

**Database (WIA2001)**

**Semester 1 2021/22**

**Tutorial Group** : K1 T4

**Lecturer** : DR. FARIZA HANUM BINTI MD NASARUDDIN

**Group** : 6   
**Group Members** :

| **Name** | **Matric ID** |
| --- | --- |
| **FRANKIE LIM QI QUAN (Leader)** | **U2005263/1** |
| **PUA JING YI** | **U2005396/1** |
| **GOH CHEE LAM** | **U2005382/1** |
| **KELVIN CHEAH** | **U2005394/1** |
| **FARISAH ATHIRA BINTI MD ZAMRI** | **17206915/2** |

**Table of Contents**

[**Report 1**](#_k1ppa227e6nk) **1**

[**1.0 Introduction**](#_drfy7mf9qvhl) **1**

[**2.0 Issues in Managing Data**](#_tzrryu5935js) **2**

[**3.0 Business Rules and Assumptions**](#_ivb7luqpkykr) **3**

[**4.0 System Objectives and Scope**](#_88p9qbt3ouq) **5**

[**5.0 User Requirements**](#_z0pvmi46d7bs) **7**

[**6.0 Entities and Attributes Before Normalization**](#_9ih9emo5l0e9) **8**

[**7.0 Relationship, Cardinalities And Constraints**](#_6a8b27yo1iwi) **10**

[**8.0 Conceptual Entity Relationship Diagram**](#_x2rotavp1g8v) **11**

[**9.0 Normalization Process**](#_8eg3golhfj4i) **12**

[9.1 Table Schemas:](#_w9emmla41ngs) 12

[9.2 UNNORMALIZED FORM (UNF)](#_83gfoyxqj0wf) 13

[9.3 FIRST NORMAL FORM (1NF)](#_egpmt56eldte) 15

[9.4 SECOND NORMAL FORM ( 2NF )](#_3e8mvwno0g3i) 17

[9.5 THIRD NORMAL FORM ( 3NF )](#_oj2fus6x7ux7) 22

[**10.0 Implementable Entity Relationship Diagram**](#_r1ndq6kec8ui) **24**

[**11.0 Entities and Attributes After Normalization**](#_8o37r0wrfw6x) **25**

[**12.0 Challenges Faced During Data Collection Process and How To Overcome**](#_jx1fmzchcqi4) **28**

[**13.0 Conclusion**](#_m9x6tx8lqbfe) **28**

[**14.0 Proof of The Interview/Data Collection**](#_p50bt6y5gifn) **29**

[14. 1 Cop:](#_hhduyscnn78l) 29

[14.2 Online Interview With Manager On Google Meet:](#_vmi2c1ryu40u) 29

[14.3 Photos with Employer and Manager:](#_qo3eymol4xap) 30

[14.4 Sample Receipt:](#_yl9he0np5loo) 31

[14.5 Name Card:](#_c5zbc53rbr0e) 31

[**Report 2**](#_5djebz646gje) **32**

[**1.0 Database Design**](#_yqhyw8evpdqb) **32**

[1.1 Introduction](#_mb4gd9v44isn) 32

[1.2 Overview of ERD](#_efizbay9sngf) 34

[1.3 Code](#_r3ekan92czu) 35

[1.3.1 Customers](#_sdb69ctph1no) 35

[1.3.2 Jobs](#_pgwd45gngor0) 37

[1.3.3 Payment\_Methods](#_1ynhn259l969) 38

[1.3.4 Supplier](#_zfugyilrz3vx) 40

[1.3.5 Ingredients](#_mc2jgrw09u4u) 41

[1.3.6 Menu](#_mbaqixv5u63d) 45

[1.3.7 Vouchers](#_3wrf1teuj3ch) 47

[1.3.8 Menu\_Ingredients](#_pgril96v3y9) 52

[1.3.9 Employees](#_wgwdfxlvk56y) 57

[1.3.10 Delivery](#_9f5tcdbf00je) 61

[1.3.11 Orders](#_ve5ct4kdz0p9) 62

[1.3.12 Order\_Menu](#_1p6tsab6r4dp) 65

[1.3.13 Supply](#_andvnuymv50w) 70

[1.3.14 Payments](#_vyozagv1trwc) 74

[**2.0 SQL Statement And Description**](#_ew7thbukubcp) **78**

[2.1 ALTER](#_ifynd0rnxpy1) 78

[2.2 LIKE, WHERE](#_pzclqgwnsxdl) 80

[2.3 IN](#_qprs2u3ew2rj) 81

[2.4 DELETE](#_5hdtduns3wdm) 82

[2.5 UPDATE](#_4sfkdjm3rbnh) 84

[2.6 INSERT INTO](#_v07ktbwq22ja) 85

[2.7 DISTINCT](#_s1zrq6r3dabl) 86

[2.8 DISTINCT, COUNT(), ALIASES](#_knac3c13r51a) 87

[2.9 ANY()](#_nxjqm2ei57p6) 88

[2.10 UNION](#_5evaw18w5ej3) 91

[2.11 MIN(), BETWEEN](#_pv1tq1759ro8) 93

[2.12 MAX(), BETWEEN](#_nlhlpt4i5amu) 94

[2.13 SELECT, INNER JOIN](#_jl6x4u9rwq3x) 95

[2.14 EXIST, HAVING, COUNT(), GROUP BY](#_tlybxfp0ciu0) 96

[2.15 SELECT, MULTIPLY (SQL Arithmetic Operator), ALIASES, LEFT JOIN, ORDER BY, SUM(), GROUP BY](#_1b7y7dcv6hgy) 98

[2.16 SELECT, MULTIPLY (SQL Arithmetic Operator), ALIASES, INNER JOIN, ORDER BY, DESCENDING](#_3myvk7goezvg) 103

[2.17 SELECT, SUM(), ALIASES, ADD (SQL Arithmetic Operator), INNER JOIN, WHERE, ORDER BY, GROUP BY](#_pvcakxcqfw4q) 104

[2.18 SELECT, COUNT(), SUM(), AVG(), ALIASES, LEFT JOIN, GROUP BY, ORDER BY, DESCENDING](#_9l5x63w06t8e) 108

[**2.0 Database Testing**](#_9hhfx13bjwo6) **110**

[**3.0 Difficulties and Problems**](#_mxd7a4poqv3r) **111**

[3.1 Problems:](#_500hbro7wl9b) 111

[3.2 Solutions:](#_5ou91wavyeri) 111

[3.3 Future improvement:](#_zskvli69z7d) 111

[3.4 Lesson learnt:](#_vtdlttyqlkcz) 112

[**4.0 Peer Work Group Evaluation Form**](#_ypd3g0n4apqy) **113**

# **Report 1**

# **1.0 Introduction**

A database design for a cafe called Ke Nina Cafe that sells drinks and food to its customers and is located at Bandar Baru, Kuala Kangsar, Perak. Its operation hours are from 10 am to 9 pm daily. The cafe focuses on milk tea drinks and western foods. Customers are allowed to order either through deliveries or walk-in. There are 4 job positions which are the cashier, chef, worker who prepares drinks and delivery man plus one Manager that manages the workers. The workers work 1 shift per day (during MCO) with 1 holiday per week. Then, the cafe uses a point of sales system, which allows them to identify the most popular food in the cafe. It also allows them to generate reports containing the revenue and net profit earned to facilitate their business decisions making. Hence, they use cloud servers to store business data.

The problems currently exist by the cafe are they do not have a proper database that records their workers working hours and can automatically calculate total salary per week. They still use manual calculation where it may lead to miscalculation and is a tedious process. Other than that, they are using spreadsheets like Excel to keep track of the food ingredients left. This means the workers will need to count the ingredients left every day and update it in the spreadsheet where it may lead to data anomalies. Plus, 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 system to help them manage all the data.

The solution to the problems stated above is to create a proper relational database for them to record every worker's working hours and the food ingredients. After implementing a proper database, the total salary of each worker and the number of ingredients left can be checked with just one click.

# **2.0 Issues in Managing Data**

1. The cafe does not have a proper database system to record their working hours and the one that can automatically calculate the total salary per week.
2. Using spreadsheets like Excel to keep track of the food ingredients left, where the worker will need to count the ingredients every day and update it in the spreadsheet where it may lead to data anomalies.
3. The workers record the customers’ orders on a piece of paper. During the peak hours, the handwriting of the workers becomes ugly and hard to read by the other workers. Besides that, there is the possibility that the workers miss some of the orders.

# **3.0 Business Rules and Assumptions**

**Business Rules:**

* One customer can place one or many orders.
* One order is placed by one and only one customer.
* One employee can handle zero orders.
* One employee can handle many orders.
* One order can be handled by one and only one employee.
* One order has one or many Menu.
* One Menu can be found in zero orders.
* One Menu can be found in many orders.
* One Menu has one or many ingredients.
* One ingredient belongs to one or many Menu.
* One supplier can supply one or many ingredients.
* One ingredient is supplied by one and many suppliers.
* One order belongs to one and only one payment.
* One payment has one and only one order.
* One payment uses zero voucher.
* One payment uses one and only one voucher.
* One voucher is used in one and only one payment.
* One payment is processed by one and only one employee.
* One employee can process zero payments.
* One employee can process many payments.
* One delivery has one and only one order.
* One order is found in zero delivery.
* One order is found in one and only one delivery.
* One delivery is delivered by one and only one employee.
* One employee can deliver zero deliveries.
* One employee can deliver many deliveries.

**Assumptions:**

* The customer must place at least one order in the restaurant.
* The delivery address is the same as the customer address.
* The customer who places the order also makes the payment at the counter.
* The currency used is Ringgit Malaysia(RM).
* The customer must give basic information before placing the order.
* The unit price of ingredients is calculated using the supply total price divided by the supply quantity.
* There is no rounding of the payment amount for the cash payment method.

# **4.0 System Objectives and Scope**

Ke Nina Cafe was a cafe located in No.12, Persiaran Bendahara 1, Bandar Baru, 33000 Kuala Kangsar, Perak. The cafe currently has only one main restaurant but is expecting to open a new branch soon at a strategic location to expand their business.

The cafe has about 4 staff working as the cashier, chef, the worker who prepares drinks and a delivery man, with 1 shift per day during Movement Control Order(MCO) from 9 am to 10 pm. There will be 2 workers assigned as chefs in the kitchen, 2 workers serving the customers and preparing the drink and another 2 workers assigned to deliver food to the customer. Also, a manager is assigned and responsible for the restaurant's daily operations. Usually, the manager will supervise other workers in the restaurant and order ingredients from their suppliers weekly.

Then, each customer that comes in will be given a paper to write the food and drinks they ordered by writing down the code. This method is used by the cafe to record and keep track of the order. Ke Nina Cafe has each food and drinks to be uniquely identified by assigning different codes for different types of food. Besides that, before ordering food, a customer needs to provide their basic information like their name so that the system can store the data for future reference. Plus, there are also deliveries provided for the customers via the cafe delivery, FoodPanda and GrabFood so that the user can choose any purchase method they want.

Other than that, customers can choose either using cash, e-wallet, debit or credit cards for the payment process. The cafe hopes that its customers can pay conveniently and have the best dine-in or take-out experiences without worrying about the payment. Furthermore, the cafe also provides 15% discounts (drinks only) for customers who had loyalty cards or vouchers. Each voucher has a unique code to be identified and can be used for one time only with no minimum spend requirements for each purchase. The coupon discount rate is fixed at 15 %, but on certain national holidays like Hari Kebangsaan, the discount rate may increase.

Next, moving on to the management of the ingredients, the worker needs to count how many ingredients are left every day and update it in the Excel spreadsheet. This is considered to be time-consuming and may lead to data anomalies, where the cafe manager hopes they have a proper database system to take care of it. The cafe has around 6 suppliers that will supply the ingredients weekly. The manager is the one in charge of the order and the amount will depend on the quantity of the ingredients left.

The cafe is still using a punch card system to keep track of the workers working hours. However, the salary is calculated manually by referring to the record on the punch card. The process is so tedious, where the owner of the cafe hopes that it can be handled automatically by a system, as it is easy to make mistakes by manual calculation. Then, the cafe is using a cloud server to generate reports about the revenue and net profit earned by the business. In brief, as Ke Nina Cafe may grow in the future, they may have difficulties in managing the increasing amount of data used and generated by the cafe. To ensure the continued success of the cafe, a database system is needed to help solve the increasing problems of data management.

# **5.0 User Requirements**

Employer has the highest access to the database. Maintaining data can be done by entering, updating and deleting data. The employer can maintain data and records on the restaurant and employees. Employers can maintain data and records on the menu, ingredients and suppliers. The employees can record the data of payments, vouchers, customers, orders and delivery.

Searching data and records can be done by the employer and employees. The employer can perform searches on employees, menu, ingredients and suppliers. The employee can perform searches on payments, vouchers, customers, orders and delivery.

Tracking data and records can be done by the employer, manager and employees. The employer is able to check employees and the salary of employees according to their job type and working hours. The employer is also able to track the information of ingredients and the information of suppliers. Besides, the employer is able to check their restaurant’s revenue, expenditures and information about the payment methods. Next, the employee is able to track the status of the payments, the status of vouchers, the status of delivery and check the status of orders.

Generating reports can be done only by the employer. The employer can generate reports on sales of the restaurant and the salary of employees. For example, the employer can calculate the total payments by using each payment method that is made by the customers and calculate the number of payments using each payment method. The employer can also generate reports on payments, the status of ingredients and orders from customers. By checking the number of orders made by each customer, the employer can get insights from the reports about the loyalty of the customers and some of the most popular menu items.

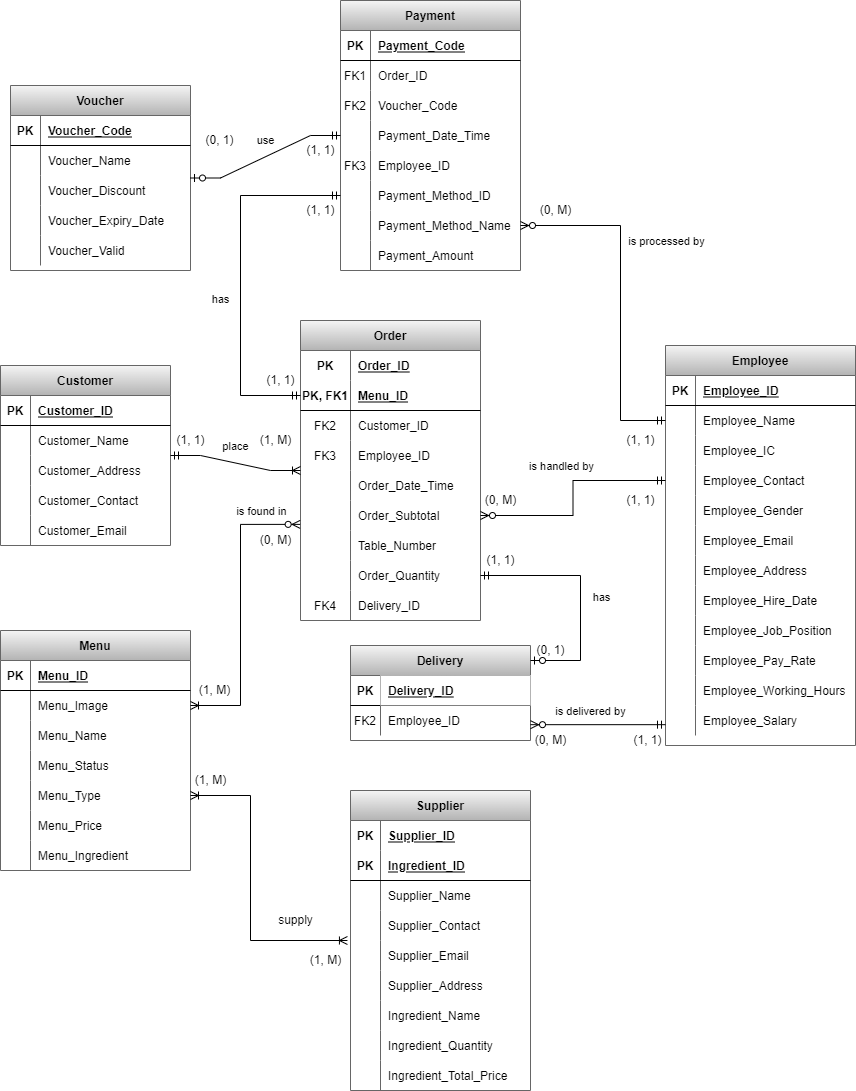
# **6.0 Entities and Attributes Before Normalization**

| **Entity** | **Attributes** |
| --- | --- |
| Customer | **Customer\_ID**  Customer\_Name  Customer\_Address  Customer\_Contact  Customer\_Email |
| Order | **Order\_ID**  **Menu\_ID**  Customer\_ID  Employee\_ID  Order\_Date\_Time  Order\_Subtotal  Table\_No  Order\_Quantity  Delivery\_ID |
| Menu | **Menu\_ID**  Menu\_Image  Menu\_Name  Menu\_Status  Menu\_Type  Menu\_Price  Menu\_Ingredient |
| Supplier | **Supplier\_ID**  **Ingredient\_ID**  Supplier\_Name  Supplier\_Contact  Supplier\_Email  Supplier\_Address  Ingredient\_Name  Ingredient\_Quantity  Ingredient\_Total\_Price |
| Payment | **Payment\_Code**  Order\_ID  Voucher\_Code  Payment\_Date\_Time  Employee\_ID  Payment\_Method\_ID  Payment\_Method\_Name  Payment\_Amount |
| Voucher | **Voucher\_Code**  Voucher\_Name  Voucher\_Discount  Voucher\_Expiry\_Date  Voucher\_Valid |
| Employee | **Employee\_ID**  Employee\_Name  Employee\_IC  Employee\_Contact  Employee\_Gender  Employee\_Email  Employee\_Address  Employee\_Hire\_Date  Employee\_Job\_Position  Employee\_Pay\_Rate  Employee\_Working\_Hours  Employee\_Salary |
| Delivery | **Delivery\_ID**  Employee\_ID |

# **7.0 Relationship, Cardinalities And Constraints**

* One employee can have only one IC.
* The contact number does not contain a country code.
* The ingredient expiry date is not tracked.
* The customer needs to order something in the restaurant.
* The customer must use only one payment method for each order payment.
* Vouchers can be used only once for payment.
* Only one person will order ingredients from the suppliers.
* The order will be delivered to the customer address.

