# SCHOOL OF ENGINEERING AND TECHNOLOGY

**COURSEWORK FOR**

*BSC (HONS) IN COMPUTER SCIENCE*

**YEAR 1;** ACADEMIC SESSION APRIL 2022

**SEG1201:** DATABASE FUNDAMENTALS

**DEADLINE:** Week 14 – Monday, 11th July 2022, 8:30 am

| No. | Student Full Name | Student ID | Signature |
|---|---|---|---|
| **1** | Aisha Sofia Binti Najidi | 20065231 | |
| **2** | Emily Teng Jie Qi | 20054607 | |
| **3** | Justin Phang Sheng Xuan | 20066502 | |

**Instructions**

1. This final assessment contributes 50% to your final grade.
2. This four-member group assignment is primarily for Course Learning Outcome 2 - Implement a database design group project using appropriate tools such as Oracle SQL.
3. Each member of the group is given a specific role to play and is required to present his/her part of the work.
4. Note that if your group has less than four members, each member in the group will need to take up some extra tasks.

**IMPORTANT**

*The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.*

**Academic Honesty Acknowledgement**

"We *Aisha Sofia Binti Najidi*, *Emily Teng Jie Qi*, and *Justin Phang Sheng Xuan*, verify that this paper contains entirely our own work. We have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, we have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. We realize the penalties (refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme) for any kind of copying or collaboration on any assignment." , ***(15th June 2022)***

# Contents

# 1.0 Construct a case scenario

Launching in May 2018, Grab (2018) food delivery service, GrabFood, was created to connect people to food businesses. It allows customers to order food online without any minimum order requirements while having a GrabFood delivery partner to pick up the order from the restaurant to bring it to you. However, due to the pandemic in 2020, there was an increase of complaints from the customers. This is because restaurants were taking in too many orders at once, furthermore, delivery drivers were handling too many orders at once. To prevent this problem from exacerbating further, Grab has hired new database designers to control the influx of consumer data while ensuring consumer data remains consistent and up to date.

Grab has a relational database to keep track of orders and deliveries. Grab acts as a delivery service for many stores. Each customer may only have one Grab account. Each customer must register a Grab account with their Identification Card or Passport. Each store may only have one menu, and customers must order from that menu. Employees must be over 18 years old and have a up to date drivers' licence to make deliveries.

The customer shall not be entitled to cancel an order after it is confirmed, if you cancel an order after it has been confirmed, the customer remains liable to pay the applicable fees for the order in full regardless of whether the food has been prepared or not. If there are no employees available for delivery, customers order will be cancelled and fully reimbursed. After each order is made, customers will be notified of an estimated time of arrival for each delivery. To ensure customer receives their orders on time, delivery employees of Grab can only take up to 3 delivery orders at a time. Moreover, the stores can only take up to 20 orders at a time to ensure the food is ready for delivery on time. Customers may place orders from different stores in multiple orders.

Grab will save customer bank details after first time entering unless customer opts out, to ensure easy ordering process moving forward. To ensure that customers receive their orders successfully, payments made will be on-hold until the delivery is made, if delivery is late, customer will be compensated with a voucher for their next order.

Each voucher will be given as a code, and each code is valid within one month of redemption. Each customer can only order from stores within a 10km radius of their current location. Each customer will be notified of the tax prices in the payment screen prior to making their payment. Each customer can have different payment methods. Every alcohol related food or beverage items must be made by a customer over 21 years old. In the event an alcohol related food or beverage is ordered and consumed by a customer under 21 years of age, Grab is not liable for any illegal acts.

