



Group Assignment Part B

AICT005-4-1 Database Systems

Intake Code: UCDF2104 ICT(SE)

Members:	Teo Jie Ing	TP063330
	Yap Huei Ling	TP064464
	Tiew Cheng Jia	TP064043
	Yap Yoong	TP064344

Lecturer Name: Sathiapriya Ramiah

Date Assigned: 1 March 2022

Date of Submit: 5 May 2022

Introduction

This assignment shows the steps that how we design and implement the database system for APU café food ordering system. First, we investigate the relationship between each entity and attribute for drawing an entity relationship diagram. After that, we also draw a database diagram to understand the relationship between the table that we will create in SQL Server Management Studio and define the primary key and foreign key in each table. Related to the case study, we create five tables which are Member, Orders, Food, Chef, Manager. In addition to basic information like ID and name, we also create column to receive alternative information such as rating and review from the member and orders.

Database Schema

Entity Relationship Diagram

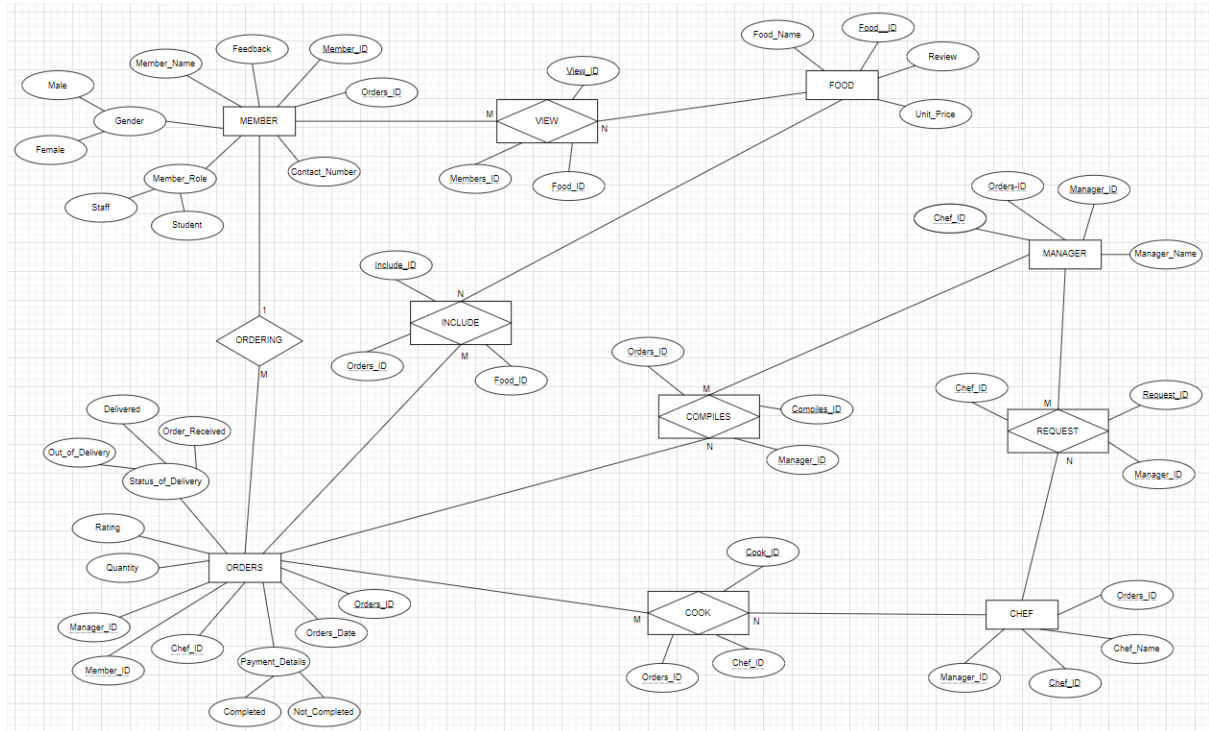


Figure 1: Entity Relationship Diagram

