each row.

SELECT * FROM DUAL: A dummy **SELECT** statement is required by Oracle to terminate the multi-row insert syntax.

```
INSERT ALL
INTO Ingredient(Ingredient_ID, Ingredient_Name, Supplier_ID, Unit_Cost, Ingredient_Category)
VALUES ('IG001', 'Sugar', 'S001', 10, 'A')
INTO Ingredient(Ingredient_ID, Ingredient_Name, Supplier_ID, Unit_Cost, Ingredient_Category)
VALUES ('IG004', 'Brown Sugar', 'S001', 25, 'B')
INTO Ingredient(Ingredient_ID, Ingredient_Name, Supplier_ID, Unit_Cost, Ingredient_Category)
VALUES ('IG002', 'Milk', 'S002', 30, 'C')
INTO Ingredient(Ingredient_ID, Ingredient_Name, Supplier_ID, Unit_Cost, Ingredient_Category)
VALUES ('IG005', 'Tea', 'S002', 5, 'D')
INTO Ingredient(Ingredient_ID, Ingredient_Name, Supplier_ID, Unit_Cost, Ingredient_Category)
VALUES ('IG003', 'Boba', 'S003', 15, 'E')
INTO Ingredient(Ingredient_ID, Ingredient_Name, Supplier_ID, Unit_Cost, Ingredient_Category)
VALUES ('IG006', 'Jelly', 'S003', 12, 'F')
SELECT * FROM DUAL;
```

The result of the above statement is as follows:

	INGREDIENT_ID	INGREDIENT_NAME	SUPPLIER_ID	UNIT_COST	INGREDIENT_CATEGORY
1	IG001	Sugar	S001	10	A
2	IG004	Brown Sugar	S001	25	B
3	IG002	Milk	S002	30	C
4	IG005	Tea	S002	5	D
5	IG003	Boba	S003	15	E
6	IG006	Jelly	S003	12	F

Query #2 - To update the data in the table by using certain conditions

INGREDIENT table before updated:

	INGREDIENT_ID	INGREDIENT_NAME	SUPPLIER_ID	UNIT_COST	INGREDIENT_CATEGORY
1	IG001	Sugar	S001	10	A
2	IG004	Brown Sugar	S001	25	B
3	IG002	Milk	S002	30	C
4	IG005	Tea	S002	5	D
5	IG003	Boba	S003	15	E
6	IG006	Jelly	S003	12	F

```
UPDATE INGREDIENT
SET UNIT_COST = 20
WHERE INGREDIENT_NAME='Boba';
```

The code above is used to update the existing data in the existing table. For this example, we are using the **INGREDIENT** table.

The **SET** keyword is used to indicate the new values for insertion into columns. In this instance, the focus is on the "**UNIT_COST**" column, set to the value 20. The **WHERE** clause specifies the rows to be updated; here, it targets the row where the "**INGREDIENT_NAME**" is 'Boba'. Thus, the statement updates the value of column "**UNIT_COST**" to '20' for the row that contains 'Boba' as the "**INGREDIENT_NAME**".

The result of the above statement is as follows:

	INGREDIENT_ID	INGREDIENT_NAME	SUPPLIER_ID	UNIT_COST	INGREDIENT_CATEGORY
1	IG001	Sugar	S001	10	A
2	IG004	Brown Sugar	S001	25	B
3	IG002	Milk	S002	30	C
4	IG005	Tea	S002	5	D
5	IG003	Boba	S003	20	E
6	IG006	Jelly	S003	12	F

Query #3 - To display the categorised information using **INNER JOIN**

```
SELECT Employee.Employee_ID AS EmployeeId,  
Employee.Employee_Working_Hours AS WorkingHours,  
Employee.Employee_Name AS EmployeeName,  
Employee_Attendance.Attendance_Date AS AttendanceDate,  
Employee_Attendance.Attendance_Status AS AttendanceStatus,  
Employee_Attendance.Clock_In_Time AS ClockInTime,  
Employee_Attendance.Clock_Out_Time AS ClockOutTime  
FROM Employee  
INNER JOIN Employee_Attendance  
ON Employee.Employee_ID=Employee_Attendance.Employee_ID  
WHERE Attendance_Status='Present';
```