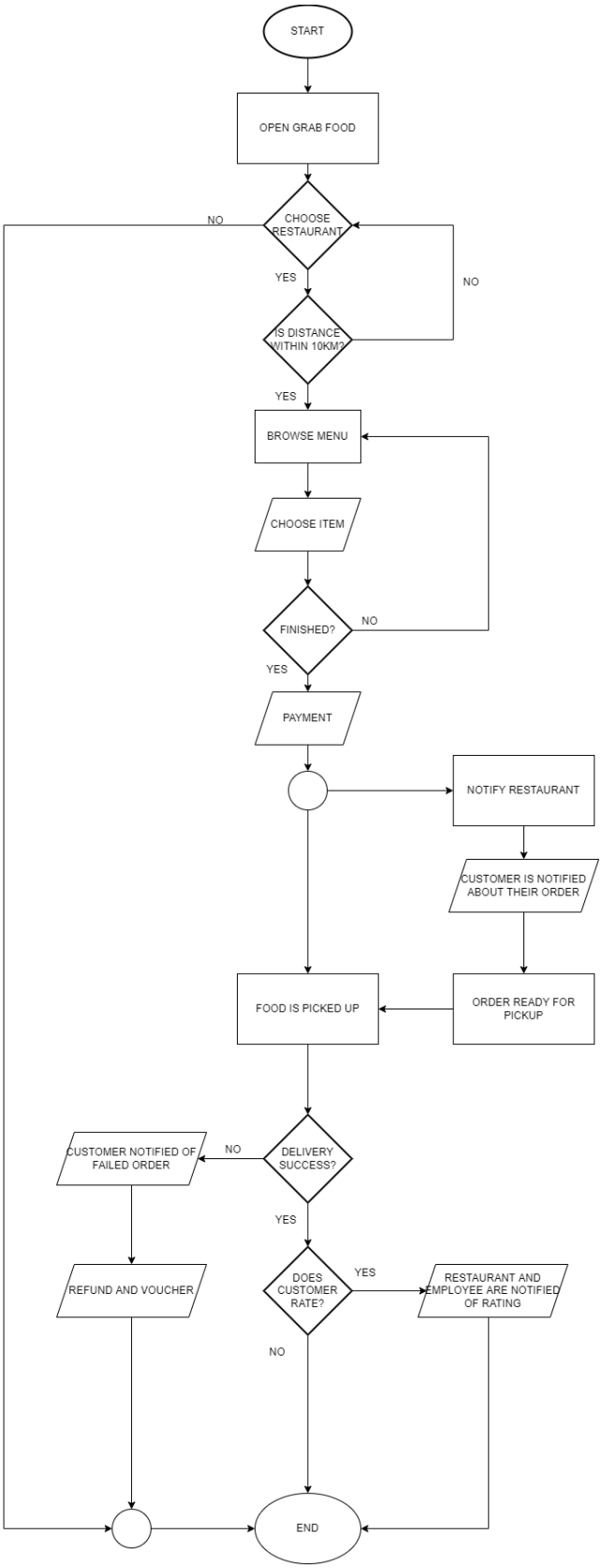
Each customer can rate the store and delivery person via the application after order has been delivered. GrabFood will halt services to stores that have a rating below 2.0 stars. Grab will temporarily halt deliveries made by employees with a rating below 1.5 stars. The tables are as follows Customer, Payment, Order, Order_line, Employee, Delivery, Store, Menu, Rating and Voucher.

## 1.1 Business Rules

1. Each customer can order multiple times, but a different OrderID will be provided to the system.
2. A unique OrderID will be provided for each order made by a customer.
3. Each store can take multiple orders
4. Each order has one or more order line.
5. Each Delivery is delivered by one employee.
6. One employee can make many deliveries.
7. Each delivery employee can take at most 3 orders at one time.
8. Each restaurant can take 20 orders at a time.
9. Each customer can order from one or more stores.
10. Each store only has one menu.
11. Each customer can only give one rating once per order.
12. Each delivery employee and store can be rated by many customers.
13. Each late delivery can only be compensated with one voucher.
14. Each voucher can only be redeemed once.