Database Diagram

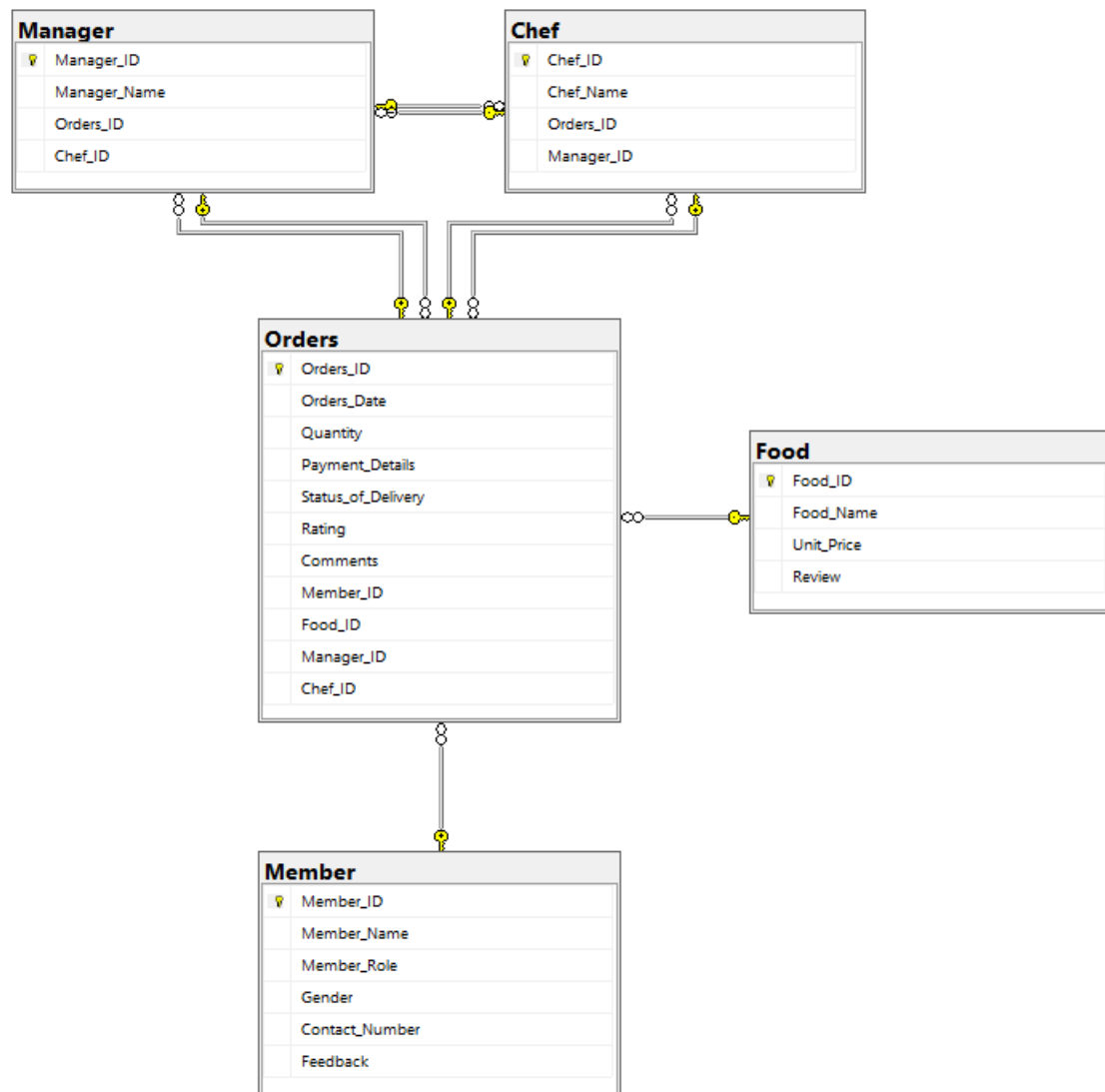


Figure 2: Database Diagram

SQL-Data Definition Language (DDL)

Teo Jie Ing

```
create database APU_Café_Food_Ordering_System  
collate SQL_Latin1_General_CP1_CS_AS;
```

Figure 3: Query of Creating Database APU Café Food Ordering System

To begin, we must construct a database called APU Café Food Ordering System. To ensure that the characters included in range searches are dependent on the collation's sorting criteria, we include a collate clause for case sensitive in the columns. The Latin1 General CS AS collation employs dictionary order to include both upper- and lower-case characters within the defined range.

```
use APU_Café_Food_Ordering_System  
  
create table Member(  
  Member_ID varchar(10) not null Primary Key check(Member_ID like ('TP____')),  
  Member_Name varchar(15) not null,  
  Member_Role varchar(10) not null check(Member_Role in('Staff','Student')),  
  Gender varchar(10) not null check(Gender in('Male','Female')),  
  Contact_Number varchar(15) not null,  
  Member_Address varchar(100) not null,  
  Feedback varchar(30) not null  
);
```