The provided code retrieves data from two tables, **EMPLOYEE** and **EMPLOYEE_ATTENDANCE**, and filters the results based on specific conditions. The **SELECT** statement is utilised to choose the desired columns, including aliases like "EmployeeId," "WorkingHours," "EmployeeName," "AttendanceDate," "AttendanceStatus," "ClockInTime," and "ClockOutTime." The **FROM** clause specifies the tables from which data is to be retrieved, followed by an **INNER JOIN** clause that combines rows based on the "EMPLOYEE_ID" column. This ensures that only rows with matching values in the "EMPLOYEE_ID" column from both tables are included in the result. The **WHERE** clause filters the data, allowing only rows where the "ATTENDANCE_STATUS" column has the value 'PRESENT' to be returned.

The result of the above statements is as follows:

	EMPLOYEEID	WORKINGHOURS	EMPLOYEENAME	ATTENDANCEDATE	ATTENDANCESTATUS	CLOCKINTIME	CLOCKOUTTIME
1	E001	45	John Doe	12/11/2023	Present	01/01/2024 08:00:00.000000000	01/01/2024 17:00:00.000000000
2	E004	30	Sarah Tan	13/11/2023	Present	01/01/2024 12:00:00.000000000	01/01/2024 21:00:00.000000000
3	E001	45	John Doe	14/11/2023	Present	01/01/2024 08:00:00.000000000	01/01/2024 17:00:00.000000000

Query #4 - To display all the categorised data using **LEFT JOIN**

```
SELECT Orders.Order_ID AS OrderId,  
Food_Delivery.Delivery_Status AS Status,  
Includes.Bvg_ID AS BeverageId,  
Customer.Customer_Name AS Customer,  
Employee.Employee_Name AS Employee  
FROM Orders  
LEFT JOIN Food_Delivery ON  
Orders.Delivery_ID=Food_Delivery.Delivery_ID  
LEFT JOIN Customer ON  
Orders.Customer_ID=Customer.Customer_ID  
LEFT JOIN Employee ON  
Orders.Employee_ID=Employee.Employee_ID  
LEFT JOIN Includes ON  
Orders.Order_ID=Includes.Order_ID  
WHERE Delivery_Status ='Delivered';
```

The provided SQL query retrieves information about delivered orders, including details such as **ORDER_ID**, **DELIVERY_STATUS**, **BEVERAGE_ID**, **CUSTOMER_NAME**, and **EMPLOYEE_NAME**. It achieves this by performing a series of **LEFT JOIN** operations on multiple tables. The first **LEFT JOIN** connects the **ORDERS** table with the **FOOD_DELIVERY** table based on the common column **DELIVERY_ID**. The second **LEFT JOIN** links the result with the **CUSTOMER** table using the **CUSTOMER_ID** column. The third **LEFT JOIN** involves the **EMPLOYEE** table and uses the **EMPLOYEE_ID** column to establish connections. Finally, the fourth **LEFT JOIN** brings in information from the **INCLUDES** table by matching the **ORDER_ID** column. The **WHERE** clause filters the results to include only rows where the delivery status is **'DELIVERED'**. As a result, the query produces a dataset that combines order details with associated delivery, customer, employee, and beverage information for those orders marked as **'DELIVERED'**.