# 2.0 Design a database

## 2.1 FLOWCHART



## 2.2 RDM

Customer

(**Customer_ID**, First_Name, Last_Name, DOB, CLocation, Phone_Number)

Payment

(***Customer_ID, Order_ID***, Total)

Orders

(**Order_ID**, *Customer_ID*, Time_Order_Placed)

Order_line

(***Order_ID, Store_ID*,** Product, Product_ID, Quantity, Price)

Employee

(**Employee_ID**, EMP_First_Name, EMP_Last_Name, EMP_Phone_Number, Managed_By, Position, Salary)

Delivery

(**Delivery_ID, *Employee_ID, Order_ID***, ETA, Del_Price)

Store

**(Store_ID**, Store_Name, Location, Store_Location, AgeRes)

Rating

(***Customer_ID, Store_ID, Employee_ID,*** Store_Rating_Amount, Emp_Rating_Amount**)**

Voucher

(**Voucher_Code**, *Customer_ID***,** Exp_Date, Redeem_Date)

*Italic* = Foreign Key

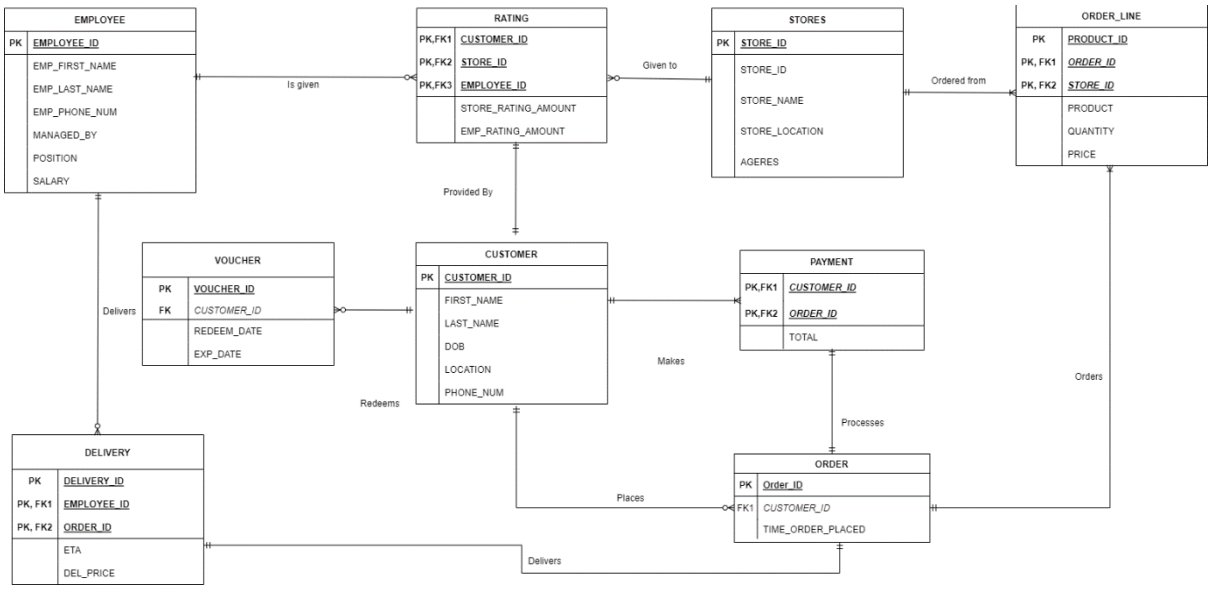
**Bold & Underline** = Primary Key

## 2.3 Attribute Description Table

| Attributes | Attribute Description | Data type | Justification |
|---|---|---|---|
| Customer_ID | ID of the customer | INT | Customer ID is in integers, so INT is used. |
| First_Name | First name of the customer | VARCHAR2(30) | Allows for customer name to be 30 letters long. |
| Last_Name | Last name of the customer | VARCHAR2(30) | Allows for customer name to be 30 letters long. |
| DOB | Date of Birth | DATE | DOB is in DATE format. |
| Location | Location of the customer | VARCHAR2(1000) | Allows for both numbers and letters in address, as well as allows long addresses. |
| Phone_Number | Customer phone number | VARCHAR2(15) | Allows multiple ways to write phone numbers. (E.g.: 60xxxxxxxxx, +60xxxxxxxxx,01X-XXX-XXXX, 01XXXXXXXX). |
| Order_ID | ID of the order | INT | The ORDER_IDs are numeric values (E.g.: 100000). |
| Total | Total price of the order | NUMBER(10,2) | The total price of the order is in numeric with decimal values. |
| Time_Order_Placed | The time when the order was placed | TIMESTAMP(2) | TIMESTAMP is used because it not only provides us with date, but it also provides us the time in HH:MM:SS. |
| Store_ID | ID of store | INT | Store_ID is in integers, so INT is used. |
| Product | Product information | VARCHAR2(100) | Allows for long name products to be inserted. |
| Quantity | Product quantity | NUMBER(3) | The quantity of the order can be a maximum of 999. |