Figure 4: Query of Table Member

Then we used the use command to choose a specific database, which was the previous database we created, and then we used SQL's built-in commands to perform operations on it. The query for generating table member, which includes member id, member name, member role, gender, contact number, member address, and feedback, is shown in the image above. Member ID is the primary key in this query. We also utilised a check constraint on member id, member role, and gender to limit the value range that can be placed in a column. When members check in, ensure sure the member id is formatted correctly ('TP'____). There is an option for member, staff, or student, male or female, for the Role and Gender. To make it easier, we used VARCHAR for each. This is because of it's a string of varying length that can include letters, numbers, and special characters.

```

insert into Member(Member_ID, Member_Name, Member_Role, Gender, Contact_Number,Member_Address, Feedback)
values('TP063290','Chloe','Student','Female','0126565435','3 jalan junid taman perdana 84000 muar johor','Good taste. '),
('TP039487','Jackson Wang','Student','Male','0127338732','4 jalan junid taman marin 84000 muar johor','Excellent food. '),
('TP028476','Jessica','Staff','Female','0178265692','24 jalan terangganu taman hijau 52100 Kuala Lumpur','Wonderful ordering system'),
('TP092739','Oman Ishah','Staff','Male','0186730284','8 alan terangganu taman hijau 52100 Kuala Lumpur','Delicious!!!'),
('TP092765','Mohammad Ali','Student','Male','0197829364','77 jalan hitam taman putih 52100 Kuala Lumpur','Amaazzzzing'),
('TP103863','Tiffany Tan','Student','Female','0157284534','99 jalan kuning taman utama 84000 muar johor','Delicious and high quality'),
('TP783614','Lavenda','Staff','Female','0126584957','4 taman pertama jalan sungai abong 84000 muar johor','Perfect. ');

```

Figure 5: Query of Insert Data into Table Member

We begin inserting values into the member table when we finish constructing table members. The image above depicts the information that we enter and the necessity to double-check that it is correct.

Results		Messages				
	Member_ID	Member_Name	Member_Role	Gender	Contact_Number	Feedback
1	TP028476	Jessica	Staff	Female	0178265692	Wonderful ordering system
2	TP039487	Jackson Wang	Student	Male	0127338732	Excellent food.
3	TP063290	Chloe	Student	Female	0126565435	Good taste.
4	TP092739	Oman Ishah	Staff	Male	0186730284	Delicious!!!
5	TP092765	Mohammad Ali	Student	Male	0197829364	Amaazzzzing
6	TP103863	Tiffany Tan	Student	Female	0157284534	Delicious and high quality
7	TP783614	Lavenda	Staff	Female	0126584957	Perfect.

Figure 6: Screenshot of Output

Yap Yoong

```
use APU_Café_Food_Ordering_System
create table Food
(
    Food_ID varchar(15) not null PRIMARY KEY,
    Food_Name varchar(50) not null,
    Unit_Price int not null,
    Review varchar(50) null,
);
```

Figure 7: Query of Creating Table Food

We had to choose a database by using command “Use” to locate the database we going to use, then we create and new table name as “Food” and enter the “Food_ID”, “Food_Name”, “Unit_Price” and “Review”. The primary key will be the “Food_ID” and all the data type will be Varchar(X) so it will be easier to insert data excluding “Unit_Price “because it is only number, so I put the data type is integer.

```
insert into Food values
(001,'Chicken Salad', 12, 'Chicken with salad and some mayonnaise' ),
(002,'Beef Burger', 20, 'Burger with Beef meat inside'),
(003,'Spaghetti', 16, 'Spaghetti and tomtato sauce'),
(004,'Mushroom Soup',13,'Mushroom soup and 2 piece of garilc bread'),
(005,'Chicken Sandwich',14,'Sandwich with chicken inside'),
(006,'Spicy Fried Chicken',17,'Hot and spciy chicken wings'),
(007,'Smoked Salmon',21,'A different fav of salmon'),
(008,'Grill Lamb',32,'Grill a whole Lamb leg for you')
```

Figure 8: Query of Insert Data into Table Food

After finish creating the table, we insert data into the Food table. Based on the sequence of insert query of column name, key in the data accordingly, and put the correct data in the correct column.

	Food_ID	Food_Name	Unit_Price	Review
1	1	Chicken Salad	12	Chicken with salad and some mayonnaise
2	2	Beef Burger	20	Burger with Beef meat inside
3	3	Spaghetti	16	Spaghetti and tomtato sauce
4	4	Mushroom Soup	13	Mushroom soup and 2 piece of garilc bread
5	5	Chicken Sandwich	14	Sandwich with chicken inside
6	6	Spicy Fried Chicken	17	Hot and spciy chicken wings
7	7	Smoked Salmon	21	A different fav of salmon
8	8	Grill Lamb	32	Grill a whole Lamb leg for you

Figure 9: Screenshot of Output

This is the outcome for the Food table.