# **8.0 Conceptual Entity Relationship Diagram**



# **9.0 Normalization Process**

## **9.1 Table Schemas:**

EMPLOYEE(**Employee\_ID**, Employee\_Name, Employee\_IC, Employee\_Contact, Employee\_Gender, Employee\_Email, Employee\_Address, Employee\_Hire\_Date, Employee\_Job\_Name, Employee\_Pay\_Rate, Employee\_Working\_Hours, Employee\_Salary)

CUSTOMER(**Customer\_ID**, Customer\_Name, Customer\_Contact, Customer\_Address, Customer\_Email)

MENU(**Menu\_ID**, Menu\_Name, Menu\_Image, Menu\_Status, Menu\_Type, Menu\_Price, Menu\_Ingredient)

ORDER(**Order\_ID**, Order\_Date\_Time. Customer\_ID, **Menu\_ID**, Order\_Quantity, Order\_Subtotal, Table\_No, Employee\_ID, Delivery\_ID)

PAYMENT(**Payment\_Code**, Order\_ID, Payment\_Date\_Time, Employee\_ID, Payment\_Method, Voucher\_Code)

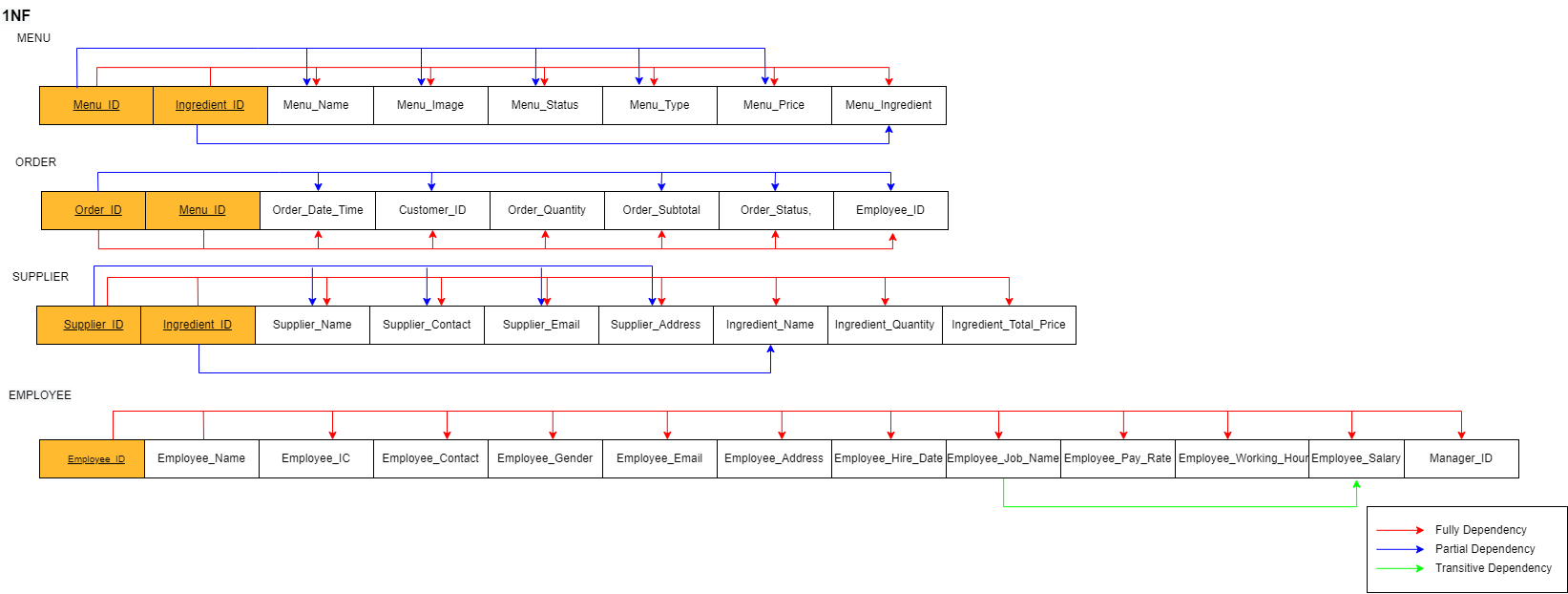
SUPPLIER(**Supplier\_ID**, Supplier\_Name, Supplier\_Contact, Supplier\_Email, Supplier\_Address, **Ingredient\_ID**, Ingredient\_Name, Ingredient\_Quantity, Ingredient\_Total\_Price)

VOUCHER(**Voucher\_Code**, Voucher\_Name, Voucher\_Discount, Voucher\_Expiry\_Date)

DELIVERY(**Delivery\_ID**, Employee\_ID)

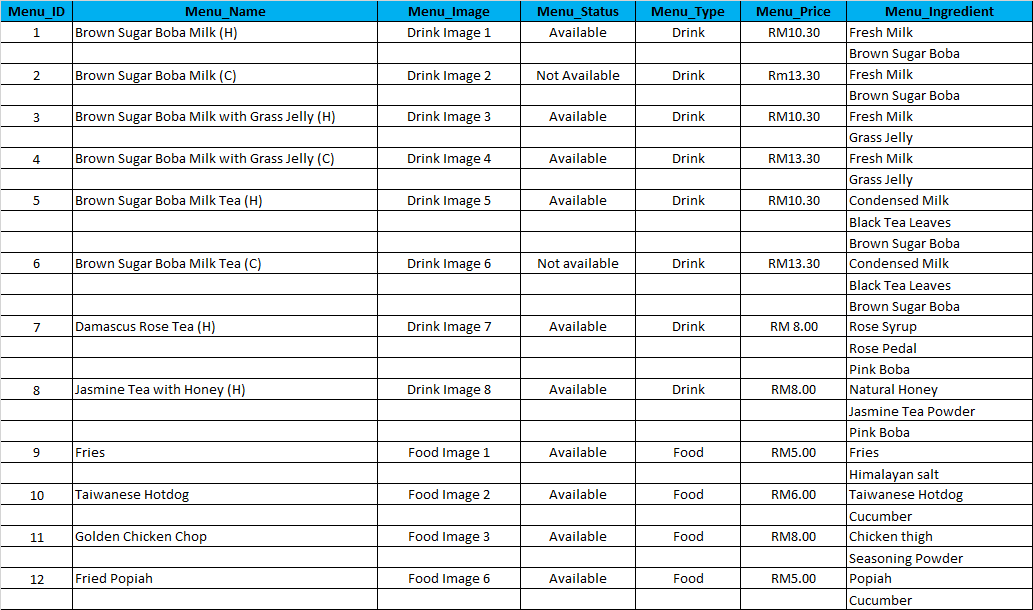
## **9.2 UNNORMALIZED FORM (UNF)**

According to the Entity-Relationship Diagram (ERD) above, it is still in an Unnormalized Form because it contains repeating groups. The tables that contain the repeating groups are MENU, ORDER and SUPPLIER. Below are the examples of the repeating group.



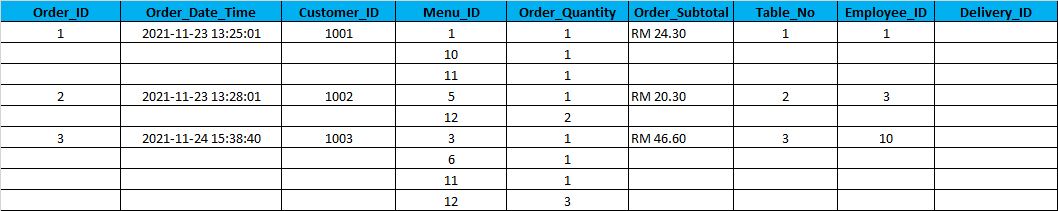
**UNF Menu Table:**

MENU(**Menu\_ID**, Menu\_Name, Menu\_Image, Menu\_Status, Menu\_Type, Menu\_Price, Menu\_Ingredient)



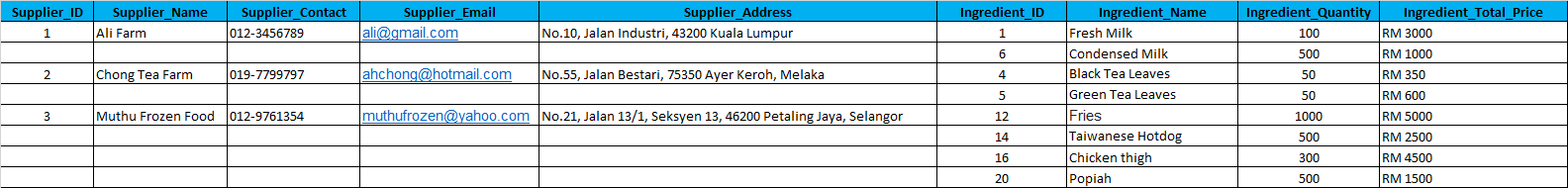
**UNF Order Table:**

ORDER(**Order\_ID**, Order\_Date\_Time, Customer\_ID, **Menu\_ID**, Order\_Quantity, Order\_Subtotal, Table\_No, Employee\_ID, Delivery\_ID)



**UNF Supplier Table:**

SUPPLIER(**Supplier\_ID**, Supplier\_Name, Supplier\_Contact, Supplier\_Email, Supplier\_Address, **Ingredient\_ID**, Ingredient\_Name, Ingredient\_Quantity, Ingredient\_Total\_Price)



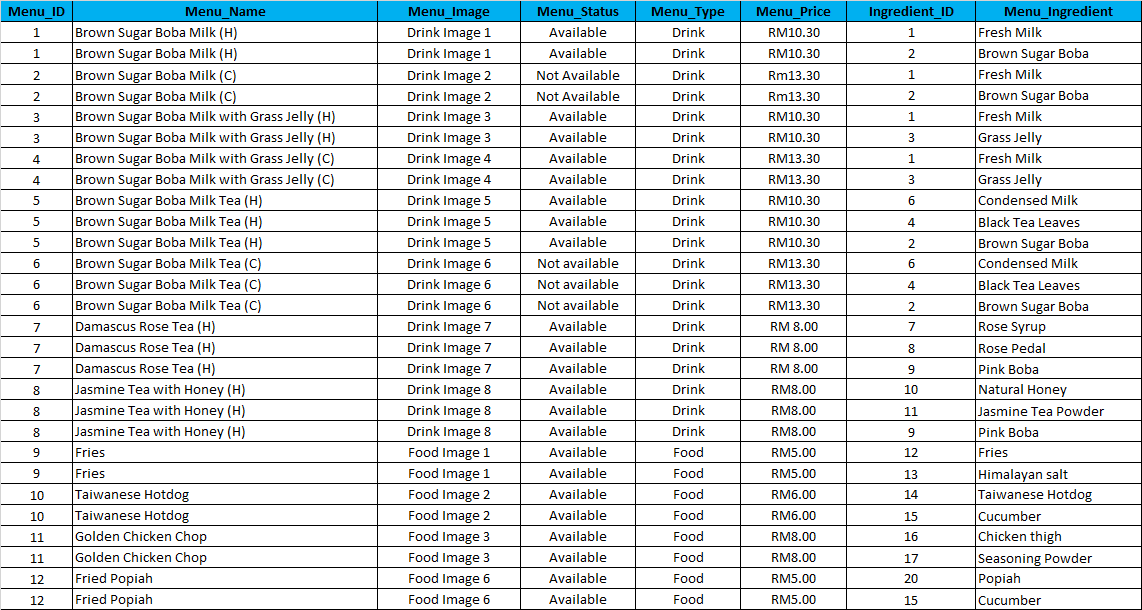
## **9.3 FIRST NORMAL FORM (1NF)**

To convert UNF into 1NF, we need to identify the primary key and all dependencies. We also need to eliminate the repeating groups.

To eliminate the above repetition, each table must be split into multiple tables.

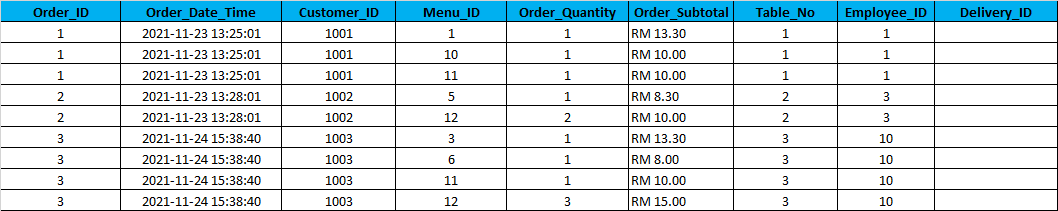
**1NF Menu Table:**

MENU(**Menu\_ID**, Menu\_Name, Menu\_Image, Menu\_Status, Menu\_Type, Menu\_Price, **Ingredient\_ID**, Menu\_Ingredient)



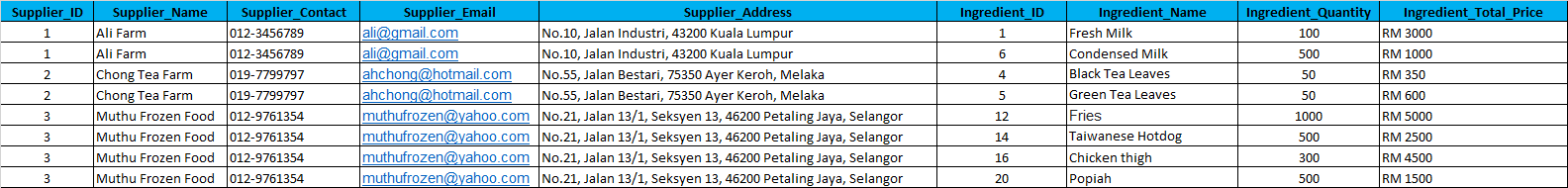
**1NF Order Table:**

ORDER(**Order\_ID**, Order\_Date\_Time, Customer\_ID, **Menu\_ID**, Order\_Quantity, Order\_Subtotal, Table\_No, Employee\_ID, Delivery\_ID)



**1NF Supplier Table:**

SUPPLIER(**Supplier\_ID**, Supplier\_Name, Supplier\_Contact, Supplier\_Email, Supplier\_Address, **Ingredient\_ID**, Ingredient\_Name, Ingredient\_Quantity, Ingredient\_Total\_Price)



## **9.4 SECOND NORMAL FORM ( 2NF )**

To convert 1NF to 2NF, we need to eliminate partial dependencies and reassign corresponding dependent attributes in new tables.