The result of the above statements is as follows:

	ORDERID	STATUS	BEVERAGEID	CUSTOMER	EMPLOYEE
1	1002	Delivered	M001	Muthu	John Doe
2	1003	Delivered	F001	John	Sarah Tan
3	1003	Delivered	M001	John	Sarah Tan
4	1003	Delivered	M002	John	Sarah Tan
5	1004	Delivered	F001	John	Sarah Tan

11.3 Best SQL Statement

Query #1 - To retrieve a list of selling drinks for a **specified month and year** ordered from highest to lowest quantity sold.

```
SELECT
    Beverage.Bvg_ID,
    Beverage.Bvg_Name,
    SUM(Includes.Bvg_Quantity) AS TotalQuantitySold
FROM
    Orders
JOIN
    Includes ON Orders.Order_ID = Includes.Order_ID
JOIN
    Beverage ON Includes.Bvg_ID = Beverage.Bvg_ID
WHERE
    EXTRACT(MONTH FROM Orders.Order_Date_Time) = 11 -- Replace with your desired month
    AND EXTRACT(YEAR FROM Orders.Order_Date_Time) = 2023 -- Replace with your desired year
GROUP BY
    Beverage.Bvg_ID, Beverage.Bvg_Name
ORDER BY
    TotalQuantitySold DESC;
```

	BVG_ID	BVG_NAME	TOTALQUANTITYSOLD
1	M001	Panda Milk Tea	6
2	F001	Passionfruit Tea	2
3	M002	Brown Sugar Boba Milk Tea	2
4	F002	Grapefruit Boba Tea	1

This SQL code retrieves information about the beverage that had the highest total quantity sold in a specified month and year. It does this by joining the **Orders**, **Includes**, and **Beverage** tables, and filtering the results based on the month and year extracted from the order date. The query calculates the total quantity sold best for each beverage, grouping the results by beverage ID and name, and then orders them in descending order based on the total quantity sold. Finally, the outer query selects the first row using the **ROWNUM** condition, providing the beverage with the highest total quantity sold for the specified month and year.

Query #2 - To retrieve the stock amount for each ingredient together with their details.

```
SELECT
    I2.Ingredient_ID,
    I2.Ingredient_Name,
    I2.Unit_Cost,
    S.Supplier_ID,
    S.Supplier_Name,
    S.Supplier_Contact,
    S.Supplier_Email,
    SUM(I.Ingredient_Quantity) AS TotalStockQuantity
FROM
    inventory I
JOIN
    ingredient I2 ON I.Ingredient_ID = I2.Ingredient_ID
JOIN
    supplier S ON I2.Supplier_ID = S.Supplier_ID
LEFT JOIN
    made M ON I.Ingredient_ID = M.Ingredient_ID
GROUP BY
    I2.Ingredient_ID, I2.Ingredient_Name, I2.Unit_Cost, S.Supplier_ID, S.Supplier_Name, S.Supplier_Contact, S.Supplier_Email
ORDER BY
    I2.Ingredient_ID ASC;
```

The provided SQL query retrieves information about ingredients within a beverage shop's inventory. This is because our inventory table has many entries for the same ingredients to differentiate between the same ingredient with different in-stock dates and expiry dates. So to ease the user to check the total stock amount and their respective supplier details, we have this SQL statement.

Part of the inventory table

	INVENTORY_ID	INGREDIENT_ID	DATE_CREATED	EXPIRATION_DATE	INGREDIENT_QUANTITY
1	IV001	IG001	25/08/2023	08/02/2024	100
2	IV002	IG002	25/08/2023	10/12/2023	50
3	IV003	IG004	04/09/2023	06/03/2024	100
4	IV004	IG002	15/09/2023	03/01/2024	75
5	IV005	IG001	18/09/2023	15/03/2024	150

The code joins three tables—**inventory**, **ingredient**, and **supplier**—to amalgamate relevant details. The resulting dataset includes the **Ingredient_ID**, **Ingredient_Name**, **Unit_Cost**, and the supplier's identification (**Supplier_ID**), name (**Supplier_Name**), contact information (**Supplier_Contact**), and email (**Supplier_Email**).

The aspect of this query is the use of the **SUM** function, aggregating the **Ingredient_Quantity** from the inventory table to calculate the total stock quantity for each ingredient. The **GROUP BY** clause ensures the aggregation is performed for each unique

combination of **Ingredient_ID**, **Ingredient_Name**, **Unit_Cost**, **Supplier_ID**, **Supplier_Name**, **Supplier_Contact**, and **Supplier_Email**. The final result is ordered in ascending order based on **Ingredient_ID**. This query provides valuable insights into the inventory management of the beverage shop, offering a summarised view of each ingredient's total stock quantity, unit cost, and supplier details.