Tiew Cheng Jia

```
Create table Manager(  
  Manager_ID varchar(10) not null Primary Key,  
  Manager_Name varchar(50) not null,  
  Orders_ID varchar(10) null,  
  Foreign key (Orders_ID) references Orders  
  On delete cascade  
  On Update cascade  
);
```

Figure 10: Query of Creating Table Manager

After the member table is created, we create a new table name Manager. The figure shows the query for creating the table which includes Manager_ID, Manager_Name and Orders_ID. Manager_ID is the primary key of the table in which the format is MXXX. The second column is Manager_Name which is the name of the manager. Furthermore, we also create the third column Orders_ID as the foreign key of the table. Therefore, we can check the orders according to the manager.

```
Create table Chef(  
  Chef_ID varchar(10) not null Primary Key,  
  Chef_Name varchar(50) not null,  
  Orders_ID varchar(10) null,  
  Manager_ID varchar(10) null,  
  foreign key (Orders_ID) references Orders,  
  foreign key (Manager_ID) references Manager  
  on delete cascade  
  on update cascade  
);
```

Figure 11: Query of Creating Table Chef

The table that created after manager is chef. There are four columns that have been created, Chef_ID, Chef_Name, Manager_ID, and Orders_ID. Each chef in the café has their own id, it is also the primary key of this table. The coulumn Chef_Name is referring to the name of our chefs. There are two foreign keys in this table which are Manager_ID and Order_ID. This because each manager will lead a different chef and each chef will be responsible for their own orders.


```

insert into Manager(Manager_ID, Manager_Name)
values('M001','David'),
('M002','Sean'),
('M003','Chloe'),
('M004','John'),
('M005','Jack');

```

Figure 12: Query of Insert Data into Table Manager

```

Insert into Chef(Chef_ID, Chef_Name, Manager_ID)
values('C001','Michael','M001'),
('C002','Bryant','M002'),
('C003','Durant','M003'),
('C004','Morant','M004'),
('C005','Kelvin','M005');

```

Figure 13: Query of Insert Data into Table Chef

We inserted the data into these tables after it were created.

```

Select Manager.Manager_ID, Manager.Manager_Name, Orders.Orders_ID, Orders.Chef_ID
from Manager join Orders
on Manager.Manager_ID = Orders.Manager_ID;

```

Figure 14: Query of Table Manager join Table Orders

```

Select Chef.Chef_ID, Chef.Chef_Name, Orders.Orders_ID, Orders.Manager_ID
from Chef join Orders
on Chef.Chef_ID = Orders.Chef_ID;

```

Figure 15: Query of Table Chef join Table Orders

We use the statement inner join to display the output. We selected the table manger to join the table orders to display Manager_ID, Manager_Name, Orders_ID and Chef_ID. This is because we faced a problem when we trying to insert multiple Orders_ID in one cell.

	Chef_ID	Chef_Name	Orders_ID	Manager_ID
1	C001	Michael	S0001	M001
2	C002	Bryant	S0002	M002
3	C003	Durant	S0003	M003
4	C004	Morant	S0004	M004
5	C005	Kelvin	S0005	M005
6	C001	Michael	S0006	M001
7	C002	Bryant	S0007	M002
8	C003	Durant	S0008	M003
9	C004	Morant	S0009	M004
10	C004	Morant	S0010	M004
11	C005	Kelvin	S0011	M005
12	C005	Kelvin	S0012	M005
13	C001	Michael	S0013	M001

Figure 16: Screenshot of Output

	Manager_ID	Manager_Name	Orders_ID	Chef_ID
1	M001	David	S0001	C001
2	M002	Sean	S0002	C002
3	M003	Chloe	S0003	C003
4	M004	John	S0004	C004
5	M005	Jack	S0005	C005
6	M001	David	S0006	C001
7	M002	Sean	S0007	C002
8	M003	Chloe	S0008	C003
9	M004	John	S0009	C004
10	M004	John	S0010	C004
11	M005	Jack	S0011	C005
12	M005	Jack	S0012	C005
13	M001	David	S0013	C001

Figure 17: Screenshot of Output

Figure 16 shows the output of Table Chef with foreign key Orders_ID and Manager_ID. Figure 17 shows the output of Table Manager with foreign key Orders_ID and Chef_ID. Both of these output was using inner join function to result.