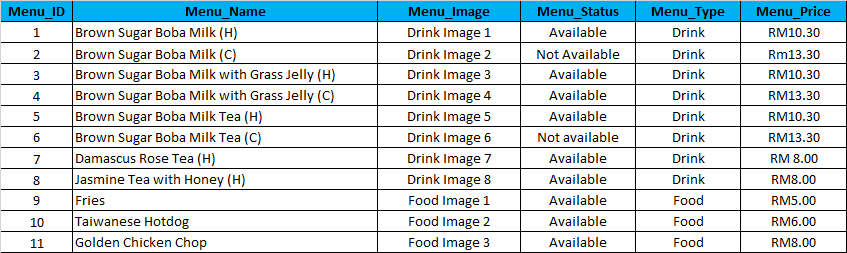
**Partial Dependencies:**

**Menu\_ID** -> Menu\_Name, Menu\_Image, Menu\_Status, Menu\_Type, Menu\_Price

**Ingredient\_ID** -> Menu\_Ingredient

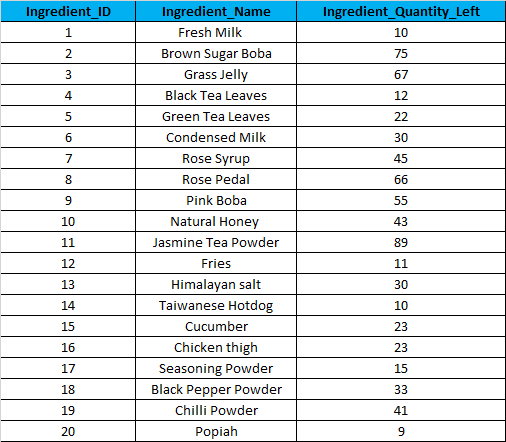
**2NF Menu Table:**

MENU(**Menu\_ID**, Menu\_Name, Menu\_Image, Menu\_Status, Menu\_Type, Menu\_Price)



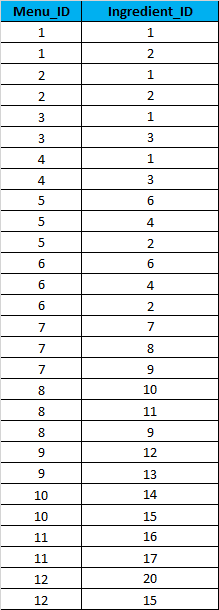
**2NF Ingredient Table:**

INGREDIENT(**Ingredient\_ID**, Ingredient\_Name, Ingredient\_Quantity\_Left)



**2NF Menu\_Ingredient Table:**

MENU\_INGREDIENT(**Menu\_ID**, **Ingredient\_ID**)

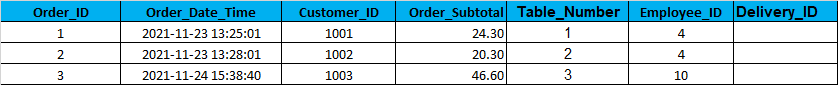


**Partial dependency:**

**Order\_ID** -> Order\_Date\_Time, Customer\_ID, Order\_Subtotal, Table\_No, Employee\_ID, Delivery\_ID

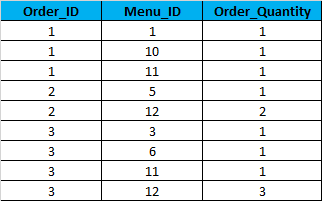
**2NF Order Table:**

ORDER(**Order\_ID**, Order\_Date\_Time, Customer\_ID, Order\_Subtotal, Table\_No, Employee\_ID, Delivery\_ID)



**2NF Order\_Menu Table:**

ORDER\_MENU(**Order\_ID**, **Menu\_ID**, Order\_Quantity)

****

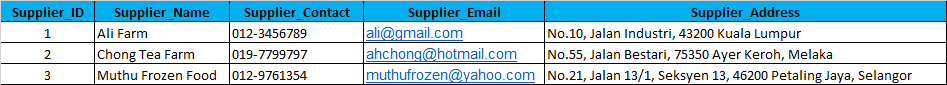
**Partial Dependencies:**

**Supplier\_ID** -> Supplier\_Name, Supplier\_Contact, Supplier\_Email, Supplier\_Address

**Ingredient\_ID** -> Ingredient\_Name

**2NF Supplier Table:**

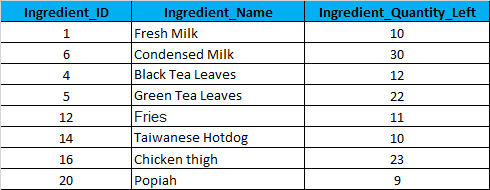
SUPPLIER(**Supplier\_ID**, Supplier\_Name, Supplier\_Contact, Supplier\_Email, Supplier\_Address)



**2NF Ingredient Table:**

Add new attribute called Ingredient\_Quantity\_Left

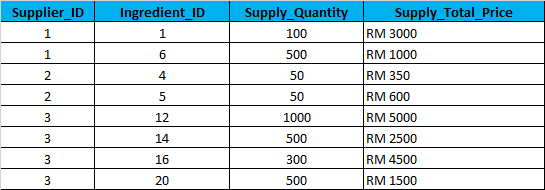
INGREDIENT(**Ingredient\_ID**, Ingredient\_Name, Ingredient\_Quantity\_Left)



**2NF Supply Table:**

Rename Ingredient\_Quantity to Supply\_Quantity

SUPPLY(**Supplier\_ID**, **Ingredient\_ID**, Supply\_Quantity, Supply\_Total\_Price)



## **9.5 THIRD NORMAL FORM ( 3NF )**

To convert 2NF to 3NF, we need to eliminate transitive dependencies and reassign corresponding dependent attributes in new tables.

**Transitive Dependency:**

Employee\_Job\_Name -> Employee\_Pay\_Rate

**Derived Attribute:**

Employee\_Salary = Employee\_Pay\_Rate x Employee\_Working\_Hours

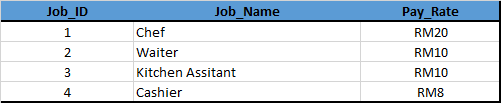
**3NF Employee Table:**

EMPLOYEE(**Employee\_ID**, Employee\_Name, Employee\_IC, Employee\_Contact, Employee\_Gender, Employee\_Email, Employee\_Address, Employee\_Hire\_Date, Job\_ID, Employee\_Working\_Hours)



**3NF Job table:**

JOB(**Job\_ID**, Job\_Name, Job\_Pay\_Rate)



# **10.0 Implementable Entity Relationship Diagram**

# **11.0 Entities and Attributes After Normalization**

| **Entities** | **Attributes** |
| --- | --- |
| Customer | **Customer\_ID**  Customer\_Name  Customer\_Address  Customer\_Contact  Customer\_Email |
| Order | **Order\_ID**  Customer\_ID  Employee\_ID  Order\_Date\_Time  Order\_Subtotal  Table\_No  Delivery\_ID |
| Menu | **Menu\_ID**  Menu\_Image  Menu\_Name  Menu\_Status  Menu\_Type  Menu\_Price |
| Supplier | **Supplier\_ID**  Supplier\_Name  Supplier\_Contact  Supplier\_Email  Supplier\_Address |
| Payment | **Payment\_Code**  Order\_ID  Voucher\_Code  Payment\_Date\_Time  Employee\_ID  Payment\_Method\_ID  Payment\_Amount |
| Voucher | **Voucher\_Code**  Voucher\_Name  Voucher\_Discount  Voucher\_Expiry\_Date  Voucher\_Valid |
| Employee | **Employee\_ID**  Employee\_Name  Employee\_IC  Employee\_Contact  Employee\_Gender  Employee\_Email  Employee\_Address  Employee\_Hire\_Date  Job\_ID  Employee\_Work\_Hours |
| Delivery | **Delivery\_ID**  Employee\_ID |
| Payment\_Method | **Payment\_Method\_ID**  Payment\_Method\_Name |
| Job | **Job\_ID**  Job\_Name  Job\_Pay\_Rate |
| Order\_Menu | **Order\_ID**  **Menu\_ID**  Order\_Quantity |
| Menu\_Ingredient | **Menu\_ID**  **Ingredient\_ID** |
| Ingredient | **Ingredient\_ID**  Ingredient\_Name  Ingredient\_Quantity\_Left |
| Supply | **Supplier\_ID**  **Ingredient\_ID**  Supply\_Quantity  Supply\_Total\_Price |

# **12.0 Challenges Faced During Data Collection Process and How To Overcome**

1. The data of business processes and business rules collected are not completed. We solved this problem by conducting several meetings with the manager of Ke Nina Cafe to get more information needed.
2. The information we collected by interviewing the Manager of Ke Nina Cafe is scattered and not in proper form. We need to rephrase the sentences and analyze each of them accordingly through group discussion to extract business rules.
3. It is difficult for us to conduct interviews with the manager of Ke Nina Cafe because she is busy. We do not want to bother her too much by suggesting different time slots during non-peak hours so that her working performance is not affected.
4. It is hard for all of us to have a physical meeting with the manager of Ke Nina Cafe due to distance. We use google meet to conduct the online interview session.
5. Hard to communicate with the manager of the restaurant using email. To make sure our communication more effectively, we go to the restaurant directly to discuss with the manager before interviewing

# **13.0 Conclusion**

A proper database for Ke Nina Cafe to record every worker's working hours and the food ingredients is designed to solve the problem of the database they have now. The total salary of each worker and the number of ingredients left can be checked easily using the database that we design.

# **14.0 Proof of The Interview/Data Collection**

## **14. 1 Cop:**

****

## **14.2 Online Interview With Manager On Google Meet:**

<https://drive.google.com/file/d/1iO0J7SyT6PZVqfSfPGIfXyoK3kKWDma0/view?usp=sharing>

## **14.3 Photos with Employer and Manager:**

****

## **14.4 Sample Receipt:**

****

## **14.5 Name Card:**

****

# **Report 2**

# **1.0 Database Design**

## **1.1 Introduction**

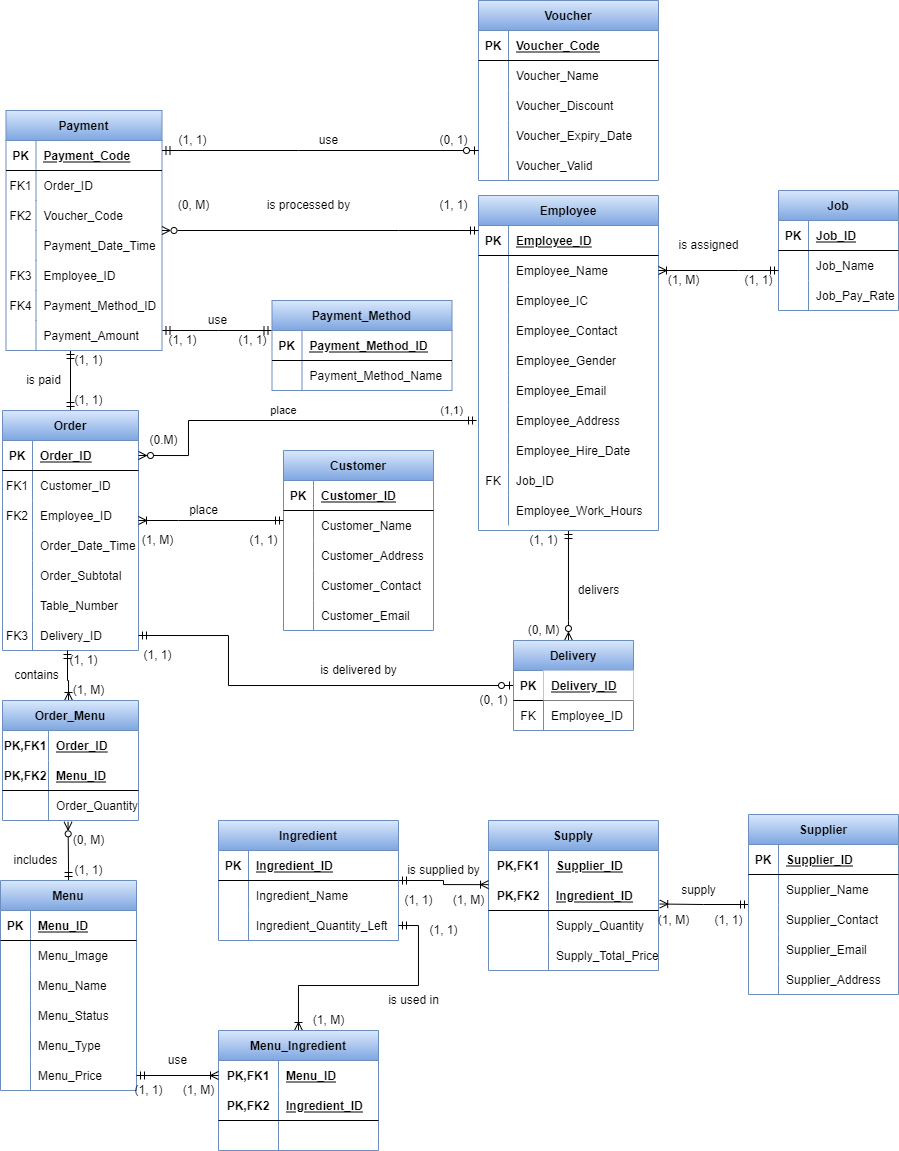
Database is a shared, integrated computer structure that stores a collection of end-user data and metadata. End-user data is the raw facts of interest to the end-user and metadata is the data about data, which the end-user data are integrated and managed. Metadata is used to describe data characteristics and relationships. In addition, databases are used by an organization as a method of storing, managing and retrieving information.

From the database management for the restaurant that we have created, there are 14 entities in total which are Vouchers, Payments, Payment\_Methods, Employees, Jobs, Customers, Orders, Delivery, Order Menu, Menu, Menu\_Ingredients, Ingredients, Supply and Suppliers. Each entity in the database has its own primary key whose values uniquely identify entity instances or each row in its table. The primary key attribute cannot be a null value and cannot have repeating values. Besides that, there are some of the entities in the database that have foreign keys. A foreign key is an attribute or combination of attributes in one table whose values must either match the primary key in another table or be null. The foreign keys can be applied to minimize data redundancy and establish relationships between entities. This is known as a relational database.

The main purpose of the database that we have created is for storing, managing and retrieving information about the restaurant. This database can help the employer and employees of the restaurant to do their work more efficiently. The employer can use this database to store the information of their orders, customers, employees, suppliers, the quantity of the ingredients left and more. If the employer uses spreadsheets like Excel to keep track of the food ingredients left, where the worker will need to count the ingredients every day and update them in the spreadsheet, it may lead to data anomalies. Next, the employees can also use this database to record the menu ID that is ordered by the customers and do not need to record on a piece of paper. During the peak hours, the handwriting of the workers becomes ugly and hard to read by the other workers. Besides that, there is the possibility that the workers miss some of the orders.

There are some future improvements to this restaurant database to make it better. If the employer expects to open more new branches at strategic locations to expand their business, we can improve this database so that this database can store the information of each branch of the restaurant. Besides, the managers are needed to help the employer to manage each branch. Each branch is managed by one manager so that we can improve this database to record the data of all the managers. Next, we can also improve this database to help the employer or managers to record the ingredients expiry date so that they will not need to remember or record using spreadsheets like Excel. In conclusion, we hope that we can do more improvement on this restaurant database and provide solutions for complicated and repetitive tasks to suit future needs. Hence, this database is able to meet the business standards and compete with other similar database management systems for restaurants.

## **1.2 Overview of ERD**



## 

## **1.3 Code**

### **1.3.1 Customers**

**Code:**

CREATE TABLE Customers(

Customer\_ID INT,

Customer\_Name VARCHAR(50) NOT NULL,

Customer\_Address VARCHAR(255) NOT NULL,

Customer\_Contact VARCHAR(20) NOT NULL,

Customer\_Email VARCHAR(255),

CONSTRAINT Customers\_PK

PRIMARY KEY(Customer\_ID),

CONSTRAINT Customer\_Email\_CHK

CHECK (Customer\_Email LIKE '%\_\_\_@\_\_\_%.\_\_%')

);

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, CUSTOMER\_ADDRESS, CUSTOMER\_CONTACT, CUSTOMER\_EMAIL)

VALUES (1001, 'Goh Chee Lam', 'NO. 12, PERSIARAN ANGGERIK 4, TAMAN ANGGERIK, 33000 KUALA KANGSAR, PERAK', '016-5429748', 'clgoh0726@gmail.com');

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, CUSTOMER\_ADDRESS, CUSTOMER\_CONTACT, CUSTOMER\_EMAIL)

VALUES (1002, 'Frankie Lim Qi Quan', 'NO. 22, PERSIARAN ANGGERIK 12, TAMAN ANGGERIK BIRU, 33000 KUALA KANGSAR, PERAK', '016-5934918', 'heidragon3045@gmail.com');

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, CUSTOMER\_ADDRESS, CUSTOMER\_CONTACT, CUSTOMER\_EMAIL)

VALUES (1003, 'Pua Jing Yi', 'NO.31, PERSIARAN SIPUT 10, TAMAN SIPUT, 33000 KUALA KANGSAR, PERAK', '011-10885068', 'u2005396@siswa.um.edu.my');

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, CUSTOMER\_ADDRESS, CUSTOMER\_CONTACT, CUSTOMER\_EMAIL)

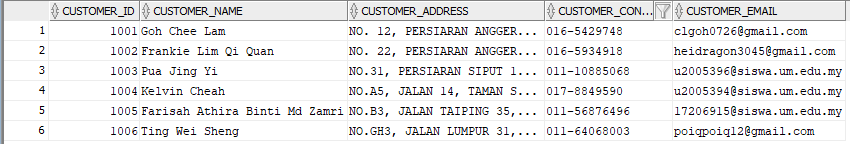
VALUES (1004, 'Kelvin Cheah', 'NO.A5, JALAN 14, TAMAN SUNGAI, 33000 KUALA KANGSAR, PERAK', '017-8849590', 'u2005394@siswa.um.edu.my');

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, CUSTOMER\_ADDRESS, CUSTOMER\_CONTACT, CUSTOMER\_EMAIL)

VALUES (1005, 'Farisah Athira Binti Md Zamri', 'NO.B3, JALAN TAIPING 35, TAMAN TAIPING, 33000 KUALA KANGSAR, PERAK', '011-56876496', '17206915@siswa.um.edu.my');

INSERT INTO CUSTOMERS (CUSTOMER\_ID, CUSTOMER\_NAME, CUSTOMER\_ADDRESS, CUSTOMER\_CONTACT, CUSTOMER\_EMAIL)

VALUES (1006, 'Ting Wei Sheng', 'NO.GH3, JALAN LUMPUR 31, TAMAN LUMPUR, 43200 DAMANSARA, KUALA LUMPUR', '011-64068003', 'poiqpoiq12@gmail.com');



### **1.3.2 Jobs**

**Code:**

CREATE TABLE Jobs(

Job\_ID INT,

Job\_Name VARCHAR(50) NOT NULL,

Job\_Pay\_Rate NUMBER(4, 2) NOT NULL,

CONSTRAINT Jobs\_PK

PRIMARY KEY(Job\_ID)

);

INSERT INTO JOBS (JOB\_ID, JOB\_NAME, JOB\_PAY\_RATE)

VALUES (1, 'Chef', 20);

INSERT INTO JOBS (JOB\_ID, JOB\_NAME, JOB\_PAY\_RATE)

VALUES (2, 'Waiter', 10);

INSERT INTO JOBS (JOB\_ID, JOB\_NAME, JOB\_PAY\_RATE)