| Price | Price per product | NUMBER (10,2) | The total price of the order is in numeric with decimal values. |
| Employee_ID | ID of the employee | NVARCHAR2(5) | To futureproof grab, NVARCHAR2 is used to allow Unicode characters to be used. This allows the ID to be kept short while still being unique. |

| | | | |
|---|---|---|---|
| **EMP_First_Name** | **Employee first name** | **VARCHAR2(30)** | **Allows for employee name to be 30 letters long.** |
| **EMP_Last_Name** | **Employee last name** | **VARCHAR2(30)** | **Allows for employee name to be 30 letters long.** |
| **EMP_Phone_Number** | **Employee phone number** | **VARCHAR2(15)** | **Allows multiple ways to write phone numbers. (Eg: 60xxxxxxxxx, +60xxxxxxxxx,01X-XXX-XXXX, 01XXXXXXXX).** |
| **Position** | **Employee position** | **VARCHAR2(20)** | **Allows some positions/titles to have only capitalized alphabets (exp: CEO)** |
| **Salary** | **Employee Salary** | **NUMBER(10,2)** | **The total salary of the employee can be in decimal format.** |
| **Managed By** | **Employee Managed by who** | **NVARCHAR2(5)** | **It must be in the same data type as the employee ID.** |
| **Emp_Rating_Amount** | **Employee rating amount** | **NUMBER(2,1)** | **The employee rating amount can be in decimal (Eg: 5.2).** |
| **Delivery_ID** | **Delivery ID** | **INT** | **Delivary_ID is in numeric values only therefore INT is used.** |
| **ETA** | **Estimated time of arrival** | **TIMESTAMP(2)** | **TIMESTAMP is used because it not only provides us with date but it also provides us the time in HH:MM:SS.** |
| **Delivery_Price** | **Price of the delivery** | **NUMBER(10,2)** | **The total price of the order is in numeric with decimal values.** |
| **STORE_Location** | **Location of the store** | **VARCHAR2(1000)** | **The location of the store can be in both alphabetical and numeric values.** |
| **Store_Rating_Amount** | **Store rating amount** | **NUMBER(2,1)** | **The store rating amount can be in decimal (Eg: 5.2).** |
| **Voucher_Code** | **Voucher code** | **CHAR(5)** | **The voucher codes are in alphabetic and numeric values (Eg: V0001).** |
| **Exp_Date** | **Expiry date** | **DATE** | **The Exp_Date attribute is in a date format.** |
| **Redeem_Date** | **Redeem date** | **DATE** | **The Redeem_Date attribute is in a date format.** |