The result of the above statement is as follows:

	INGREDIENT_ID	INGREDIENT_NAME	UNIT_COST	SUPPLIER_ID	SUPPLIER_NAME	SUPPLIER_CONTACT	SUPPLIER_EMAIL	TOTALSTOCKQUANTITY
1	IG001	Sugar	10	S001	SweetHarvest	013-4455667	taeyeon@sweetharvest.com	1400
2	IG002	Milk	30	S002	PureLeaf Teas	014-9955667	minnie@pureleaf teas.com	450
3	IG003	Boba	20	S003	BobaDelight Supplies	011-4455667	79sana@bobadelight.com	150
4	IG004	Brown Sugar	25	S001	SweetHarvest	013-4455667	taeyeon@sweetharvest.com	100
5	IG005	Tea	5	S002	PureLeaf Teas	014-9955667	minnie@pureleaf teas.com	200
6	IG006	Jelly	12	S003	BobaDelight Supplies	011-4455667	79sana@bobadelight.com	120

Query #3 - To retrieve the total number of orders placed by each customer including customer information, including customer ID, name, contact, and email.

```
SELECT
    c.Customer_ID, c.Customer_Name, c.Customer_Contact, c.Customer_Email,
    COUNT(o.Order_ID) AS OrderCount
FROM
    customer c
LEFT JOIN
    orders o ON c.Customer_ID = o.Customer_ID
GROUP BY
    c.Customer ID, c.Customer Name, c.Customer Contact, c.Customer Email;
```

	CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_CONTACT	CUSTOMER_EMAIL	ORDERCOUNT
1	230592	John	017-2345678	johntan@gmail.com	2
2	230601	Ali	012-3456789	aliali@gmail.com	1
3	230602	Muthu	019-8765432	muthu@gmail.com	1

This query provides valuable information for implementing a loyalty program based on customer order frequency. It uses a **LEFT JOIN** to combine the "**customer**" and "**orders**" tables based on the common column "**Customer_ID**", ensuring that all customers are included in the result, regardless of whether they have placed orders. The **SELECT** clause specifies the columns to be retrieved, including customer details (**ID**, **name**, **contact**, and **email**) and the count of orders for each customer, aliased as "**OrderCount**." The **GROUP BY** clause groups the results by customer attributes, ensuring that the aggregation function (**COUNT**) operates on each customer. This provides insights into customers who are more frequent in making orders.

Query #4 - To retrieve the total revenue for each beverage in the specified year and month.

```
SELECT
    B.Bvg_ID,
    B.Bvg_Name,
    SUM(I.Bvg_Quantity * B.Selling_Price) AS TotalRevenue
FROM
    includes I
JOIN
    beverage B ON I.Bvg_ID = B.Bvg_ID
JOIN
    orders O ON I.Order_ID = O.Order_ID
WHERE
    EXTRACT(YEAR FROM O.Order_Date_Time) = 2024
    AND EXTRACT(MONTH FROM O.Order_Date_Time) = 1
GROUP BY
    B.Bvg_ID, B.Bvg_Name
ORDER BY
    TotalRevenue DESC;
```

	BVG_ID	BVG_NAME	TOTALREVENUE
1	M001	Panda Milk Tea	53.4
2	M002	Brown Sugar Boba Milk Tea	21.8
3	F001	Passionfruit Tea	15.8
4	F002	Grapefruit Boba Tea	8.9

As well as finding out the quantity sold for each beverage, users should also determine which beverage generates the highest revenue. This SQL code combines the **Includes**, **Beverage**, and **Orders** tables, filtering orders for a particular year (2023) and month (11). The **SUM** function computes the total revenue for each beverage, factoring in the product of quantity sold (**I.Bvg_Quantity**) and the beverage's selling price (**B.Selling_Price**). The results are grouped by **Bvg_ID** and **Bvg_Name**, with the output sorted in descending order of total revenue.