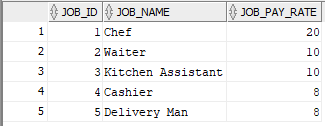
VALUES (3, 'Kitchen Assistant', 10);

INSERT INTO JOBS (JOB\_ID, JOB\_NAME, JOB\_PAY\_RATE)

VALUES (4, 'Cashier', 8);

INSERT INTO JOBS (JOB\_ID, JOB\_NAME, JOB\_PAY\_RATE)

VALUES (5, 'Delivery Man', 8);



### **1.3.3 Payment\_Methods**

**Code:**

CREATE TABLE Payment\_Methods(

Payment\_Method\_ID INT,

Payment\_Method\_Name VARCHAR(100) NOT NULL,

CONSTRAINT Payment\_Methods\_PK

PRIMARY KEY(Payment\_Method\_ID)

);

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (1, 'Shopee Pay');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (2, 'Touch And Go e-wallet');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (3, 'QR Pay');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (4, 'Cash');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (5, 'Maybank2u');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (6, 'Cash On Delivery');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

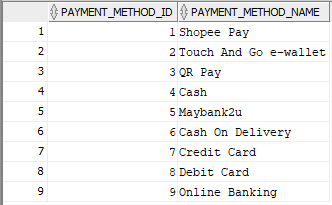
VALUES (7, 'Credit Card');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (8, 'Debit Card');

INSERT INTO PAYMENT\_METHODS (PAYMENT\_METHOD\_ID, PAYMENT\_METHOD\_NAME)

VALUES (9, 'Online Banking');



### **1.3.4 Supplier**

**Code:**

CREATE TABLE Suppliers(

Supplier\_ID INT,

Supplier\_Name VARCHAR(100) NOT NULL,

Supplier\_Contact VARCHAR(20) NOT NULL,

Supplier\_Email VARCHAR(100) NOT NULL,

Supplier\_Address VARCHAR(255) NOT NULL,

CONSTRAINT Suppliers\_PK

PRIMARY KEY(Supplier\_ID),

CONSTRAINT Supplier\_Email\_CHK

CHECK (Supplier\_Email LIKE '%\_\_\_@\_\_\_%.\_\_%')

);

INSERT INTO SUPPLIERS (SUPPLIER\_ID, SUPPLIER\_NAME, SUPPLIER\_CONTACT, SUPPLIER\_EMAIL, SUPPLIER\_ADDRESS)

VALUES (1, 'Ali Farm', '012-3456789', 'ali@gmail.com', 'No.10, Jalan Industri, 43200 Kuala Lumpur');

INSERT INTO SUPPLIERS (SUPPLIER\_ID, SUPPLIER\_NAME, SUPPLIER\_CONTACT, SUPPLIER\_EMAIL, SUPPLIER\_ADDRESS)

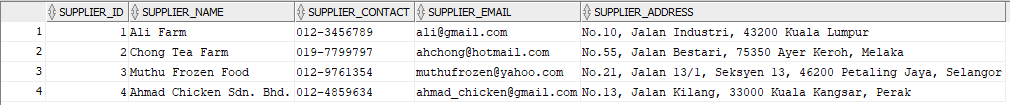
VALUES (2, 'Chong Tea Farm', '019-7799797', 'ahchong@hotmail.com', 'No.55, Jalan Bestari, 75350 Ayer Keroh, Melaka');

INSERT INTO SUPPLIERS (SUPPLIER\_ID, SUPPLIER\_NAME, SUPPLIER\_CONTACT, SUPPLIER\_EMAIL, SUPPLIER\_ADDRESS)

VALUES (3, 'Muthu Frozen Food', '012-9761354', 'muthufrozen@yahoo.com', 'No.21, Jalan 13/1, Seksyen 13, 46200 Petaling Jaya, Selangor');

INSERT INTO SUPPLIERS (SUPPLIER\_ID, SUPPLIER\_NAME, SUPPLIER\_CONTACT, SUPPLIER\_EMAIL, SUPPLIER\_ADDRESS)

VALUES (4, 'Ahmad Chicken Sdn. Bhd.', '012-4859634', 'ahmad\_chicken@gmail.com', 'No.13, Jalan Kilang, 33000 Kuala Kangsar, Perak');



### 

### **1.3.5 Ingredients**

**Code:**

CREATE TABLE Ingredients(

Ingredient\_ID INT,

Ingredient\_Name VARCHAR(100) NOT NULL,

Ingredient\_Quantity\_Left INT NOT NULL,

CONSTRAINT Ingredients\_PK

PRIMARY KEY(Ingredient\_ID)

);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (1, 'Fresh Milk ', 10);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (2, 'Brown Sugar Boba', 75);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (3, 'Grass Jelly', 67);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (4, 'Black Tea Leaves', 12);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (5, 'Green Tea Leaves', 22);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (6, 'Condensed Milk', 30);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (7, 'Rose Syrup', 45);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (8, 'Rose Pedal', 66);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (9, 'Pink Boba', 55);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (10, 'Natural Honey ', 43);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (11, 'Jasmine Tea Powder', 89);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (12, 'Fries', 11);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (13, 'Himalayan salt', 30);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (14, 'Taiwanese Hotdog', 10);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (15, 'Cucumber', 23);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (16, 'Chicken thigh', 23);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (17, 'Seasoning Powder', 15);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

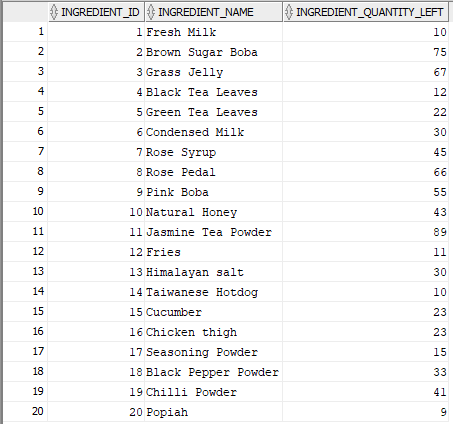
VALUES (18, 'Black Pepper Powder', 33);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (19, 'Chilli Powder', 41);

INSERT INTO INGREDIENTS (INGREDIENT\_ID, INGREDIENT\_NAME, INGREDIENT\_QUANTITY\_LEFT)

VALUES (20, 'Popiah', 9);



### **1.3.6 Menu**

**Code:**

CREATE TABLE Menu(

Menu\_ID INT,

Menu\_Image BLOB,

Menu\_Name VARCHAR(255) NOT NULL,

Menu\_Status VARCHAR(2) NOT NULL,

Menu\_Type VARCHAR(20) NOT NULL,

Menu\_Price NUMBER(4, 2) NOT NULL,

CONSTRAINT Menu\_PK

PRIMARY KEY(Menu\_ID),

CONSTRAINT Menu\_Status\_CHK

CHECK (Menu\_Status IN ('A', 'NA')),

CONSTRAINT Menu\_Price\_CHK

CHECK (Menu\_Price > 0)

);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (1, '', 'Brown Sugar Boba Milk (H)', 'A', 'Drink', 10.30);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (2, '', 'Brown Sugar Boba Milk (C)', 'NA', 'Drink', 13.30);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (3, '', 'Brown Sugar Boba Milk with Grass Jelly (H)', 'A', 'Drink', 10.30);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (4, '', 'Brown Sugar Boba Milk with Grass Jelly (C)', 'A', 'Drink', 13.30);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (5, '', 'Brown Sugar Boba Milk Tea (H)', 'A', 'Drink', 10.30);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (6, '', 'Brown Sugar Boba Milk Tea (C)', 'NA', 'Drink', 13.30);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (7, '', 'Damascus Rose Tea (H)', 'A', 'Drink', 8.00);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (8, '', 'Jasmine Tea with Honey (H)', 'A', 'Drink', 8.00);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (9, '', 'Fries', 'A', 'Food', 5.00);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

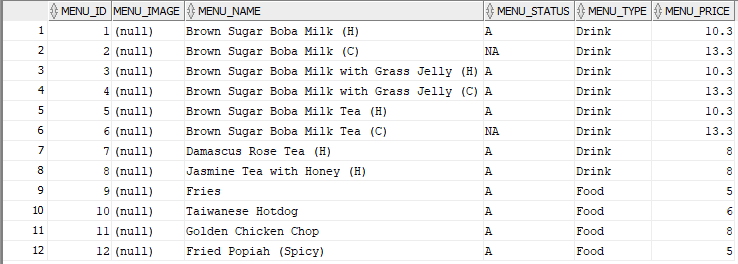
VALUES (10, '', 'Taiwanese Hotdog', 'A', 'Food', 6.00);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (11, '', 'Golden Chicken Chop ', 'A', 'Food', 8.00);

INSERT INTO MENU (MENU\_ID, MENU\_IMAGE, MENU\_NAME, MENU\_STATUS, MENU\_TYPE, MENU\_PRICE)

VALUES (12, '', 'Fried Popiah (Spicy)', 'A', 'Food', 5.00);



### **1.3.7 Vouchers**

**Code:**

CREATE TABLE Vouchers(

Voucher\_Code VARCHAR(255),

Voucher\_Name VARCHAR(100) NOT NULL,

Voucher\_Discount INT NOT NULL,

Voucher\_Expiry\_Date DATE NOT NULL,

Voucher\_Valid CHAR DEFAULT 'T',

CONSTRAINT Vouchers\_PK

PRIMARY KEY(Voucher\_Code),

CONSTRAINT Voucher\_Valid\_CHK

CHECK (Voucher\_Valid IN ('T', 'F'))

);

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00001', 'New Year Promotional Voucher', 20, to\_date('2022-01-10 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00002', 'New Year Promotional Voucher', 20, to\_date('2022-01-10 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00003', 'New Year Promotional Voucher', 20, to\_date('2022-01-10 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'F');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00004', 'New Year Promotional Voucher', 20, to\_date('2022-01-10 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'F');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00005', 'New Year Promotional Voucher', 20, to\_date('2022-01-10 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00006', 'Happy CNY 15% off', 15, to\_date('2022-02-16 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00007', 'Happy CNY 15% off', 15, to\_date('2022-02-16 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00008', 'Happy CNY 15% off', 15, to\_date('2022-02-16 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00009', 'Happy CNY 15% off', 15, to\_date('2022-02-16 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'F');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00010', 'Happy CNY 15% off', 15, to\_date('2022-02-16 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'F');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00011', 'Crazy Hours', 10, to\_date('2022-12-31 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00012', 'Crazy Hours', 10, to\_date('2022-12-31 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00013', 'Crazy Hours', 10, to\_date('2022-12-31 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00014', 'Crazy Hours', 10, to\_date('2022-12-31 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00015', 'Crazy Hours', 10, to\_date('2022-12-31 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00016', 'Buy 2 at 15% discount', 15, to\_date('2022-03-03 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00017', 'Buy 2 at 15% discount', 15, to\_date('2023-03-03 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

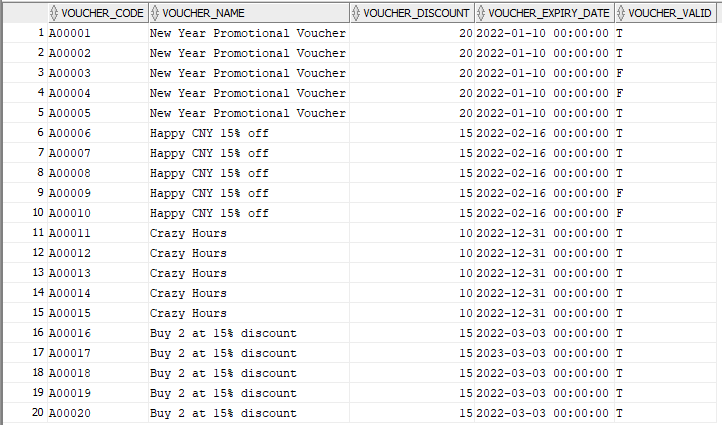
VALUES ('A00018', 'Buy 2 at 15% discount', 15, to\_date('2022-03-03 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00019', 'Buy 2 at 15% discount', 15, to\_date('2022-03-03 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');

INSERT INTO VOUCHERS (VOUCHER\_CODE, VOUCHER\_NAME, VOUCHER\_DISCOUNT, VOUCHER\_EXPIRY\_DATE, VOUCHER\_VALID)

VALUES ('A00020', 'Buy 2 at 15% discount', 15, to\_date('2022-03-03 00:00:00', 'YYYY-MM-DD:HH24:MI:SS'), 'T');



### **1.3.8 Menu\_Ingredients**

**Code:**

CREATE TABLE Menu\_Ingredients(

Menu\_ID INT,

Ingredient\_ID INT,

CONSTRAINT Menu\_Ingredients\_PK

PRIMARY KEY(Menu\_ID, Ingredient\_ID),

CONSTRAINT MenuIngredient\_Menu\_FK

FOREIGN KEY (Menu\_ID)

REFERENCES Menu(Menu\_ID),

CONSTRAINT Menu\_Ingredient\_Ingredient\_FK

FOREIGN KEY (Ingredient\_ID)

REFERENCES Ingredients(Ingredient\_ID)

);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (1, 1);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (1, 2);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (2, 1);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (2, 2);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (3, 1);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (3, 3);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (4, 1);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (4, 3);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (5, 6);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (5, 4);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (5, 2);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (6, 6);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (6, 4);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (6, 2);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (7, 7);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (7, 8);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (7, 9);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (8, 10);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (8, 11);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (8, 9);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (9, 12);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (9, 13);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (10, 14);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (10, 15);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (11, 16);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

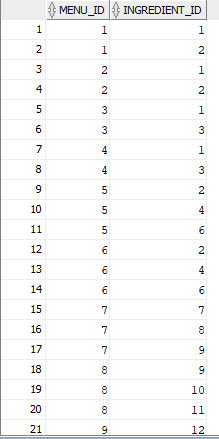
VALUES (11, 17);

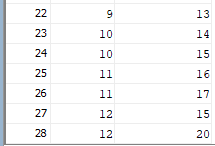
INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (12, 20);

INSERT INTO MENU\_INGREDIENTS (MENU\_ID, INGREDIENT\_ID)

VALUES (12, 15);





### **1.3.9 Employees**

**Code:**

CREATE TABLE Employees(

Employee\_ID INT,

Employee\_Name VARCHAR(50) NOT NULL,

Employee\_IC VARCHAR(20) NOT NULL,

Employee\_Contact VARCHAR(20) NOT NULL,

Employee\_Gender VARCHAR(1) NOT NULL,

Employee\_Email VARCHAR(30) NOT NULL,

Employee\_Address VARCHAR(100),

Employee\_Hire\_Date DATE NOT NULL,

Job\_ID INT NOT NULL,

Employee\_Work\_Hours NUMBER(4,2) NOT NULL,

CONSTRAINT Employees\_PK

PRIMARY KEY(Employee\_ID),

CONSTRAINT Employee\_Job\_FK

FOREIGN KEY (Job\_ID)

REFERENCES Jobs(Job\_ID),

CONSTRAINT Employee\_Gender\_CHK

CHECK (Employee\_Gender IN ('M', 'F')),

CONSTRAINT Employee\_Email\_CHK

CHECK (Employee\_Email LIKE '%\_\_\_@\_\_\_%.\_\_%')

);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (1, 'Chong Wee Wah', '991231-03-1456', '011-1153445', 'F', 'weewah@gmail.com', ' 19 Uoa Centre Office Block Jln Pinang, 50450 Kuala Lumpur, Wilayah Persekutuan.', to\_date('2015-06-07', 'RRRR-MM-DD'), 1, 10);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (2, 'Siti Aisyah Binti Abudul Razali', '930228-01-9976', '019-1345325', 'F', 'siti1993@live.com', '88, Taman Perindustrian Puchong Utama, Seksyen 2, 47100 Puchong, Selangor.', to\_date('2017-08-07', 'RRRR-MM-DD'), 2, 7);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (3, 'Mariah Huda Binti Admad Danish', '010908-06-2278', '017-2145004', 'F', 'mariahhuda@yahoo.com', '78 Treacher ST, 30000 Ipoh, Perak.', to\_date('2018-05-05', 'RRRR-MM-DD'), 3, 9);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (4, 'Tan Wee Ren', '000501-02-1187', '016-7433506', 'M', 'tan0501@yahoo.com', ' 8 Jln Kemajuan Desa Rahmat, 81200 Johor Bahru, Johor.', to\_date('2018-12-13', 'RRRR-MM-DD'), 4, 5);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (5, 'Palani Subramaniam A/L Navin ', '960605-01-3241', '012-4394752', 'M', 'palasu@gmail.com', 'Lot 10, Bandar Baru Darulaman, 06000 Jitra, Kedah.', to\_date('2019-09-02', 'RRRR-MM-DD'), 2, 8);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (6, 'Lim Zhi Ming', '020815-09-1977', '018-39408578', 'M', 'alenlim@gmail.com', '45, Section 16/11, Off Jalan Damansara, 46350 Petaling Jaya, Selangor.', to\_date('2020-04-03', 'RRRR-MM-DD'), 3, 9);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (7, 'Tan Qiu Yu', '990101-01-3334', '010-90391334', 'F', 'yuyu11@live.com', 'G 147 Jln Tun H S Lee, 50000 Kuala Lumpur, Wilayah Persekutuan.', to\_date('2020-11-19', 'RRRR-MM-DD'), 1, 11);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (8, 'Ali Haikal Bin Abu Bakar', '881009-10-2687', '011-3990523', 'M', 'haikalali@yahoo.com', '312 Kampung Baru Semenyih Semenyih, 43500 Petaling Jaya, Selangor.', to\_date('2020-09-12', 'RRRR-MM-DD'), 1, 9);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