## 2.4 ERD


EMPLOYEE
PK EMPLOYEE_ID
EMP_FIRST_NAME
EMP_LAST_NAME
EMP_PHONE_NUM
MANAGED_BY
POSITION
SALARY
Is given
RATING
PK,FK1 CUSTOMER_ID
PK,FK2 STORE_ID
PK,FK3 EMPLOYEE_ID
STORE_RATING_AMOUNT
EMP_RATING_AMOUNT
Given to
STORES
PK STORE_ID
STORE_ID
STORE_NAME
STORE_LOCATION
AGERES
Ordered from
ORDER_LINE
PK PRODUCT_ID
PK, FK1 ORDER_ID
PK, FK2 STORE_ID
PRODUCT
QUANTITY
PRICE
Provided By
VOUCHER
PK VOUCHER_ID
FK CUSTOMER_ID
REDEEM_DATE
EXP_DATE
Delivers
CUSTOMER
PK CUSTOMER_ID
FIRST_NAME
LAST_NAME
DOB
LOCATION
PHONE_NUM
PAYMENT
PK,FK1 CUSTOMER_ID
PK,FK2 ORDER_ID
TOTAL
Makes
Orders
Redeems
Processes
DELIVERY
PK DELIVERY_ID
PK, FK1 EMPLOYEE_ID
PK, FK2 ORDER_ID
ETA
DEL_PRICE
Places
ORDER
PK Order_ID
FK1 CUSTOMER_ID
TIME_ORDER_PLACED
Delivers

# 3.0 Implement a database

## 3.1 User Views

**Create View For Customer Age**

CREATE VIEW customer_age AS

SELECT CUSTOMER_ID AS ID, FIRST_NAME, LAST_NAME, ((EXTRACT (YEAR FROM CURRENT_DATE)) - (EXTRACT (YEAR FROM DOB))) AS Age

FROM Customer

**Results:**

```
SELECT *
FROM customer_age
```

Results | Explain | Describe | Saved SQL | History

| ID | FIRST_NAME | LAST_NAME | AGE |
|---|---|---|---|
| 111 | Ashley | Lee | 23 |
| 123 | Jason | Yu | 20 |
| 189 | Muhammad | Arif | 22 |
| 222 | John | Doe | 19 |
| 235 | Ashley | Lee | 32 |
| 333 | Muthu | Segaran | 28 |
| 432 | Ashley | McKinsey | 22 |
| 444 | Alex | Tan | 32 |
| 456 | Aly | Kew | 23 |
| 457 | Justin | Yu | 66 |
| 478 | Muhammad | Aqeel | 32 |
| 516 | Ali | Ahsan | 12 |
| 666 | Jody | Yu | 51 |
| 777 | Daniel | Amin | 32 |
| 790 | Muhammad | Areef | 22 |
| 883 | Muhammad | Zahar | 33 |
| 886 | Laura | McKinsey | 22 |
| 888 | Marcus | Tan | 14 |
| 1001 | Nur | Aisha | 14 |
| 1002 | Nur | Salma | 16 |

**Create View For Employee Average Rating**

CREATE VIEW employee_average_ratings AS

SELECT Employee.EMPLOYEE_ID AS Employee, ROUND((AVG(RATING.EMP_RATING_AMOUNT)),1) AS Rating_Amount

FROM Employee, Rating

WHERE EMPLOYEE.EMPLOYEE_ID = RATING.EMPLOYEE_ID

GROUP BY EMPLOYEE.EMPLOYEE_ID

**Results:**

```
SELECT *
FROM employee_average_ratings
```

Results | Explain | Describe | Saved SQL | History

| EMPLOYEE | RATING_AMOUNT |
|---|---|
| A48 | 4.5 |
| A10 | 4.5 |
| A17 | 3.3 |
| A49 | 5 |
| A01 | 3.7 |
| A02 | 3 |
| A12 | 3.5 |

7 rows returned in 0.02 seconds Download

**<u>Create View For Store Average Rating</u>**

CREATE VIEW store_average_ratings AS

SELECT STORES.STORE_ID AS ID, ROUND((AVG (RATING.STORE_RATING_AMOUNT)),1) AS Rating_Amount

FROM Stores, Rating

WHERE Stores.Store_ID = Rating.Store_ID

GROUP BY Stores.Store_ID

**Results:**

```
1  SELECT *
2  FROM store_average_ratings
```

Results | Explain | Describe | Saved SQL | History

| ID | RATING_AMOUNT |
|---|---|
| 6 | 3 |
| 14 | 2 |
| 1 | 3.3 |
| 11 | 2 |
| 12 | 4 |
| 17 | 3.5 |
| 3 | 2 |
| 16 | 4 |