Yap Huei Ling

```
Create table Orders(  
Orders_ID varchar(10) not null Primary Key check (Orders_ID like 'S_____'),  
Orders_Date date not null,  
Quantity int not null,  
Payment_Details varchar(20) not null check (Payment_Details in('Completed', 'Not Completed')),  
Status_of_Delivery varchar(20) not null check (Status_of_Delivery in('Order_Received', 'Out_of_Delivery', 'Delivered')),  
Rating smallint not null check (Rating between 1 and 5),  
Comments varchar (100) null,  
Member_ID varchar (10) null,  
Food_ID varchar (15) null,  
Foreign Key (Member_ID) References Member,  
Foreign Key (Food_ID) References Food  
On delete cascade  
On Update cascade  
);
```

Figure 18: Query of Creating Table Orders

After creating tables Member and Food, we start creating table Orders. Figure 1 shows the query of creating table Orders which include the summary of order, payment details, status of delivery, rating on food, optional short-text comments, MemberID who made the order, ordered food ID, manager ID who compiles the order and the chef ID who handle the order.

The primary key in this table is Orders_ID, we make a check constraint which let the Orders_ID start with 'S' and continue as number, for example the first order ID was S0001. The second column we put Orders_Date which mean the date of order made. Quantity is the number of food(s) ordered which shows as integer in table Orders. The Payment_Details also has check constraint which are 'Completed' and 'Not Completed'. Only the order that made the payment will send to the café, otherwise the order will be failed to receive by the café. Status_of_Delivery will show Order_Received when the order sent; shows Out_of_Delivery when sending the cook meals to member who ordered; Delivered when the meals delivered to the member. Member can rate the food by 1 point to 5 points and give an optional comment on the order. The data in Rating and Comments will be shows in the table after the member rate and comment. Due to the comment was optional, the member can choose to not give comment on the order, then the value(s) will be null. The Member_ID and Food_ID will be the foreign key of table Orders. Member_ID will be the member who made the order, and the Food_ID will be the food that ordered in this order.

```

Alter table Orders add Manager_ID varchar (10) null,
Foreign Key (Manager_ID) References Manager;
Alter table Orders add Chef_ID varchar (10) null,
Foreign Key (Chef_ID) References Chef;

```

Figure 19: Query of Alter Table Orders

After adding the table Manager and table Chef, we alter table Orders by adding Manager_ID and Chef_ID as foreign key. Both data types are varchar (10) and allows null.

```

Insert into Orders(Orders_ID, Orders_Date, Quantity, Payment_Details, Status_of_Delivery, Rating, Comments, Member_ID, Food_ID, Manager_ID, Chef_ID)
values ('S0001', '2022-2-1', '2', 'Completed', 'Delivered', 4, 'It is delicious', 'TP063290', 001, 'M001', 'C001'),
('S0002', '2022-2-1', '4', 'Completed', 'Delivered', 5, 'Good Packaging', 'TP039487', 003, 'M002', 'C002'),
('S0003', '2022-2-2', '1', 'Completed', 'Delivered', 3, 'Everything is good', 'TP028476', 008, 'M003', 'C003'),
('S0004', '2022-2-3', '2', 'Completed', 'Delivered', 4, 'Nice and healthy', 'TP063290', 005, 'M004', 'C004'),
('S0005', '2022-2-3', '3', 'Completed', 'Out_of_Delivery', 1, 'I have been waiting for a long time', 'TP092739', 001, 'M005', 'C005'),
('S0006', '2022-2-4', '3', 'Completed', 'Delivered', 4, 'Tastes great as always', 'TP039487', 003, 'M001', 'C001'),
('S0007', '2022-2-4', '6', 'Completed', 'Order_Received', 2, 'Takes too long time for preparing my order', 'TP092765', 002, 'M002', 'C002'),
('S0008', '2022-2-4', '7', 'Completed', 'Delivered', 1, 'The Fried Chicken was too spicy', 'TP103863', 006, 'M003', 'C003'),
('S0009', '2022-2-4', '4', 'Completed', 'Delivered', 5, 'Food standards and portion smaller than before', 'TP039487', 007, 'M004', 'C004'),
('S0010', '2022-2-5', '1', 'Completed', 'Delivered', 4, 'The food was delicious but got one strand of hair inside it', 'TP783614', 004, 'M004', 'C004'),
('S0011', '2022-2-5', '2', 'Completed', 'Order_Received', 5, 'Well pacakaging, it is pro-environment', 'TP063290', 003, 'M005', 'C005'),
('S0012', '2022-2-5', '3', 'Completed', 'Order_Received', 2, 'Why preparing time so long', 'TP028476', 007, 'M005', 'C005'),
('S0013', '2022-2-5', '1', 'Completed', 'Order_Received', 3, null, 'TP092739', 005, 'M001', 'C001');

```

Figure 20: Query of inset data into table Orders

Starting insert data into table Orders after the name of table and column inserted. Based on the sequence of insert query of column name, key in the data accordingly, and make sure all of the data follow the data type and format.