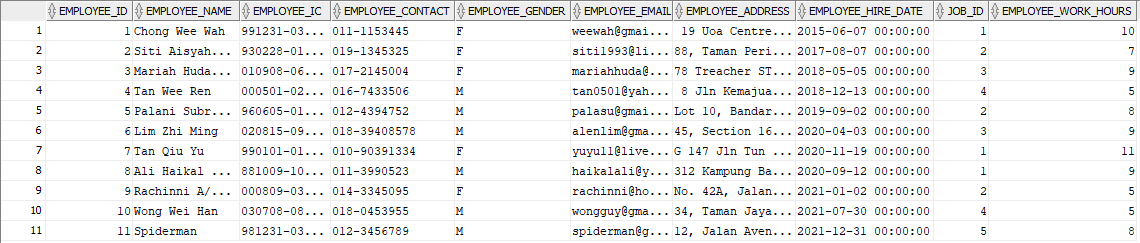
VALUES (9, 'Rachinni A/P Saravana ', '000809-03-4566', '014-3345095', 'F', 'rachinni@hotmail.com', 'No. 42A, Jalan Market, 30000 Ipoh, Perak.', to\_date('2021-01-02', 'RRRR-MM-DD'), 2, 5);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (10, 'Wong Wei Han', '030708-08-0377', '018-0453955', 'M', 'wongguy@gmail.com', '34, Taman Jaya, Ara Damansara, 47301 Petaling Jaya, Selangor.', to\_date('2021-07-30', 'RRRR-MM-DD'), 4, 5);

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (11, 'Spiderman', '981231-03-0001', '012-3456789', 'M', 'spiderman@gmail.com', '12, Jalan Avengers, Taman Marvel, 33000 Kuala Kangsar, Perak', to\_date('2021-12-31', 'RRRR-MM-DD'), 5, 8);



### **1.3.10 Delivery**

**Code:**

CREATE TABLE Delivery(

Delivery\_ID INT,

Employee\_ID INT NOT NULL,

CONSTRAINT Delivery\_PK

PRIMARY KEY(Delivery\_ID),

CONSTRAINT Delivery\_Employee\_FK

FOREIGN KEY(Employee\_ID)

REFERENCES Employees(Employee\_ID)

);

INSERT INTO DELIVERY (DELIVERY\_ID, EMPLOYEE\_ID)

VALUES (1, 11);

INSERT INTO DELIVERY (DELIVERY\_ID, EMPLOYEE\_ID)

VALUES (2, 11);



### **1.3.11 Orders**

**Code:**

CREATE TABLE Orders(

Order\_ID INT,

Order\_Date\_Time DATE NOT NULL,

Customer\_ID INT NOT NULL,

Table\_Number INT,

Delivery\_ID INT,

Employee\_ID INT NOT NULL,

Order\_Subtotal NUMBER(8, 2) NOT NULL,

CONSTRAINT Orders\_PK

PRIMARY KEY (Order\_ID),

CONSTRAINT Order\_Customer\_FK

FOREIGN KEY (Customer\_ID)

REFERENCES Customers(Customer\_ID),

CONSTRAINT Order\_Employee\_FK

FOREIGN KEY (Employee\_ID)

REFERENCES Employees(Employee\_ID),

CONSTRAINT Order\_Delivery\_FK

FOREIGN KEY (Delivery\_ID)

REFERENCES Delivery(Delivery\_ID)

);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (1, to\_date('2021-11-23 13:25:01', 'YYYY-MM-DD:HH24:MI:SS'), 1001, 1, '', 4, 24.30);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (2, to\_date('2021-11-23 13:28:01', 'YYYY-MM-DD:HH24:MI:SS'), 1002, 2, '', 4, 20.30);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (3, to\_date('2021-11-24 15:38:40', 'YYYY-MM-DD:HH24:MI:SS'), 1003, 3, '', 10, 46.60);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (4, to\_date('2021-11-24 16:01:03', 'YYYY-MM-DD:HH24:MI:SS'), 1004, '', 1, 4, 42.60);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (5, to\_date('2021-11-24 17:45:35', 'YYYY-MM-DD:HH24:MI:SS'), 1005, '', 2, 10, 21.30);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (6, to\_date('2021-11-24 17:55:24', 'YYYY-MM-DD:HH24:MI:SS'), 1002, 2, '', 10, 25.6);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (7, to\_date('2021-11-25 10:34:35', 'YYYY-MM-DD:HH24:MI:SS'), 1006, 1, '', 10, 44.20);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (8, to\_date('2021-11-25 11:03:23', 'YYYY-MM-DD:HH24:MI:SS'), 1003, 2, '', 4, 15.30);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (9, to\_date('2021-11-25 12:27:56', 'YYYY-MM-DD:HH24:MI:SS'), 1005, 1, '', 4, 37.60);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

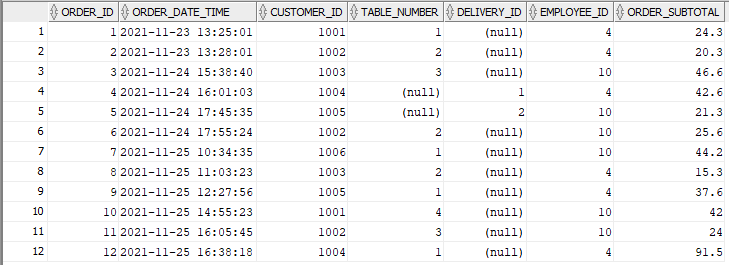
VALUES (10, to\_date('2021-11-25 14:55:23', 'YYYY-MM-DD:HH24:MI:SS'), 1001, 4, '', 10, 42.00);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (11, to\_date('2021-11-25 16:05:45', 'YYYY-MM-DD:HH24:MI:SS'), 1002, 3, '', 10, 24.00);

INSERT INTO ORDERS (ORDER\_ID, ORDER\_DATE\_TIME, CUSTOMER\_ID, TABLE\_NUMBER, DELIVERY\_ID, EMPLOYEE\_ID, ORDER\_SUBTOTAL)

VALUES (12, to\_date('2021-11-25 16:38:18', 'YYYY-MM-DD:HH24:MI:SS'), 1004, 1, '', 4, 91.50);



### **1.3.12 Order\_Menu**

**Code:**

CREATE TABLE Order\_Menu(

Order\_ID INT NOT NULL,

Menu\_ID INT NOT NULL,

Order\_Quantity INT NOT NULL,

CONSTRAINT Order\_Menu\_PK

PRIMARY KEY(Order\_ID, Menu\_ID),

CONSTRAINT OrderMenu\_Order\_FK

FOREIGN KEY(Order\_ID)

REFERENCES Orders(Order\_ID),

CONSTRAINT OrderMenu\_Menu\_FK

FOREIGN KEY(Menu\_ID)

REFERENCES Menu(Menu\_ID),

CONSTRAINT Order\_Quantity\_CHK

CHECK (Order\_Quantity > 0)

);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (1, 1, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (1, 10, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (1, 11, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (2, 5, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (2, 12, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (3, 3, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (3, 6, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (3, 11, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (3, 12, 3);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (4, 8, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (4, 6, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (5, 2, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (6, 3, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (6, 9, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (5, 11, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (7, 1, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (7, 3, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (7, 4, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (8, 5, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (8, 9, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (9, 10, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (9, 12, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (9, 2, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (10, 7, 4);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (10, 9, 2);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (11, 12, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (11, 11, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (11, 10, 1);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

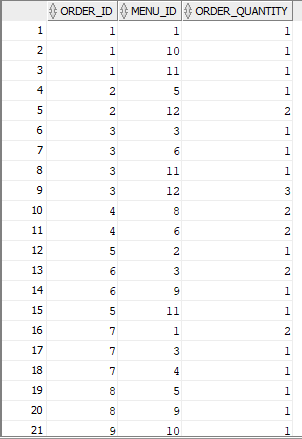
VALUES (11, 9, 1);

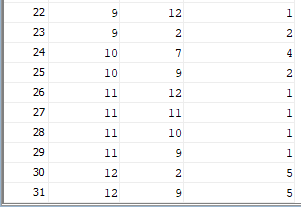
INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (12, 2, 5);

INSERT INTO ORDER\_MENU (ORDER\_ID, MENU\_ID, ORDER\_QUANTITY)

VALUES (12, 9, 5);





### **1.3.13 Supply**

**Code:**

CREATE TABLE Supply(

Supplier\_ID INT,

Ingredient\_ID INT,

Supply\_Quantity INT NOT NULL,

Supply\_Total\_Price NUMBER(8, 2) NOT NULL,

CONSTRAINT Supply\_PK

PRIMARY KEY (Supplier\_ID, Ingredient\_ID),

CONSTRAINT Supply\_Supplier\_FK

FOREIGN KEY (Supplier\_ID)

REFERENCES Suppliers(Supplier\_ID),

CONSTRAINT Supply\_Ingredient\_FK

FOREIGN KEY (Ingredient\_ID)

REFERENCES Ingredients(Ingredient\_ID),

CONSTRAINT Supply\_Quantity\_CHK

CHECK (Supply\_Quantity > 0),

CONSTRAINT Supply\_Total\_Price\_CHK

CHECK (Supply\_Total\_Price > 0)

);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (1, 1, 100, 3000);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (1, 6, 500, 1000);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (1, 2, 200, 600);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (1, 9, 200, 600);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (1, 13, 100, 100);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (2, 4, 50, 350);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (2, 5, 50, 600);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (2, 7, 150, 450);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (2, 8, 150, 450);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (2, 15, 50, 150);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (3, 12, 1000, 5000);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (3, 14, 500, 2500);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (3, 16, 300, 4500);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (3, 20, 500, 1500);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (3, 3, 50, 200);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (4, 10, 10, 1000);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (4, 11, 100, 200);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

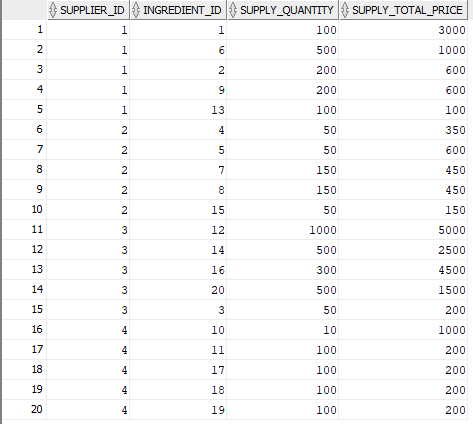
VALUES (4, 17, 100, 200);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (4, 18, 100, 200);

INSERT INTO SUPPLY (SUPPLIER\_ID, INGREDIENT\_ID, SUPPLY\_QUANTITY, SUPPLY\_TOTAL\_PRICE)

VALUES (4, 19, 100, 200);



### **1.3.14 Payments**

**Code:**

CREATE TABLE Payments(

Payment\_Code INT,

Order\_ID INT NOT NULL,

Voucher\_Code VARCHAR(255),

Payment\_Date\_Time DATE NOT NULL,

Employee\_ID INT NOT NULL,

Payment\_Method\_ID INT NOT NULL,

Payment\_Amount NUMBER(8, 2) NOT NULL,

CONSTRAINT Payments\_PK

PRIMARY KEY(Payment\_Code),

CONSTRAINT Payment\_Order\_FK

FOREIGN KEY(Order\_ID)

REFERENCES Orders(Order\_ID),

CONSTRAINT Payment\_Voucher\_FK

FOREIGN KEY(Voucher\_Code)

REFERENCES Vouchers(Voucher\_Code),

CONSTRAINT Payment\_Employee\_FK

FOREIGN KEY(Employee\_ID)

REFERENCES Employees(Employee\_ID),

CONSTRAINT Payment\_Method\_FK

FOREIGN KEY(Payment\_Method\_ID)

REFERENCES Payment\_Methods(Payment\_Method\_ID)

);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (1, 1, 'A00003', to\_date('2021-11-23 14:10:20', 'YYYY-MM-DD:HH24:MI:SS'), 4, 4, 19.45);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (2, 2, 'A00004', to\_date('2021-11-23 15:00:03', 'YYYY-MM-DD:HH24:MI:SS'), 4, 4, 16.25);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (3, 3, 'A00009', to\_date('2021-11-24 17:18:30', 'YYYY-MM-DD:HH24:MI:SS'), 4, 1, 39.60);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (4, 4, 'A00010', to\_date('2021-11-24 16:30:03', 'YYYY-MM-DD:HH24:MI:SS'), 4, 5, 36.21);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (5, 5, '', to\_date('2021-11-24 18:30:31', 'YYYY-MM-DD:HH24:MI:SS'), 4, 6, 21.30);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (6, 6, '', to\_date('2021-11-24 19:02:24', 'YYYY-MM-DD:HH24:MI:SS'), 10, 4, 25.60);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (7, 7, '', to\_date('2021-11-25 11:23:56', 'YYYY-MM-DD:HH24:MI:SS'), 10, 2, 44.20);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (8, 8, '', to\_date('2021-11-25 12:15:05', 'YYYY-MM-DD:HH24:MI:SS'), 4, 2, 15.30);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (9, 9, '', to\_date('2021-11-25 14:05:23', 'YYYY-MM-DD:HH24:MI:SS'), 4, 2, 37.60);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

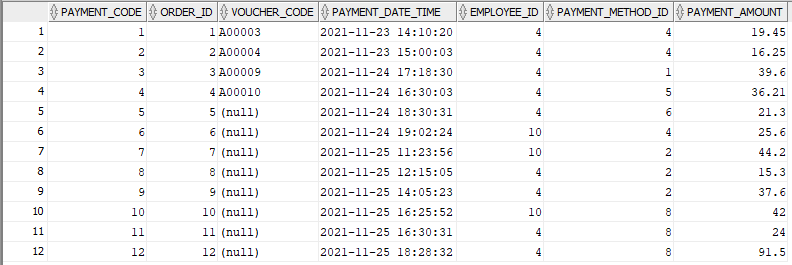
VALUES (10, 10, '', to\_date('2021-11-25 16:25:52', 'YYYY-MM-DD:HH24:MI:SS'), 10, 8, 42.00);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (11, 11, '', to\_date('2021-11-25 16:30:31', 'YYYY-MM-DD:HH24:MI:SS'), 4, 8, 24.00);

INSERT INTO PAYMENTS (PAYMENT\_CODE, ORDER\_ID, VOUCHER\_CODE, PAYMENT\_DATE\_TIME, EMPLOYEE\_ID, PAYMENT\_METHOD\_ID, PAYMENT\_AMOUNT)

VALUES (12, 12, '', to\_date('2021-11-25 18:28:32', 'YYYY-MM-DD:HH24:MI:SS'), 4, 8, 91.50);



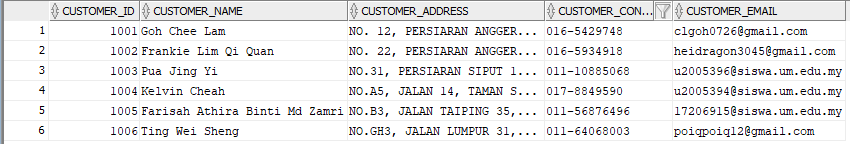
# **2.0 SQL Statement And Description**

## **2.1 ALTER**

**Description:** The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

**Situation 1:** The employee wants to alter the data(column) and information in the table customers. They want to add a new column named Birthday in the Customers table.

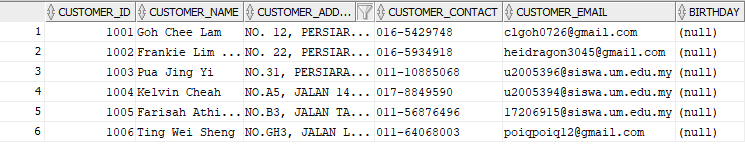
**CUSTOMERS TABLE**

****

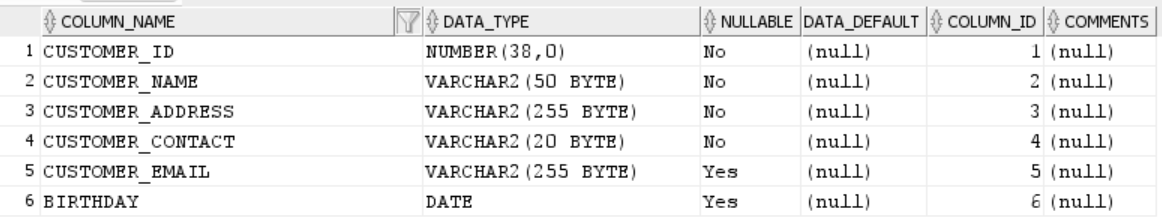
**SQL Statement**

**ALTER TABLE** Customers

**ADD** Birthday date;



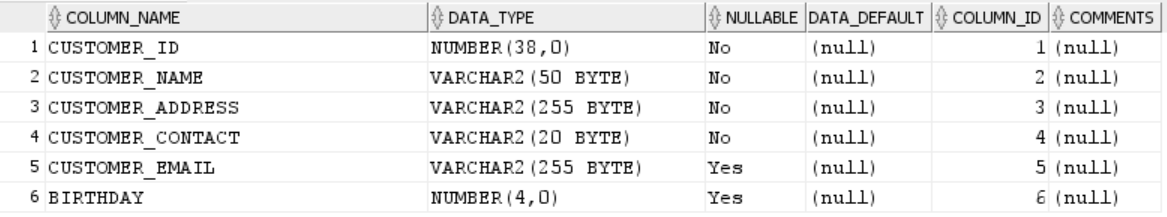
**Situation 2:** If the employee wants to alter the data(column) and information in the table customers. They can modify the data type if they thought the data type needed some changes.