## 3.2 Indexing

CREATE INDEX CUSTOMER_AGE on CUSTOMER (CUSTOMER_ID, FIRST_NAME, LAST_NAME, DOB);

CREATE INDEX STORE_RATING on STORES (STORE_ID, STORE_NAME, AGERES);

CREATE INDEX EMPLOYEE_RATING on EMPLOYEE (EMPLOYEE_ID, EMP_FIRST_NAME, EMP_LAST_NAME);

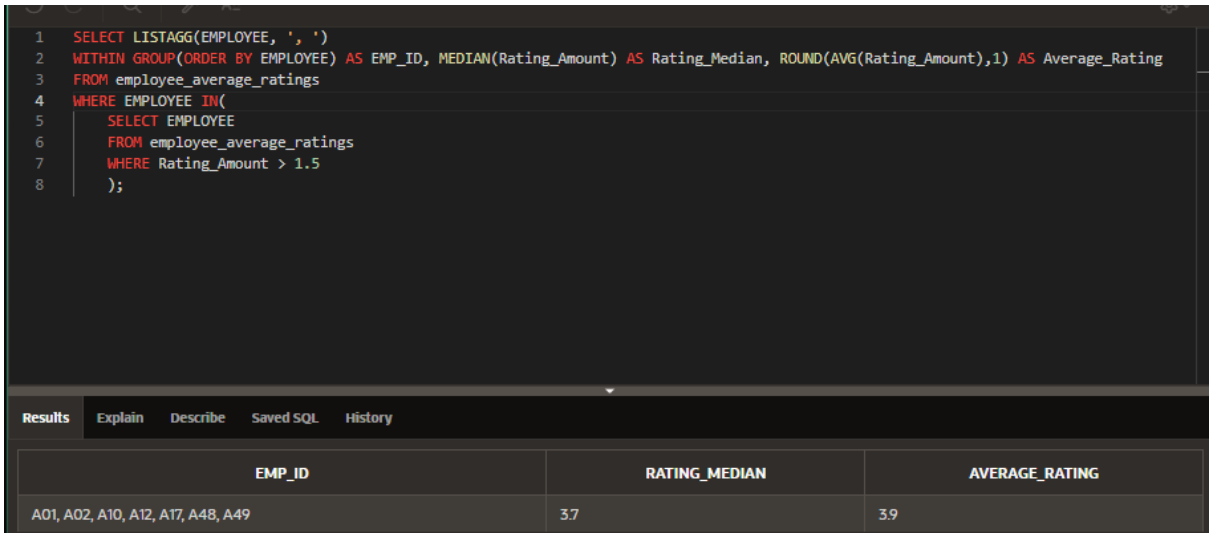
# 4.0 Query a database

a. Write a user query with one sub-query, and two new aggregate functions which have not been taught in this subject. Please explore the functions from the ORACLE website

- Grab wishes to analyse employee's rating during the pandemic for their year-end report, so the statistics team from grab food wishes to see the delivery employees and the discrepancy between the median rating, and average rating.
- **CODE:**

```
SELECT LISTAGG(EMPLOYEE, ', ')
WITHIN GROUP(ORDER BY EMPLOYEE) AS EMP_ID,
MEDIAN(Rating_Amount) AS Rating_Median,
ROUND(AVG(Rating_Amount),1) AS Average_Rating
FROM employee_average_ratings
WHERE EMPLOYEE IN(
   SELECT EMPLOYEE
   FROM employee_average_ratings
   WHERE Rating_Amount > 1.5
   );
```