Results		Messages									
	Orders_ID	Orders_Date	Quantity	Payment_Details	Status_of_Delivery	Rating	Comments	Member_ID	Food_ID	Manager_ID	Chef_ID
1	S0001	2022-02-01	2	Completed	Delivered	4	It is delicious	TP063290	1	M001	C001
2	S0002	2022-02-01	4	Completed	Delivered	5	Good Packaging	TP039487	3	M002	C002
3	S0003	2022-02-02	1	Completed	Delivered	3	Everything is good	TP028476	8	M003	C003
4	S0004	2022-02-03	2	Completed	Delivered	4	Nice and healthy	TP063290	5	M004	C004
5	S0005	2022-02-03	3	Completed	Out_of_Delivery	1	I have been waiting for a long time	TP092739	1	M005	C005
6	S0006	2022-02-04	3	Completed	Delivered	4	Tastes great as always	TP039487	3	M001	C001
7	S0007	2022-02-04	6	Completed	Order_Received	2	Takes too long time for preparing my order	TP092765	2	M002	C002
8	S0008	2022-02-04	7	Completed	Delivered	1	The Fried Chicken was too spicy	TP103863	6	M003	C003
9	S0009	2022-02-04	4	Completed	Delivered	5	Food standards and portion smaller than...	TP039487	7	M004	C004
10	S0010	2022-02-05	1	Completed	Delivered	4	The food was delicious but got one stran...	TP783614	4	M004	C004
11	S0011	2022-02-05	2	Completed	Order_Received	5	Well pacakaging, it is pro-environment	TP063290	3	M005	C005
12	S0012	2022-02-05	3	Completed	Order_Received	2	Why preparing time so long	TP028476	7	M005	C005
13	S0013	2022-02-05	1	Completed	Order_Received	3	NULL	TP092739	5	M001	C001

Figure 21: Output of Table Orders

SQL–Data Manipulation Language (DML)

Teo Jie Ing

```
select Food.Food_ID, Food.Food_Name, Max(Rating) as 'Highest Rating'
from Orders
inner join Food on Food.Food_ID = Orders.Food_ID
group by Food.Food_ID, Food_Name
having Max(Rating) = 5;
```

Figure 22: Query of Question 1

Question 1 requested that we display the highest rating in our food ordering system, as well as grouping by product id and food name. We used the max function to find the highest rating. Then, we utilized the inner join function to combine records from food and orders. The group by clause is then used to group the food id and food name that question 1 requested. Finally, we used the having clause to allow the group by search criteria to be applied. The below diagram shows the results of the Question 1.

	Food_ID	Food_Name	Highest Rating
1	3	Spaghetti	5
2	7	Smoked Salmon	5

Figure 23: Screenshot of Output (Food with Highest Rating)

```
select Member.Member_ID, Member.Member_Name, count(Feedback) as 'Total number of feedback'
from Member
group by Member.Member_ID, Member_Name
```

Figure 24: Query of Question 2

Question 2 asked us to calculate the total feedback per member and to display both the member id and the member's name. As a result, we must choose the title that corresponds to the query. Then we needed to write a from clause that specified which table contained these data. The member id, member name, and feedback were all created in the same table. As a result, we only need to utilise the group by clause to group the generated rows into sets to make it easier. The picture below shows that the results of the Question 2.

Results Messages			
	Member_ID	Member_Name	Total number of feedback
1	TP028476	Jessica	1
2	TP039487	Jackson Wang	1
3	TP063290	Chloe	1
4	TP092739	Oman Ishah	1
5	TP092765	Mohammad Ali	1
6	TP103863	Tiffany Tan	1
7	TP783614	Lavenda	1

Figure 25: Screenshot of Output (Total Number of Feedback)

```

select count(Quantity) as 'Number of meal cooked', Chef.Chef_ID, Chef.Chef_Name
from Orders
inner join Chef on Chef.Chef_ID = Orders.Chef_ID
group by Chef.Chef_ID, Chef.Chef_Name;

```

Figure 26: Query of Question 3

Question 3: We need to know how many meals each chef prepared. The final product must include the chef's ID, name, and number of meals made. As before, we must first select the title to be inserted in the table. Then we need to figure out which table the titles are from and utilise the inner join procedure. Finally, sort by the outcomes of Question 3 are shown in the graphic below.