****

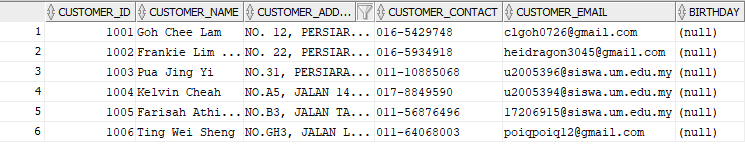
**SQL Statement**

**ALTER TABLE** Customers

**MODIFY** Birthday NUMBER(4, 0);

****

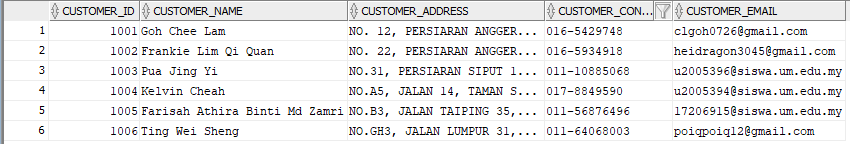
**Situation 3:** If the employee wants to alter the data(column) and information in the table customers. They want to delete the column name Birthday in the Customers table.



**SQL Statement**

**ALTER TABLE** Customers

**DROP COLUMN** Birthday;



## 

## **2.2 LIKE, WHERE**

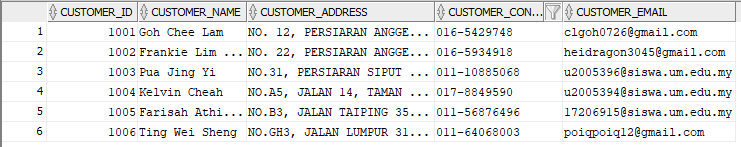
**Description:**

**LIKE:** The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

**WHERE:** The WHERE clause is used to filter records. It is used to extract only those records that fulfil a specified condition.

**Situation:** The employer wants to search all customers with CUSTOMER\_NAME starting with "Pua".

**CUSTOMERS TABLE**



**SQL Statement**

**SELECT \***

**FROM** CUSTOMERS

**WHERE** Customer\_Name

**LIKE** '%Pua%';



## 

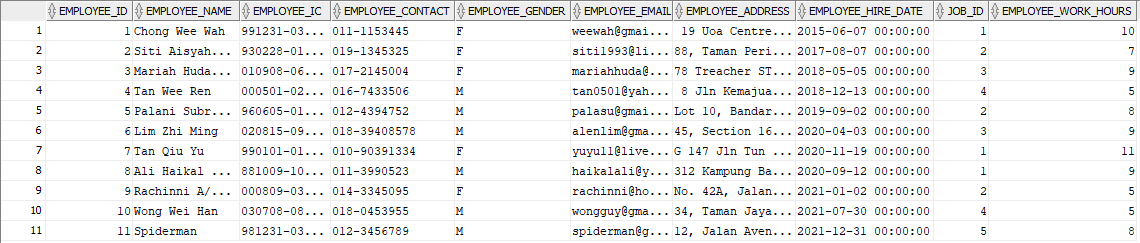
## **2.3 IN**

**Description:**

**IN:** The IN operator is used to specify multiple values in a WHERE clause.

**Situation:** The employer wants to select employees who work 9, 10 and 11 hours from the EMPLOYEES table.

**EMPLOYEES TABLE**



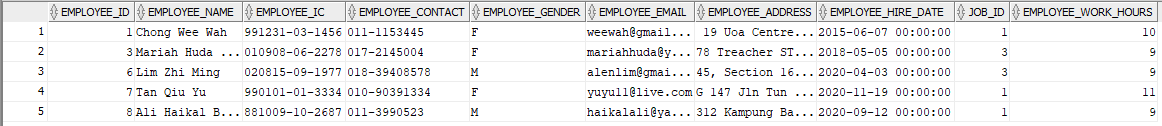
**SQL Statement**

**SELECT \***

**FROM** EMPLOYEES

**WHERE** Employee\_Wrk\_Hours

**IN** (9, 10, 11);



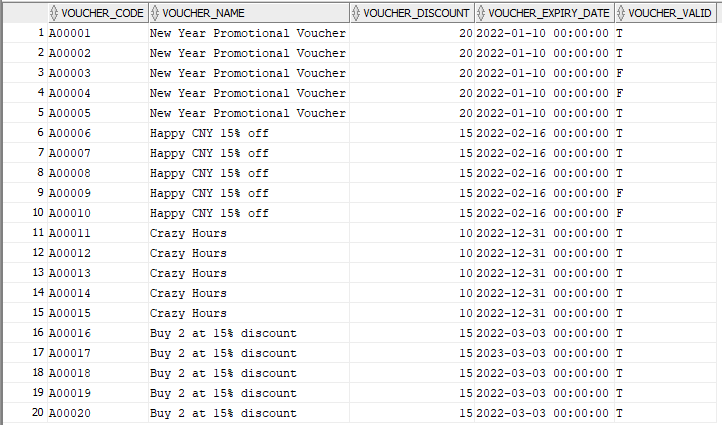
## **2.4 DELETE**

**Description:**

**DELETE:** The DELETE statement is used to delete existing records in a table.

**Situation:** The employer wants to delete existing records in the VOUCHERS table because the employer does not want to do ‘Buy 2 at 15% discount’ promotions.

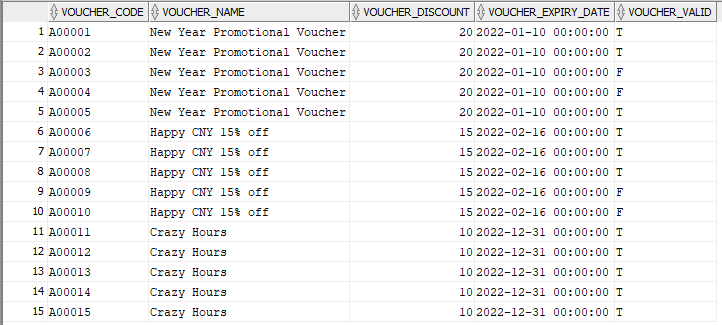
**VOUCHERS TABLE**



**SQL Statement**

**DELETE** FROM VOUCHERS

**WHERE** Voucher\_Name = 'Buy 2 at 15% discount';



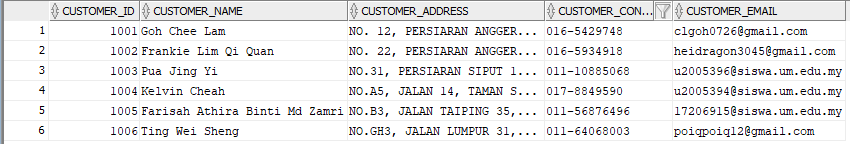
## **2.5 UPDATE**

**Description:**

**UPDATE:** The UPDATE statement is used to modify the existing records in a table.

**Situation:** The employer wants to modify the existing records in a table. If the customer name is wrongly written, the employer wants to modify the record in the CUSTOMERS table.

**CUSTOMERS TABLE**

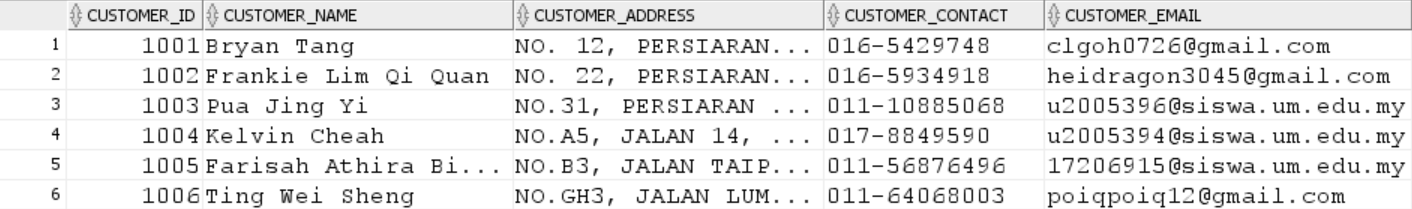


**SQL Statement**

**UPDATE** CUSTOMERS

**SET** Customer\_Name = 'Bryan Tang'

**WHERE** Customer\_ID = 1001;

****

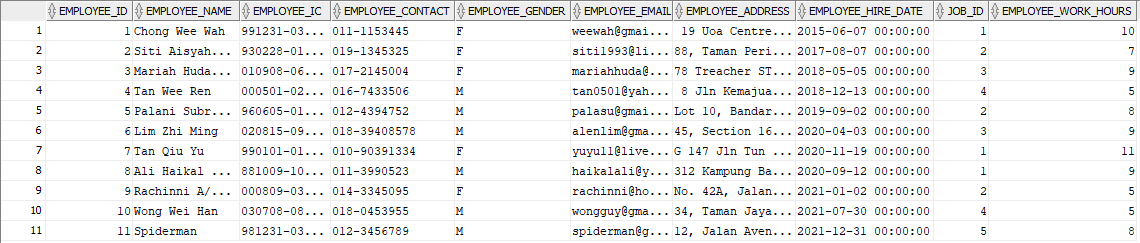
## **2.6 INSERT INTO**

**Description:**

**INSERT INTO:** insert new data records into a table.

**Situation:** The employer hires a new employee to work at the restaurant and wants to insert new records containing employee information into the EMPLOYEES table.

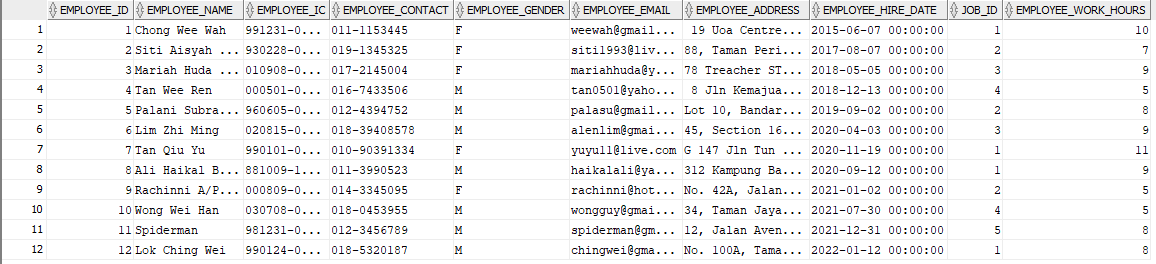
**EMPLOYEES TABLE**



**SQL Statement**

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, EMPLOYEE\_NAME, EMPLOYEE\_IC, EMPLOYEE\_CONTACT, EMPLOYEE\_GENDER, EMPLOYEE\_EMAIL, EMPLOYEE\_ADDRESS, EMPLOYEE\_HIRE\_DATE, JOB\_ID, EMPLOYEE\_WORK\_HOURS)

VALUES (12, 'Lok Ching Wei', '990124-08-9877', '018-5320187', 'M', 'chingwei@gmail.com', 'No. 100A, Taman Perindustrian Oxford, Seksyen 3, 47100 Puchong, Selangor.', to\_date('2022-01-12', 'RRRR-MM-DD'), 1, 8);



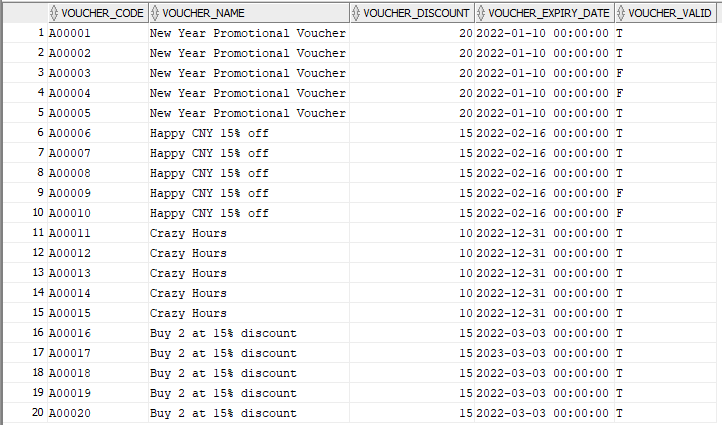
## **2.7 DISTINCT**

**Description:**

**SELECT DISTINCT:** returns only distinct values from the selected table.

**Situation:** List out the unique vouchers that are released by the restaurant.

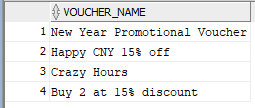
**VOUCHERS TABLE**



**SQL Statement**

**SELECT** **DISTINCT** Voucher\_Name

**FROM** VOUCHERS;



## **2.8 DISTINCT, COUNT(), ALIASES**

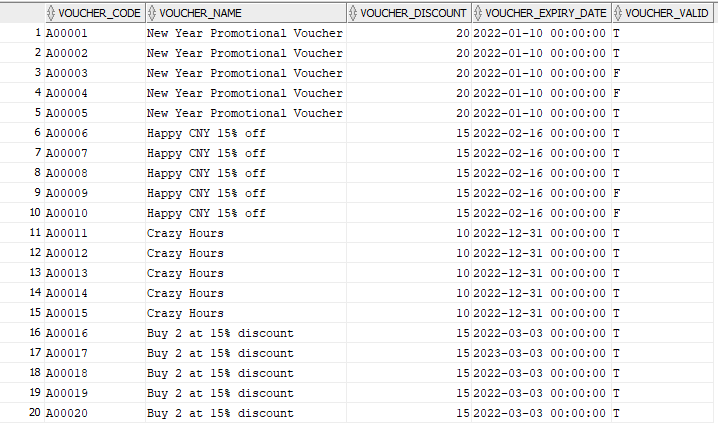
**Description:**

**COUNT():** returns the total number of rows that meets a selected condition.

**ALIASES:** makes the column names to be more readable by using the “AS” keyword.

**Situation:** Calculate the number of unique vouchers that are released by the restaurant.

**VOUCHERS TABLE**



**SQL Statement**

**SELECT COUNT(**DISTINCT Voucher\_Name**) AS** NUMBER\_TYPE\_VOUCHERS

**FROM** VOUCHERS;



## 

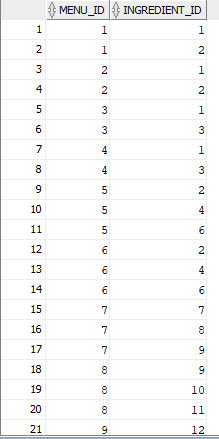
## **2.9 ANY()**

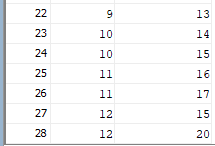
**Description:**

**ANY():** An operator used to perform a comparison between a single column value and a range of other values. This operator will return true if any of the subquery values meet the stated condition.

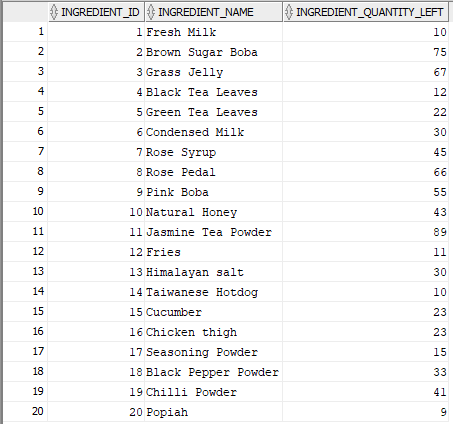
**Situation:** If employees want to search the record of the menu that can be done if they still have ingredients less than 11

**MENU\_INGREDIENTS TABLE**





**INGREDIENTS TABLE**

****

**SQL Statement**

**SELECT** Menu\_ID

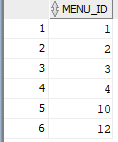
**FROM** MENU\_INGREDIENTS

**WHERE** Ingredient\_ID = **ANY**

(**SELECT** Ingredient\_ID

**FROM** INGREDIENTS

**WHERE** Ingredient\_Quantity\_Left < 11);



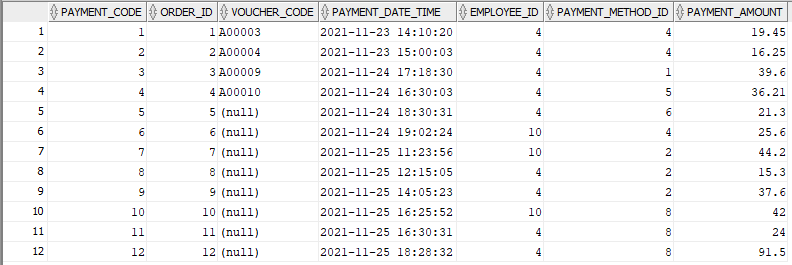
## **2.10 UNION**

**Description:**

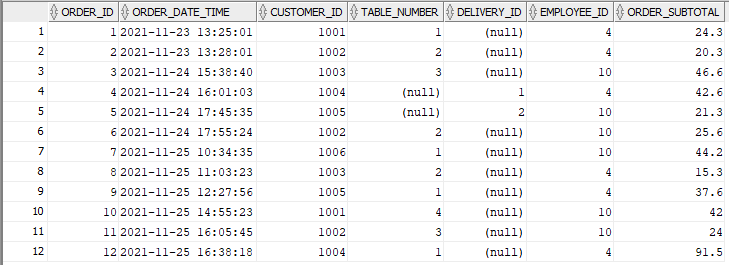
**UNION:** An operator that is used to combine the result-set of two or more SELECT statements. The column in the select statement within the union operator must have similar data types, the same number of columns picked, and in the same order.