- **RESULTS:**



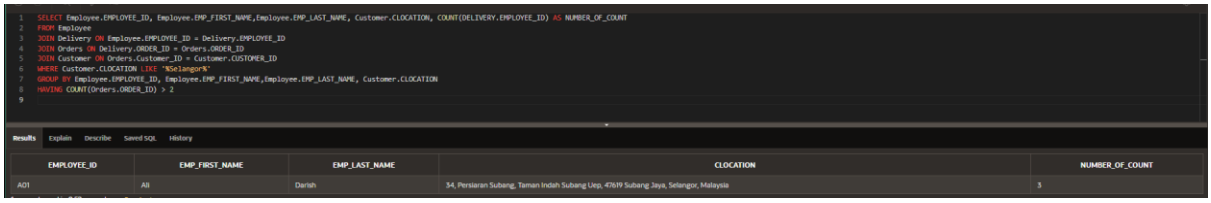

| EMP_ID | RATING_MEDIAN | AVERAGE_RATING |
|---|---|---|
| A01, A02, A10, A12, A17, A48, A49 | 3.7 | 3.9 |

b. Write a user query with a 4-table join, 2 user conditions (note: and user conditions are not the same) and a GROUP BY clause and HAVING subclause. Ensure that your scenario reflects the reason for such join, and not join those tables for the sake of joining.

- Grab wishes to find the most active employee in the Selangor area to give them a reward. As such, GrabFood has given the database designers a task to find a list of the most active employees in terms of delivery in Selangor.
- **CODE:**

```
SELECT Employee.EMPLOYEE_ID,
Employee.EMP_FIRST_NAME,Employee.EMP_LAST_NAME,
Customer.CLOCATION,
COUNT(DELIVERY.EMPLOYEE_ID) AS
NUMBER_OF_COUNT
FROM Employee
JOIN Delivery ON Employee.EMPLOYEE_ID =
Delivery.EMPLOYEE_ID
JOIN Orders ON Delivery.ORDER_ID = Orders.ORDER_ID
JOIN Customer ON Orders.Customer_ID =
Customer.CUSTOMER_ID
WHERE Customer.CLOCATION LIKE '%Selangor%'
GROUP BY Employee.EMPLOYEE_ID,
Employee.EMP_FIRST_NAME,Employee.EMP_LAST_NAME,
Customer.CLOCATION
HAVING COUNT(Orders.ORDER_ID) > 2
```

- **RESULTS:**



```
SELECT Employee.EMPLOYEE_ID, Employee.EMP_FIRST_NAME,Employee.EMP_LAST_NAME, Customer.CLOCATION, COUNT(DELIVERY.EMPLOYEE_ID) AS NUMBER_OF_COUNT
FROM Employee
JOIN Delivery ON Employee.EMPLOYEE_ID = Delivery.EMPLOYEE_ID
JOIN Orders ON Delivery.ORDER_ID = Orders.ORDER_ID
JOIN Customer ON Orders.Customer_ID = Customer.CUSTOMER_ID
WHERE Customer.CLOCATION LIKE '%Selangor%'
GROUP BY Employee.EMPLOYEE_ID, Employee.EMP_FIRST_NAME,Employee.EMP_LAST_NAME, Customer.CLOCATION
HAVING COUNT(Orders.ORDER_ID) > 2
```