12.0 Screenshots of Tables from the Developed System

BEVERAGE

BVG_ID	BVG_NAME	SELLING_PRICE	BVG_TYPE
1 M001	Panda Milk Tea	8.9	Milk Tea
2 M002	Brown Sugar Boba Milk Tea	10.9	Milk Tea
3 F001	Passionfruit Tea	7.9	Fruit Tea
4 F002	Grapefruit Boba Tea	8.9	Fruit Tea

INGREDIENT

INGREDIENT_ID	INGREDIENT_NAME	SUPPLIER_ID	UNIT_COST	INGREDIENT_CATEGORY
1 IG001	Sugar	S001	10	A
2 IG004	Brown Sugar	S001	25	B
3 IG002	Milk	S002	30	C
4 IG005	Tea	S002	5	D
5 IG003	Boba	S003	20	E
6 IG006	Jelly	S003	12	F

SUPPLIER

SUPPLIER_ID	SUPPLIER_NAME	SUPPLIER_CONTACT	SUPPLIER_EMAIL	SUPPLIER_ADDRESS
1 S001	SweetHarvest	013-4455667	taeyeon@sweetharvest.com	123 Sugar Lane, Jalan Ipoh
2 S002	PureLeaf Teas	014-9955667	minnie@pureleaf teas.com	456 Tea Terrace, Jalan Johor
3 S003	BobaDelight Supplies	011-4455667	79sana@bobadelight.com	789 Boba Boulevard, Jalan Melaka

FOOD_DELIVERY

DELIVERY_ID	DELIVERY_DATE_TIME	DELIVERY_STATUS	PLATFORM_NAME
1 GF399-102	18-NOV-23 11.00.00.000000000	AM Delivered	GrabFood
2 FP456-901	16-NOV-23 03.40.00.000000000	PM Delivered	Foodpanda
3 FP487-928	14-NOV-23 08.40.00.000000000	PM Delivered	Foodpanda

EMPLOYEE_POSITION

EMPLOYEE_POSITION_ID	EMPLOYEE_POSITION_NAME	EMPLOYEE_PAY_RATE
1 P001	Tea Master	14.25
2 P002	Tea Blender	8.5
3 P003	Tea Packer	12
4 P004	Tea Quality Control	10.5
5 P005	Tea Sales Representative	13.5

CUSTOMER

	❖ CUSTOMER_ID	❖ CUSTOMER_NAME	❖ CUSTOMER_CONTACT	❖ CUSTOMER_EMAIL
1	230601	Ali	012-3456789	aliali@gmail.com
2	230602	Muthu	019-8765432	muthu@gmail.com
3	230592	John	017-2345678	johntan@gmail.com

INVENTORY

	❖ INVENTORY_ID	❖ INGREDIENT_ID	❖ DATE_CREATED	❖ EXPIRATION_DATE	❖ INGREDIENT_QUANTITY
1	IV001	IG001	25/08/2023	08/02/2024	100
2	IV002	IG002	25/08/2023	10/12/2023	50
3	IV003	IG004	04/09/2023	06/03/2024	100
4	IV004	IG002	15/09/2023	03/01/2024	75
5	IV005	IG001	18/09/2023	15/03/2024	150
6	IV006	IG003	08/10/2023	08/03/2024	50
7	IV007	IG005	12/10/2023	12/10/2024	100
8	IV008	IG001	05/11/2023	09/04/2024	100
9	IV009	IG006	06/11/2023	07/03/2024	120
10	IV010	IG002	20/11/2023	23/02/2024	100

MADE

	❖ INGREDIENT_ID	❖ BVG_ID
1	IG001	M001
2	IG002	M001
3	IG003	M001
4	IG001	M002
5	IG002	M002
6	IG003	M002
7	IG004	M002
8	IG001	F001
9	IG005	F001
10	IG001	F002
11	IG005	F002
12	IG003	F002