**Situation:** If the employee wants to search the information of all the order and payment date-time

**PAYMENTS TABLE**

****

**ORDERS TABLE**

****

**SQL Statement**

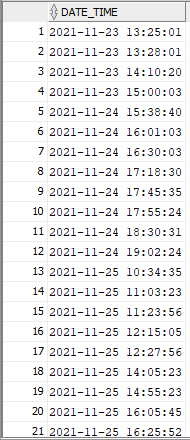
**SELECT** Payment\_Date\_Time **AS** Date\_time

**FROM** PAYMENTS

**UNION**

(**SELECT** Order\_Date\_Time

**FROM** ORDERS);





## **2.11 MIN(), BETWEEN**

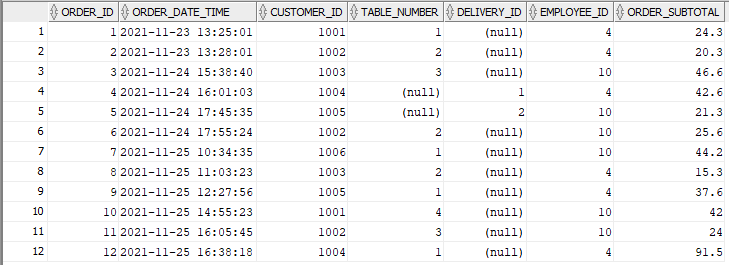
**Description:**

**MIN():** Min()function returns the smallest value of the selected column.

**BETWEEN:** Between operator will select values within the specified range and those values can be numbers, text, or dates.

**Situation:** The employer wants to check the lowest order amount that is made by the customer between ‘2021-11-24 00:00:00’ and ‘2021-11-25 23:59:59’.

**ORDERS TABLE**

****

**SQL Statement**

**SELECT** **MIN(**Order\_Subtotal**) AS** Lowest\_Order\_Amount

**FROM** ORDERS

**WHERE** Order\_Date\_Time **BETWEEN** '2021-11-24 00:00:00' **AND** '2021-11-25 23:59:59';



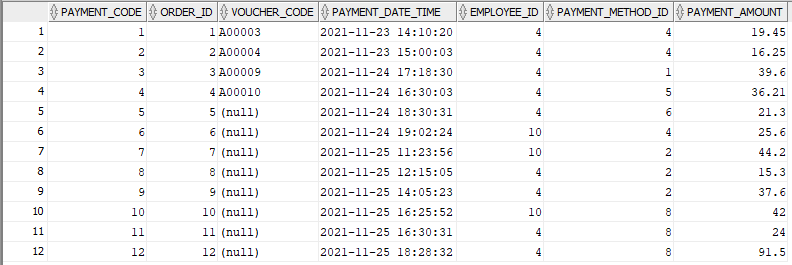
## **2.12 MAX(), BETWEEN**

**Description:**

**MAX():** returns the largest value of the selected column.

**Situation:** The employer wants to check the highest payment amount that is made by the customer between ‘2021-11-23 00:00:00’ and ‘2021-11-24 23:59:59’.

**PAYMENTS TABLE**



**SQL Statement**

**SELECT** **MAX(**Payment\_Amount**)** **AS** Highest\_Payment\_Amount

**FROM** PAYMENTS

**WHERE** Payment\_Date\_Time **BETWEEN** '2021-11-23 00:00:00' **AND** '2021-11-24 23:59:59';



## **2.13 SELECT, INNER JOIN**

**Description:**

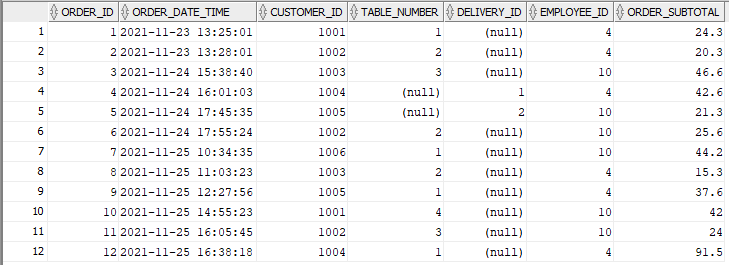
**INNER JOIN:** select records that have matching values in both tables.

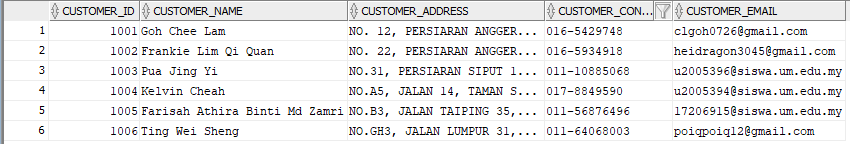
**Situation:** When the customers do the ordering online, the restaurant will provide the delivery service to the customers. In each delivery service, we will provide the information of the delivery man and customer.

**DELIVERY TABLE**

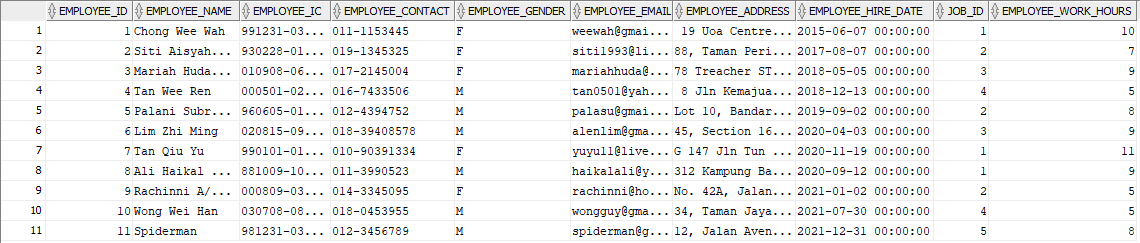
****

**ORDERS TABLE**

****

**CUSTOMERS TABLE**

**EMPLOYEES TABLE**

****

**SQL Statement:**

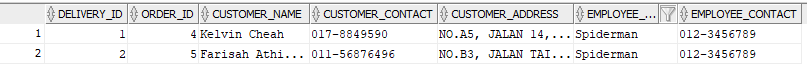
**SELECT** DELIVERY.Delivery\_ID, ORDERS.Order\_ID, CUSTOMERS.Customer\_Name, CUSTOMERS.Customer\_Contact, CUSTOMERS.Customer\_Address, EMPLOYEES.Employee\_Name, EMPLOYEES.Employee\_Contact

**FROM** (((ORDERS

**INNER JOIN** CUSTOMERS **ON** ORDERS.Customer\_ID = CUSTOMERS.Customer\_ID)

**INNER JOIN** DELIVERY **ON** ORDERS.Delivery\_ID = DELIVERY.Delivery\_ID)

**INNER JOIN** EMPLOYEES **ON** Delivery.Employee\_ID = EMPLOYEES.Employee\_ID);



## 

## **2.14 EXIST, HAVING, COUNT(), GROUP BY**

**Description:**

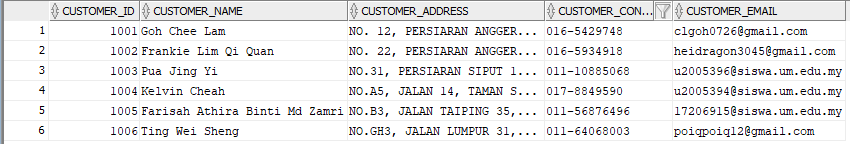
**EXIST:** used to test for the existence of any record in a subquery and returns TRUE if the subquery returns one or more records.

**HAVING:** The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

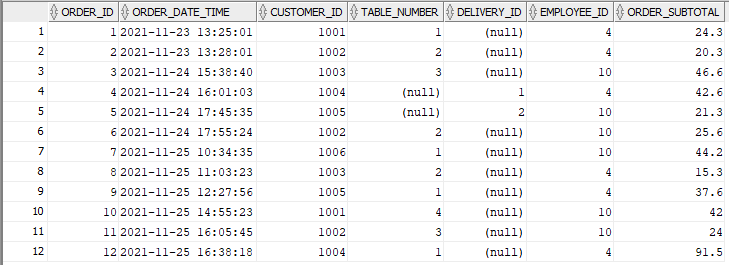
**GROUP BY:** groups rows that have the same values into summary rows.

**Situation:** Employees want to know the customer that orders more than 2 orders (frequent customer).

**CUSTOMERS TABLE**



**ORDERS TABLE**

****

**SQL Statement**

**SELECT** Customer\_ID, Customer\_Name

**FROM** CUSTOMERS

**WHERE EXISTS** (

**SELECT COUNT**(Order\_ID)

**FROM** ORDERS

**WHERE** ORDERS.Customer\_ID = CUSTOMERS.Customer\_ID

**GROUP BY** Customer\_ID

**HAVING COUNT**(Order\_ID)>2);



## **2.15 SELECT, MULTIPLY (SQL Arithmetic Operator), ALIASES, LEFT JOIN, ORDER BY, SUM(), GROUP BY**

**Description:**

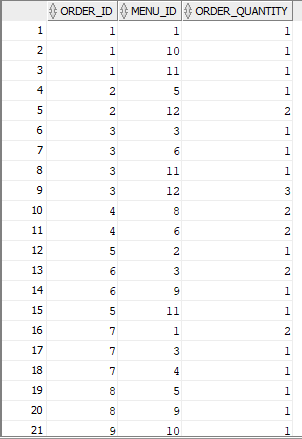
**LEFT JOIN:** returns all records from the left table (table1), and the matching records from the right table (table2). If there is no match, the result is 0 records from the right side.

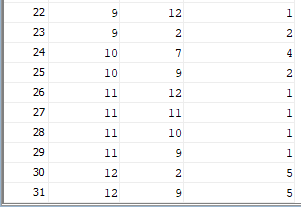
**ORDER BY:** used to sort the result-set in ascending or descending order.

**SUM():** returns the total sum of a numeric column.

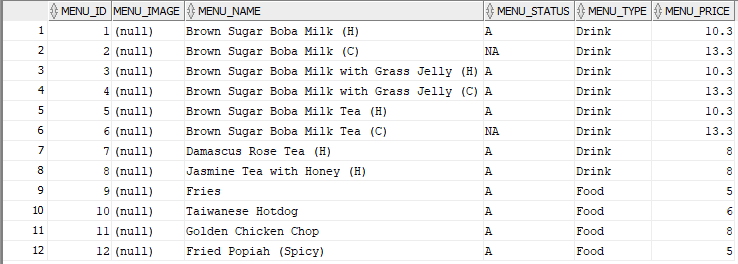
**Situation 1:** Calculate the total price of each menu listed in each order based on the quantity of the menu.

**ORDER\_MENU TABLE**

****

****

**MENU TABLE**

****

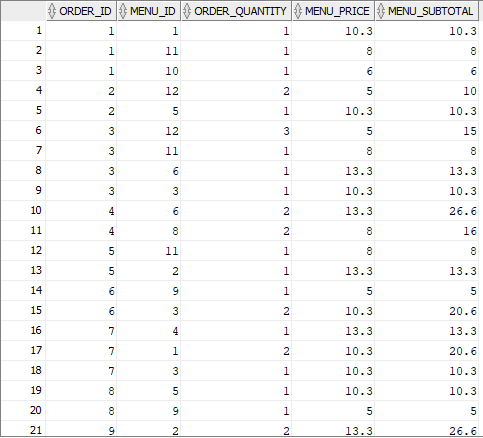
**SQL Statement**

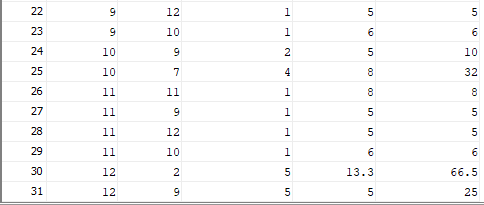
**SELECT** ORDER\_MENU.Order\_ID, ORDER\_MENU.Menu\_ID, ORDER\_MENU.Order\_Quantity, MENU.Menu\_Price, ORDER\_MENU.Order\_Quantity **\*** MENU.Menu\_Price AS Menu\_Subtotal

**FROM** ORDER\_MENU

**LEFT JOIN** MENU **ON** ORDER\_MENU.Menu\_ID = MENU.Menu\_ID

**ORDER BY** ORDER\_MENU.Order\_ID;





**Situation 2:** Calculate the total price of each order before payment or using vouchers.

**SQL Statement**

**SELECT** Order\_ID, **SUM(**Menu\_Subtotal**)** **AS** Order\_Subtotal

**FROM**

**(SELECT** ORDER\_MENU.Order\_ID, ORDER\_MENU.Menu\_ID, ORDER\_MENU.Order\_Quantity, MENU.Menu\_Price, ORDER\_MENU.Order\_Quantity **\*** MENU.Menu\_Price **AS** Menu\_Subtotal

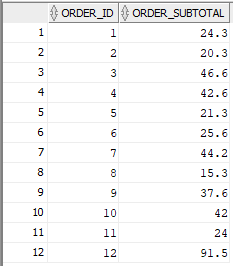
**FROM** ORDER\_MENU

**LEFT JOIN** MENU **ON** ORDER\_MENU.Menu\_ID = MENU.Menu\_ID

**ORDER BY** ORDER\_MENU.Order\_ID**)**

**GROUP BY** Order\_ID

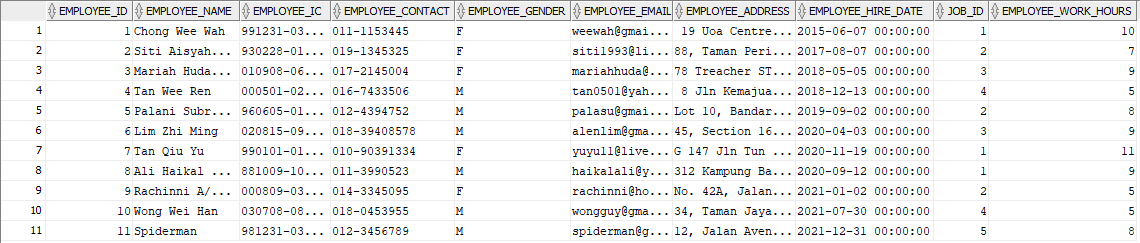
**ORDER BY** Order\_ID;



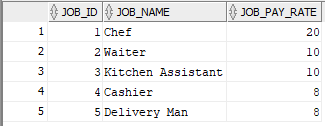
## **2.16 SELECT, MULTIPLY (SQL Arithmetic Operator), ALIASES, INNER JOIN, ORDER BY, DESCENDING**

**Situation**: The employer is responsible to pay the salary to each employee once per week. Calculate each employees’ salary per week according to their job type in descending order.

**EMPLOYEES TABLE**

****

**JOBS TABLE**

****

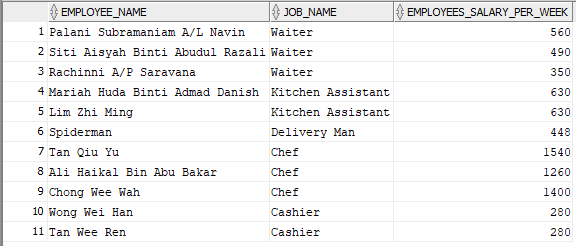
**SQL Statement**

**SELECT** EMPLOYEES.Employee\_Name, JOBS.Job\_Name, EMPLOYEES.Employee\_Work\_Hours **\*** JOBS.Job\_Pay\_Rate **\*** 7 **AS** Employees\_Salary\_Per\_Week

**FROM** EMPLOYEES

**INNER JOIN** JOBS **ON** EMPLOYEES.Job\_ID = JOBS.Job\_ID

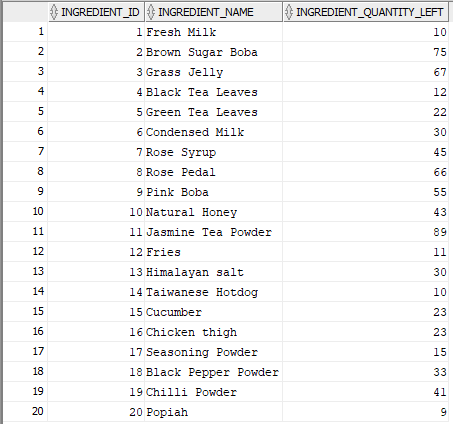
**ORDER BY** JOBS.Job\_Name DESC;



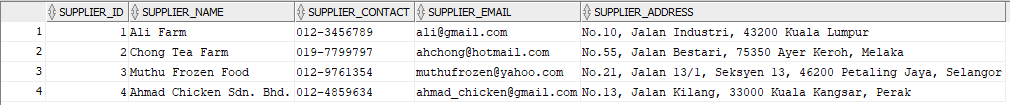
## **2.17 SELECT, SUM(), ALIASES, ADD (SQL Arithmetic Operator), INNER JOIN, WHERE, ORDER BY, GROUP BY**

**Situation 1:** The employer or manager will order the ingredients with the suppliers when the quantity of each ingredient left is less than equal 30. Calculate the total quantity of each ingredient after adding the quantity of each ingredient left with each ingredient supplied by the suppliers.