Results Messages			
	Number of meal cooked	Chef_ID	Chef_Name
1	3	C001	Michael
2	2	C002	Bryant
3	2	C003	Durant
4	3	C004	Morant
5	3	C005	Kelvin

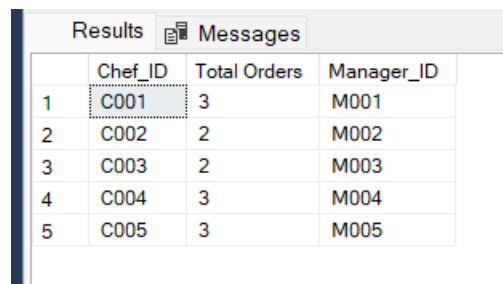
Figure 27: Screenshot of Output (Number of Meal Cooked)

Yap Yoong

```
Select Chef_ID, count(Orders_ID) as 'Total Orders', Manager_ID  
from Orders  
group by Chef_ID, Manager_ID;
```

Figure 28: Query of Question 4

Question 4 is to find out the total number of order that manager ordered form each chef. In this query we used COUNT function to find out the total number of orders, and group by Chef_ID and Manager_ID from table Orders.



	Chef_ID	Total Orders	Manager_ID
1	C001	3	M001
2	C002	2	M002
3	C003	2	M003
4	C004	3	M004
5	C005	3	M005

Figure 29: Screenshot of Output (Total Orders Ordered by Manager)

```
Select Orders.Food_ID, Food.Food_Name  
from Orders join Food  
on Orders.Food_ID = Food.Food_ID  
where Rating > (Select AVG(Rating) from Orders);
```

Figure 30: Query of Question 5

Question 5 is to find out the list of food where the food rating is more than the average rating of all food in table food. First, we used select command and join function to list out all the food and its rating. Then, we used WHERE command to distinguish the list of food that have more than average rating.



	Food_ID	Food_Name
1	1	Chicken Salad
2	3	Spaghetti
3	5	Chicken Sandwich
4	3	Spaghetti
5	7	Smoked Salmon
6	4	Mushroom Soup
7	3	Spaghetti

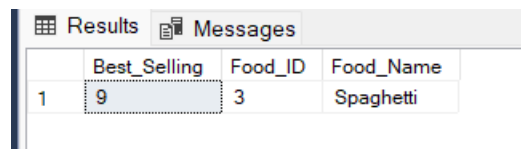
Figure 31: Screenshot of Output

Tiew Cheng Jia

```
Select top 1 sum(Quantity) as Best_Selling, Orders.Food_ID, Food.Food_Name
from Orders join Food
on Orders.Food_ID = Food.Food_ID
group by Orders.Food_ID, Food_Name
order by Best_Selling desc;
```

Figure 32: Query of Question 6

Question 6 need to find out the bestselling food in APU Café. We need to get the total quantity of food, the Food_ID and Food_Name. So, we need to sum up the quantity of food which named by Best_Selling, then group it by Food_ID and Food_Name after table Orders join table Food. Lastly, we select the top 1 from the output to get the bestselling food which is Spaghetti with a total of 9 sales.



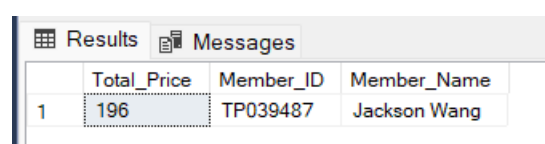
	Best_Selling	Food_ID	Food_Name
1	9	3	Spaghetti

Figure 33: Screenshot of Output (Best Selling)

```
Select top 1 sum(Unit_Price*Quantity) as Total_Price, Member.Member_ID, Member.Member_Name
from Orders join Member
on Orders.Member_ID = Member.Member_ID
join Food
on Orders.Food_ID = Food.Food_ID
group by Member.Member_ID, Member.Member_Name
order by Total_Price desc;
```

Figure 34: Query of Question 7

Question 7 need to find out the member(s) who spend most on ordering food. We select Unit_Price and Quantity from table Orders and Food, multiply them to get the order. Select Member_ID and Member_Name who made the order from table Member. Then we used JOIN command with these three tables. However, we only can get the total price made by a member, per order. So, we used SUM to get the Total_Price of each member spends in these days. Lastly, we order the table of output by descending order and select the top 1 which will show the result who spent most on ordering food.