EMPLOYEE

	❖ EMPLOY...	❖ EMPLOYEE_NAME	❖ EMPLOYEE_NRIC	❖ EMPLOYEE_BANKDETAILS	❖ EMPLOYEE_CONTACT	❖ EMPLOYEE_GENDER	❖ EMPLOYEE_HIRE_DATE	❖ EMPLOYEE_POSITION_ID	❖ EMPLOYEE_WORKING_HOURS
1	E001	John Doe	1234567890	60146400236236	60-1155577037	Male	04-NOV-23	P001	45
2	E004	Sarah Tan	2134567890	60146400236233	60-1155577040	Female	09-NOV-23	P004	30
3	E005	David Lee	3456789012	60146400236232	60-1155577041	Male	10-NOV-23	P005	40
4	E007	Mariah Tan	4569971236	60146400236230	60-1155577043	Female	12-NOV-23	P003	20

ORDERS

ORDER_ID	EMPLOYEE_ID	CUSTOMER_ID	ORDER_DATE_TIME	DELIVERY_ID	PAYMENT_METHOD_NAME	PAYMENT_AMOUNT
1	1001 E001	230601	19-NOV-23 11.30.00.0000000000 AM	(null)	Cash	26.7
2	1002 E001	230602	18-NOV-23 10.30.00.0000000000 AM	GF399-102	E-wallet	8.9
3	1003 E004	230592	16-NOV-23 03.00.00.0000000000 PM	FP456-901	Credit/Debit Card	50
4	1004 E004	230592	14-NOV-23 08.00.00.0000000000 PM	FP487-928	Credit/Debit Card	40

INCLUDES

ORDER_ID	BVG_ID	BVG_QUANTITY
1	1001 M001	2
2	1001 F002	1
3	1002 M001	1
4	1003 M001	3
5	1003 M002	2
6	1003 F001	2
7	1004 F001	1

SHIFT_SCHEDULE

SHIFT_ID	EMPLOYEE_ID	SHIFT_DATE	SHIFT_START_TIME	SHIFT_END_TIME	SHIFT_TYPE
1 S001	E001	13-NOV-23	01-JAN-24 08.00.00.0000000000 AM	01-JAN-24 05.00.00.0000000000 PM	Morning Shift
2 S002	E004	13-NOV-23	01-JAN-24 12.00.00.0000000000 PM	01-JAN-24 09.00.00.0000000000 PM	Afternoon Shift
3 S003	E005	13-NOV-23	01-JAN-24 10.00.00.0000000000 AM	01-JAN-24 07.00.00.0000000000 PM	Day Shift
4 S004	E007	13-NOV-23	01-JAN-24 12.00.00.0000000000 PM	01-JAN-24 09.00.00.0000000000 PM	Afternoon Shift
5 S005	E001	14-NOV-23	01-JAN-24 08.00.00.0000000000 AM	01-JAN-24 05.00.00.0000000000 PM	Morning Shift

EMPLOYEE_ATTENDANCE

ATTENDANCE_ID	ATTENDANCE_DATE	EMPLOYEE_ID	CLOCK_IN_TIME	CLOCK_OUT_TIME	ATTENDANCE_STATUS
1 A001	12/11/2023	E001	01/01/2024 08:00:00.0000000000	01/01/2024 17:00:00.0000000000	Present
2 A002	13/11/2023	E004	01/01/2024 12:00:00.0000000000	01/01/2024 21:00:00.0000000000	Present
3 A003	13/11/2023	E005	01/01/2024 10:00:00.0000000000	01/01/2024 19:00:00.0000000000	Not Present
4 A004	13/11/2023	E007	01/01/2024 12:00:00.0000000000	01/01/2024 21:00:00.0000000000	Not Present
5 A005	14/11/2023	E001	01/01/2024 08:00:00.0000000000	01/01/2024 17:00:00.0000000000	Present

12.0 Discussion

12.1 Challenges Faced by Group

Our group faced difficulty at the beginning stage of the project as we found it hard to find a collaborator who willingly cooperated with us. We had approached several companies but with no response or reply. At first, our group chose He&She Coffee as the collaborator because the store is at the University of Malaya so it is easier for us to conduct the interview. However, we changed our collaborator because we did not receive any responses from the organisation. Then we tried to find another collaborator and finally, we decided to choose Presotea as our collaborator.