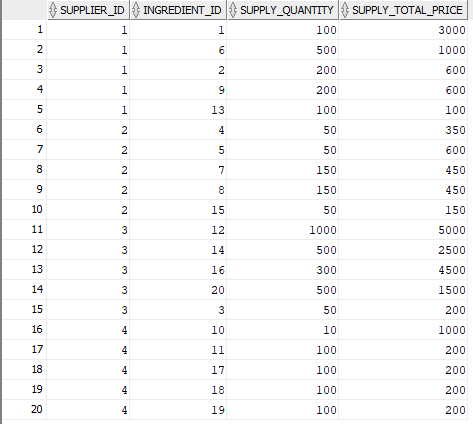
**INGREDIENTS TABLE**



**SUPPLIERS TABLE**

****

**SUPPLY TABLE**

****

**SQL Statement**

**SELECT** INGREDIENTS.Ingredient\_Name, SUPPLIERS.Supplier\_Name, SUPPLY.Supply\_Quantity, SUPPLY.Supply\_Total\_Price, INGREDIENTS.Ingredient\_Quantity\_Left **+** SUPPLY.Supply\_Quantity **AS** Total\_Ingredient\_Quantity

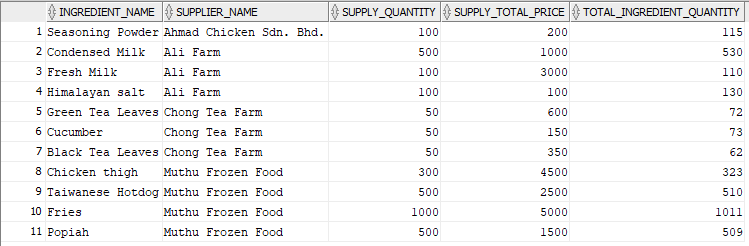
**FROM** SUPPLY

**INNER JOIN** INGREDIENTS **ON** SUPPLY.Ingredient\_ID = INGREDIENTS.Ingredient\_ID

**INNER JOIN** SUPPLIERS **ON** SUPPLY.Supplier\_ID = SUPPLIERS.Supplier\_ID

**WHERE** INGREDIENTS.Ingredient\_Quantity\_Left <= 30

**ORDER BY** SUPPLIERS.Supplier\_Name;



**Continue Situation 1:** After the employer receives the ingredients from the suppliers, the employer needs to pay the suppliers. Calculate the total price that is needed to pay to each supplier.

**SQL Statement**

**SELECT** Supplier\_Name, SUM(Supply\_Total\_Price) **AS** Total\_Price\_Paid\_To\_Supplier

**FROM(**

**SELECT** INGREDIENTS.Ingredient\_Name, SUPPLIERS.Supplier\_Name, SUPPLY.Supply\_Quantity, SUPPLY.Supply\_Total\_Price, INGREDIENTS.Ingredient\_Quantity\_Left **+** SUPPLY.Supply\_Quantity **AS** Total\_Ingredient\_Quantity

**FROM** SUPPLY

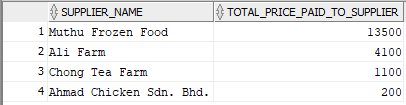
**INNER JOIN** INGREDIENTS **ON** SUPPLY.Ingredient\_ID = INGREDIENTS.Ingredient\_ID

**INNER JOIN** SUPPLIERS **ON** SUPPLY.Supplier\_ID = SUPPLIERS.Supplier\_ID

**WHERE** INGREDIENTS.Ingredient\_Quantity\_Left <= 30

**ORDER BY** Suppliers.Supplier\_Name)

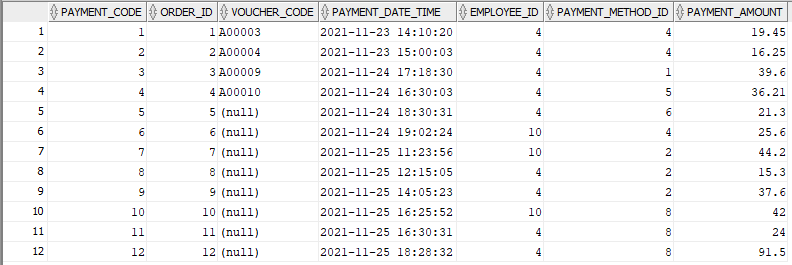
**GROUP BY** Supplier\_Name;



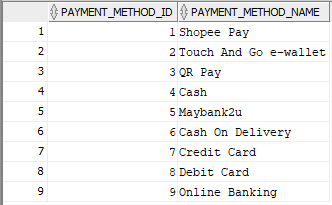
## **2.18 SELECT, COUNT(), SUM(), AVG(), ALIASES, LEFT JOIN, GROUP BY, ORDER BY, DESCENDING**

**Situation:** The employer can calculate the total payments by using each payment method that is made by the customers. Besides, the employer also can calculate the number of payments using each payment method and the average price in each payment by using the payment method.

**PAYMENTS TABLE**

****

**PAYMENT\_METHODS TABLE**



**SQL Statement**

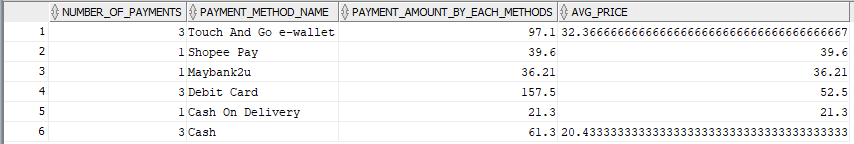
**SELECT COUNT(**PAYMENTS.Payment\_Code**) AS** Number\_Of\_Payments, PAYMENT\_METHODS.Payment\_Method\_Name, **SUM(**PAYMENTS.Payment\_Amount**) AS** Payment\_Amount\_By\_Each\_Methods, **AVG(**PAYMENTS.Payment\_Amount**)** **AS** AVG\_Price

**FROM** PAYMENTS

**LEFT JOIN** PAYMENT\_METHODS **ON** PAYMENTS.Payment\_Method\_ID = PAYMENT\_METHODS.Payment\_Method\_ID

**GROUP BY** Payment\_Method\_Name

**ORDER BY** Payment\_Method\_Name DESC;



# **2.0 Database Testing**

| **No** | **Date** | **Test Description** | **Input** | **Expected Output** | **Result** | **Action** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 27/12/2021 | Column testing- Validation of the presence of any unused/ unmapped database tables/ columns | Voucher, Payment, Employee, Job, Payment\_Method, Order, Customer, Delivery, Order\_Menu, Menu, Ingredient, Supply, Supplier, Menu\_Ingredient | All the tables created are mapped to the database tables. | The output is just like the expected outcome. | None |
| 2 | 27/12/2021 | Validation for all the constraints (Primary key and Foreign key) between the database table | Voucher\_Code, Payment\_Code, Employee\_ID, Job\_ID, Payment\_Method\_ID, Order\_ID, Customer\_ID, Menu\_ID, Ingredient\_ID, Supplier\_ID | All the constraints created are perfectly joined in the database table | The output is just like the expected outcome. | None |
| 3 | 05/01/2022 | Validation of the compatibility of the data type and field lengths. | All the data is inserted into each of the tables with a minimum of one set of data in each table | All the data is inserted into each of the tables with a minimum of 5 data in each table | Some of the data types required modification and some of the field lengths were too huge. (waste space) | Change the data type to the compatible type and fixed the field lengths just suitable to the data. |
| 4 | 10/01//2022 | Checking whether the data queries are logically well organized and the database tables are well implemented | All the data, records, table, constraints and queries inside the database tables | All the data, records, table, constraints and inside the database tables are well implemented. | The output is just like the expected outcome. | None |

# **3.0 Difficulties and Problems**

## **3.1 Problems:**

1. The data of business processes and business rules collected are not completed during the data collection process.
2. No idea on how to design a database based on the data collected before.
3. Not familiar with the knowledge of designing the database using SQL statements.
4. Limitation of communication during the discussion because some of our group members practice online learning at home.
5. Time constraints in completing this assignment due to heavy workload from other courses.
6. Errors on the implementable Entity Relationship Diagram.

## **3.2 Solutions:**

1. We solved this problem by conducting several physical and virtual meetings with the manager of Ke Nina Cafe to get more information needed.
2. We have a group discussion with our group members, and also seek opinions and suggestions from other friends and our seniors in order to have a clear understanding of how to design the database.
3. We do the self-learning session on the SQL tutorial provided by the W3schools online learning platform.
4. We have meetings every weekend to make sure all of us understand the assignment task and work in the correct direction.
5. We distribute our tasks to each of our group members and set a due date for the tasks to ensure our progress meets the schedule that we had planned.
6. We do the correction from the previous version of the implementable Entity Relationship Diagram and replace it with the correct version for the database design using SQL statements.

## **3.3 Future improvement:**

1. Learn cloud-based database management to design an online database so that the employer can access it at any time.
2. Approach the company earlier to get business processes and business rules to ensure more features can be added to the database.
3. Improve database to store the information of each branch of the restaurant if the employer opens new branches.
4. Improve database to record the data of all the managers of each branch.
5. Improve this database to help the employer or managers to record the ingredients expiry date so that they will not need to remember or record using spreadsheets like Excel.

## **3.4 Lesson learnt:**

1. Time management is important for all members of the team to complete the assignment on time.
2. Cooperation between group members can help us solve the problems we face more easily.
3. There are many resources we can find from online platforms such as the W3schools platform.

# **4.0 Peer Work Group Evaluation Form**

Course: WIA2001 Semester: 1Session: 2021/2022

Lecturer: DR. FARIZA HANUM BINTI MD NASARUDDIN

Assignment:**Group Project**

Evaluator (Student’s Name): **FRANKIE LIM QI QUAN**

Date: 13 JANUARY 2022

Group Members:

|  | Matrix Number | Name |
| --- | --- | --- |
| 1 | U2005382/1 | GOH CHEE LAM |
| 2 | 17206915/2 | FARISAH ATHIRA BINTI MD ZAMRI |
| 3 | U2005396/1 | PUA JING YI |
| 4 | U2005394/1 | KELVIN CHEAH |

Directions: In the space below, honestly evaluate the work of other students in your group by

answering **yes** or **no** and by using a scale from 1 to 3, **1 being poor, 2 being average, 3 being**

**above average. Please circle your answer.**

|  |  | Group Member 1 | Group Member 2 | Group Member 3 | Group Member 4 |
| --- | --- | --- | --- | --- | --- |
| 1 | Did this group member complete his/her assigned tasks for the group | Yes | Yes | Yes | Yes |
| 2 | How would you rate the quality of this person’s work | 3 | 3 | 3 | 3 |
| 3 | How would you rate the timeliness of the completion of the work? | 3 | 3 | 3 | 3 |
| 4 | How would you rate the accuracy of the work | 3 | 3 | 3 | 3 |
| 5 | Overall, how would you rank this group member’s performance in the group? | 3 | 3 | 3 | 3 |
| 6 | Would you want to work with this person again?  Explain why in the space below. | Yes | Yes | Yes | Yes |
| Perfect group members. | | | |

Course: WIA2001 Semester: 1Session: 2021/2022

Lecturer: DR. FARIZA HANUM BINTI MD NASARUDDIN

Assignment:**Group Project**

Evaluator (Student’s Name): **PUA JING YI**

Date: 14 JANUARY 2022

Group Members:

|  | Matrix Number | Name |
| --- | --- | --- |
| 1 | U2005382/1 | GOH CHEE LAM |
| 2 | U2005263/1 | FRANKIE LIM QI QUAN |
| 3 | 17206915/2 | FARISAH ATHIRA BINTI MD ZAMRI |
| 4 | U2005394/1 | KELVIN CHEAH |

Directions: In the space below, honestly evaluate the work of other students in your group by

answering **yes** or **no** and by using a scale from 1 to 3, **1 being poor, 2 being average, 3 being**

**above average. Please circle your answer.**

|  |  | Group Member 1 | Group Member 2 | Group Member 3 | Group Member 4 |
| --- | --- | --- | --- | --- | --- |
| 1 | Did this group member complete his/her assigned tasks for the group | Yes | Yes | Yes | Yes |
| 2 | How would you rate the quality of this person’s work | 3 | 3 | 3 | 3 |
| 3 | How would you rate the timeliness of the completion of the work? | 3 | 3 | 3 | 3 |
| 4 | How would you rate the accuracy of the work | 3 | 3 | 3 | 3 |
| 5 | Overall, how would you rank this group member’s performance in the group? | 3 | 3 | 3 | 3 |
| 6 | Would you want to work with this person again?  Explain why in the space below. | Yes | Yes | Yes | Yes |
| Clear division of work and help each other between the group members. | | | |

Course: WIA2001 Semester: 1 Session: 2021/2022

Lecturer: DR. FARIZA HANUM BINTI MD NASARUDDIN

Assignment: **Group Project**

Evaluator (Student’s Name): **Farisah Athira Binti Md Zamri**

Date: 13/02/2022

Group Members:

|  | Matrix Number | Name |
| --- | --- | --- |
| 1 | U2005263/1 | FRANKIE LIM QI QUAN |
| 2 | U2005382/1 | GOH CHEE LAM |
| 3 | U2005396/1 | PUA JING YI |
| 4 | U2005394/1 | KELVIN CHEAH |

Directions: In the space below, honestly evaluate the work of other students in your group by

answering **yes** or **no** and by using a scale from 1 to 3, **1 being poor, 2 being average, 3 being**

**above average. Please circle your answer.**

|  |  | Group Member 1 | Group Member 2 | Group Member 3 | Group Member 4 |
| --- | --- | --- | --- | --- | --- |
| 1 | Did this group member complete his/her assigned tasks for the group | Yes No | Yes No | Yes No | Yes No |
| 2 | How would you rate the quality of this person’s work | 1 2 3 | 1 2 3 | 1 2 3 | 1 2 3 |
| 3 | How would you rate the timeliness of the completion of the work? | 1 2 3 | 1 2 3 | 1 2 3 | 1 2 3 |
| 4 | How would you rate the accuracy of the work | 1 2 3 | 1 2 3 | 1 2 3 | 1 2 3 |
| 5 | Overall, how would you rank this group member’s performance in the group? | 1 2 3 | 1 2 3 | 1 2 3 | 1 2 3 |
| 6 | Would you want to work with this person again?  Explain why in the space below. | Yes No | Yes No | Yes No | Yes No |
| Because all of them are very cooperative and contribute to the group. | | | |

Course: WIA2001 Semester: 1 Session: 2021/2022

Lecturer: DR. FARIZA HANUM BINTI MD NASARUDDIN

Assignment: **Group Project**

Evaluator (Student’s Name): **GOH CHEE LAM**

Date: 13 JANUARY 2022

Group Members:

|  | Matrix Number | Name |
| --- | --- | --- |
| 1 | U2005263/1 | FRANKIE LIM QI QUAN |
| 2 | 17206915/2 | FARISAH ATHIRA BINTI MD ZAMRI |
| 3 | U2005396/1 | PUA JING YI |
| 4 | U2005394/1 | KELVIN CHEAH |

Directions: In the space below, honestly evaluate the work of other students in your group by

answering **yes** or **no** and by using a scale from 1 to 3, **1 being poor, 2 being average, 3 being**

**above average. Please circle your answer.**

|  |  | Group Member 1 | Group Member 2 | Group Member 3 | Group Member 4 |
| --- | --- | --- | --- | --- | --- |
| 1 | Did this group member complete his/her assigned tasks for the group | Yes | Yes | Yes | Yes |
| 2 | How would you rate the quality of this person’s work | 3 | 3 | 3 | 3 |
| 3 | How would you rate the timeliness of the completion of the work? | 3 | 3 | 3 | 3 |
| 4 | How would you rate the accuracy of the work | 3 | 3 | 3 | 3 |
| 5 | Overall, how would you rank this group member’s performance in the group? | 3 | 3 | 3 | 3 |
| 6 | Would you want to work with this person again?  Explain why in the space below. | Yes | Yes | Yes | Yes |
| I like to work with them because they show high responsibility in doing the project. | | | |

Course: WIA2001 Semester: 1 Session: 2021/2022

Lecturer: DR. FARIZA HANUM BINTI MD NASARUDDIN

Assignment: **Group Project**

Evaluator (Student’s Name): **KELVIN CHEAH**

Date: 13 JANUARY 2022

Group Members:

|  | Matrix Number | Name |
| --- | --- | --- |
| 1 | U2005263/1 | FRANKIE LIM QI QUAN |
| 2 | 17206915/2 | FARISAH ATHIRA BINTI MD ZAMRI |
| 3 | U2005396/1 | PUA JING YI |
| 4 | U2005382/1 | GOH CHEE LAM |

Directions: In the space below, honestly evaluate the work of other students in your group by

answering **yes** or **no** and by using a scale from 1 to 3, **1 being poor, 2 being average, 3 being**

**above average. Please circle your answer.**

|  |  | Group Member 1 | Group Member 2 | Group Member 3 | Group Member 4 |
| --- | --- | --- | --- | --- | --- |
| 1 | Did this group member complete his/her assigned tasks for the group | Yes | Yes | Yes | Yes |
| 2 | How would you rate the quality of this person’s work | 3 | 3 | 3 | 3 |
| 3 | How would you rate the timeliness of the completion of the work? | 3 | 3 | 3 | 3 |
| 4 | How would you rate the accuracy of the work | 3 | 3 | 3 | 3 |
| 5 | Overall, how would you rank this group member’s performance in the group? | 3 | 3 | 3 | 3 |
| 6 | Would you want to work with this person again?  Explain why in the space below. | Yes | Yes | Yes | Yes |
| I like to work with them because they are cooperative. | | | |