	Total_Price	Member_ID	Member_Name
1	196	TP039487	Jackson Wang

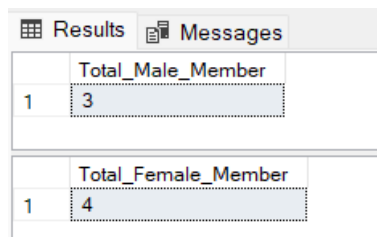
Figure 35: Screenshot of Output (Member Who Spent Most)

Yap Huei Ling

```
Select Count(Gender) AS Total_Male_Member from Member where Gender = 'Male';  
Select Count(Gender) As Total_Female_Member from Member where Gender = 'Female';
```

Figure 36: Query for Question 8

Question 8 need to show the total member based on gender who are registered as member. To create query 8 using DML, we use SELECT command and COUNT function. As created before, the table Member only include member who registered in APU Café Food Ordering System, so only need to select from the table Member. Figure 6 shows the output.



	Total_Male_Member
1	3

	Total_Female_Member
1	4

Figure 37: Screenshot of Output (Question 8)

```
Select  
Orders.Member_ID, Member.Member_Role, Member.Contact_Number,  
Orders.Food_ID, Food.Food_Name,  
Orders.Quantity, Orders.Orders_Date, Orders.Status_of_Delivery  
from Orders join Member  
on Orders.Member_ID = Member.Member_ID  
join Food  
on Orders.Food_ID = Food.Food_ID  
where Status_of_Delivery <> 'Delivered'  
order by Orders_Date;
```

Figure 38: Query of Question 9

In this query we used SELECT command, JOIN function with three tables, WHERE command and ORDER BY command. It joined three tables which are Orders, Member and Food. The reason of using ORDER BY command is to make the result table looks tidy. Figure 8 shows the output.



	Member_ID	Member_Role	Contact_Number	Food_ID	Food_Name	Quantity	Orders_Date	Status_of_Delivery
1	TP092739	Staff	0186730284	1	Chicken Salad	3	2022-02-03	Out_of_Delivery
2	TP092765	Student	0197829364	2	Beef Burger	6	2022-02-04	Order_Received
3	TP063290	Student	0126565435	3	Spaghetti	2	2022-02-05	Order_Received
4	TP028476	Staff	0178265692	7	Smoked Salmon	3	2022-02-05	Order_Received
5	TP092739	Staff	0186730284	5	Chicken Sandwich	1	2022-02-05	Order_Received

Figure 39: Screenshot of Output (Question 9)

```

Select count(*) as Total_Orders, Orders.Member_ID, Member.Member_Name
from Orders join Member
on Orders.Member_ID = Member.Member_ID
group by Orders.Member_ID, Member_Name
having count(*) >2;

```

Figure 40: Query of Question 10

In this query we used SELECT command, JOIN function, GROUP BY function and HAVING function. The HAVING function used in this query was to find the member who having more than 2 orders. Figure 10 shows the output.

Results Messages			
	Total_Orders	Member_ID	Member_Name
1	3	TP039487	Jackson Wang
2	3	TP063290	Chloe

Figure 41: Screenshot of Output (Total Orders)

Conclusion

In conclusion, to make the APU Café Food Ordering System we had to create the “Food,” “Orders,” “Member,” “Manager,” “Chef” table. First, we had to create a new database then we include a collate clause for case sensitive in the columns. Second, we start to create the table and put all the “ID” as primary key, then add other column and put them a difference data type such as varchar(X), integer and other. After finish creating the table, we started to insert our data into the table and make sure all the data are correct, after insert the data we had start the query that the question had given such as the total number of orders, the highest rating, best-selling food and other more. After all the things we had done we finally make it to the end.

Workload Breakdown

Student Name	Contribution	Signature
Teo Jie Ing	<ul style="list-style-type: none">- Database Diagram- DDL for Member- DML for Questions 1, 2, 3	
Yap Huei Ling	<ul style="list-style-type: none">- Entity Relationship Diagram- DDL for Orders- DML Questions 8, 9, 10	
Tiew Cheng Jia	<ul style="list-style-type: none">- Introduction- DDL for Chef & Manager- DML for Questions 6, 7	
Yap Yoong	<ul style="list-style-type: none">- DDL for Food- DML for Questions 4, 5- Conclusion	