Apart from that, we find it quite challenging to generate questions to ask the interviewer about the database system. This is because we need to think of interview questions that could help us build this database system. However, we overcame this difficulty by conducting meetings to share ideas and working on the interview questions together.

Other than that, our group faced hardship in implementing the concept of the database as we had limited knowledge of normalisation. However, in Week 5 we had a lecture on the topic of normalisation and tried to understand the concept of it. Fortunately, by utilising lecture slides and online resources such as YouTube videos on normalisation and the Oracle database, we improved our grasp of the topic and successfully implemented the database in Oracle, overcoming the challenges faced earlier in the project.

On top of that, we also found it challenging when we started to write the SQL statements and develop the database in Oracle. This is because we haven't familiarised ourselves with the Oracle as it was our first time using it. We made many errors writing the SQL statement and also had problems when trying to run the SQL queries. We overcame it by searching videos on YouTube for tutorials and also from any website that might help us.

12.2 Lessons Learnt

Throughout this project, our group has learned to adapt to difficult conditions and handle problems with remarkable cooperation among team members. When new ideas, disagreements, or difficulties develop, we normally discuss them in our WhatsApp group conversation or meet in person to resolve them.

We have learned that not everything goes as planned, so we should always have a backup plan in case our initial plan fails. An example from the project would be when we initially wanted to work with the He and She cafe for the project, but we failed to do so. Then we learned how to effectively utilise the time and plan an effective meeting to discuss our project. We usually prepare what we want to discuss ahead of time so that we can get right to the point during the meeting. Despite this, we were able to complete the conceptual and logical design of the database based on some background research. We conducted an interview for us to understand the business rules of how the Presotea Cafe operation works and their requirements.

Aside from that, we have learned that to design a database, it is crucial to understand the business requirements, including the types of data, the relationships between the data, and how the data will be used. A consistent naming convention should be used for tables, columns, and other database objects to make the database easier to understand and maintain. Documentation is essential to understand and maintain the database, including the schema, relationships between tables, and data types used. The database should be designed for scalability to accommodate future business growth by using appropriate data types, indexes, and partitioning. We were able to apply database knowledge learned throughout the course. We successfully constructed an Entity Relationship Diagram with their respective entities and attributes and carried out the normalisation process. We were also able to use DDL and DML queries in our database.

Overall, the project was completed successfully and smoothly, thanks to the insightful knowledge from our lecture notes and tutorial, our very helpful lecturer, Dr. Norjihan binti Abdul Ghani, and, most importantly, the team members' cooperation and clear communication.

13.0 Future Improvement

BACKUP AND RECOVERY

- Regularly test the backup and recovery processes to verify their effectiveness.
- This procedure is crucial to prevent data loss if accidents happen such as hardware failures, software bugs, human errors, or even malicious activities like hacking or ransomware attacks.
- Regularly testing and validating backups help ensure that the backup data is complete, accurate and can be successfully restored.

EASY SEARCHING

- Create a database that has appropriate indexes on the columns of frequently searched data.
- Helps significantly speed up searches.

SCALING

- Ensure the current database architecture can handle the expected growth in data and user load.
- Consider using scaling options like vertical scaling or horizontal scaling.

DATA TYPES

- Use appropriate data types for each column to minimise storage requirements and improve query performance.
- Avoid using generic data types when more specific ones are available.

SECURITY

- Review and enhance database security measures, including access controls.
- Keep database software and security patches up-to-date.

COLLABORATIVE TOOLS FOR DEVELOPMENT

- Utilise collaboration development tools to streamline database development
- Everyone in the team can collaborate to develop the database.

PERFORMANCE OPTIMIZATION

- Investigate and implement strategies to optimise database performance, especially for frequently queried tables or operations.

Appendix

i. Contributions by Each Member

Member	Task
Teo Wen Thing	<ul style="list-style-type: none">● Met up with the Outlet Manager together with group members to interview him on the database entities and attributes and help to record the information given.● Summarising the content of the interview to come up with the Entities and Attributes.● Design the ERD● Helped in identifying the relationships between entities to develop business rules.● Worked on the normalisation process.● Presenter for Presentation 1● Oversee and manage the overall progress of the group for our database project.● Wrote and tested SQL statements.● Group leader for the project
Phua Wen Teng	<ul style="list-style-type: none">● Met up with the Outlet Manager together with group members to interview him on the database entities and attributes and help to record the information given.● Design the ERD.● Helped in identifying the relationships between entities to develop business rules.● Helped on the normalisation process.● Presenter for Presentation 1● Wrote and tested SQL statements.
Hon Yao Zhi	<ul style="list-style-type: none">● Met up with the Outlet Manager together with

	<p>group members to interview him on the database entities and attributes and help to record the information given.</p> <ul style="list-style-type: none"> • Summarising the content of the interview • Design the ERD • Helped in identifying the relationships between entities to develop business rules. • Helped with the normalisation process. • Wrote and tested SQL statements. • Presenter for Presentation 1 and 2
Ameera Binti Omar	<ul style="list-style-type: none"> • Helped to prepare presentation slide • Helped with the normalisation process. • Helped in report writing • Helped with ERD • Developed the database system. • Loading data into the project database. • Preparing DDL and DML queries. • Wrote and tested SQL statements. • Presenter for Presentation 2
Nadhea Binti Ismail	<ul style="list-style-type: none"> • Helped to prepare presentation slide • Helped with the normalisation process. • Helped with ERD • Helped in report writing • Wrote and tested SQL statements.

ii. Meeting with Collaborator



iii. Letter to Company

Teo Wen Thing
Faculty of Computer Science & Information Technology
University of Malaya
50603 Kuala Lumpur
Malaysia

Sir/Madam
Outlet Manager
Presotea SS2
46 Jalan SS2/61
SS2, Petaling Jaya, 47300
Selangor



Date 30/10/2023

FW - Foo Wai Loon
1/11/2023

Dear Sir/Madam,

Request To Presotea SS2 As A Model for Database Project

I am pursuing my Bachelor's Degree in Computer Science/ Information Technology at the University of Malaya. Design and development of a database is one of the assignments which we have to complete as a group project. I am writing on behalf of my group members to enquire whether you would allow us to use **Presotea SS2** as our model for studying the needs for a database.

For the database project, we would like to understand the workflow and data flow of your company and we do not need to go into details or to see any confidential parts of your system or your data. We would also appreciate your acknowledging our discussions by signing our report.

We sincerely hope that you will agree to meet with us and assist us in understanding the flow of data and the needs for data management in your company. This will help us to complete our database project. We look forward to receiving a positive and prompt reply.

Thank you.

Yours sincerely,

wthing

(Teo Wen Thing)

TEL: 019-2188442

Email: wenthingteo2@gmail.com

Database Lecturer,

DR. NORJIHAN BINTI ABDUL GHANI

(PROFESOR MADYA DR. NORJIHAN

BINTI ABDUL GHANI)

iv. Organisation Verification

Organisation Agreement Letter

I, the Outlet Manager of Presotea SS2, hereby provide my formal agreement to serve as a study model for the database group project led by Teo Wen Thing. The project is for the WIA2001 Database course at Universiti Malaya, focusing on the design, development, and optimisation of a comprehensive database system. By participating in this project, we aim to offer our company work flow that will assist the group in achieving their academic objectives and contribute to the enhancement of our database system, aligning it with the specific needs of Presotea SS2. We are excited about this collaborative endeavour and look forward to the knowledge exchange and innovations that will benefit both our business and the academic community.

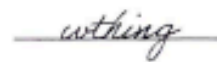
Outlet Manager,



Name: Foo Wa. Loon

Date: 1/11/2023

Project Leader,



Name: TEO WEN THING

Date: 30/10/2023

v. Peer Work Group Evaluation Form

[Peer Work Group Evaluation Form Folder\(T7 Group8\)](#)