Results | Explain | Describe | Saved SQL | History

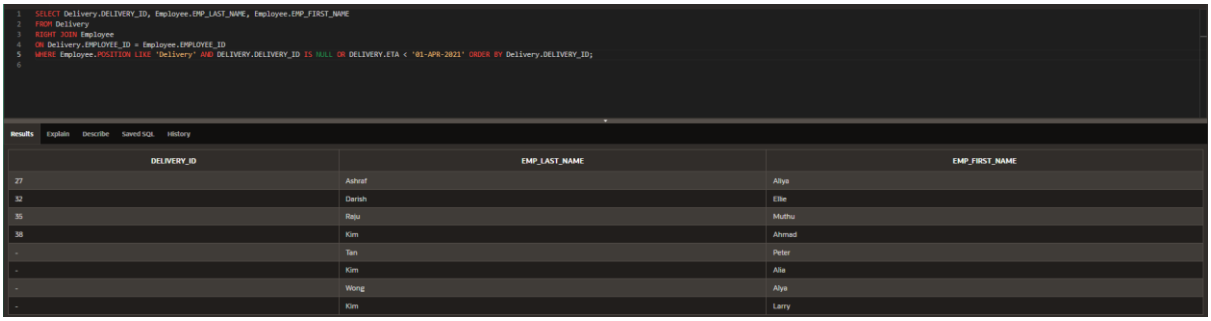
| EMPLOYEE_ID | EMP_FIRST_NAME | EMP_LAST_NAME | CLOCATION | NUMBER_OF_COUNT |
|---|---|---|---|---|
| A01 | Ali | Darish | 34, Persiaran Subang, Taman Indah Subang Uep, 47619 Subang Jaya, Selangor, Malaysia | 3 |

c. Write a user query with a right outer join and 3 user conditions. One of the conditions uses the LIKE keyword.

- Grab wishes to find employees that have not done a delivery, or any employees that have recently (In the past 3 months) not done any deliveries in order to take disciplinary actions towards them.
- **CODE:**

```
SELECT Delivery.DELIVERY_ID, Employee.EMP_LAST_NAME,
Employee.EMP_FIRST_NAME FROM Delivery RIGHT JOIN Employee
ON Delivery.EMPLOYEE_ID = Employee.EMPLOYEE_ID WHERE
Employee.POSITION LIKE 'Delivery' AND DELIVERY.DELIVERY_ID
IS NULL OR DELIVERY.ETA < '01-APR-2021' ORDER BY
Delivery.DELIVERY_ID;
```

- **RESULTS:**



```
SELECT Delivery.DELIVERY_ID, Employee.EMP_LAST_NAME, Employee.EMP_FIRST_NAME
FROM Delivery
RIGHT JOIN Employee
ON Delivery.EMPLOYEE_ID = Employee.EMPLOYEE_ID
WHERE Employee.POSITION LIKE 'Delivery' AND DELIVERY.DELIVERY_ID IS NULL OR DELIVERY.ETA < '01-APR-2021' ORDER BY Delivery.DELIVERY_ID;
```

Results | Explain | Describe | Saved SQL | History

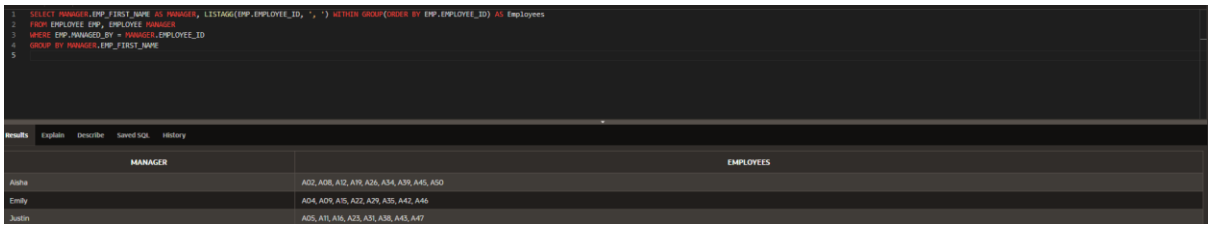
| DELIVERY_ID | EMP_LAST_NAME | EMP_FIRST_NAME |
|---|---|---|
| 27 | Ashraf | Aliya |
| 32 | Darish | Ellie |
| 35 | Raju | Muthu |
| 38 | Kim | Ahmad |
| - | Tan | Peter |
| - | Kim | Alia |
| - | Wong | Alya |
| - | Kim | Larry |

d. Write a user query using a unary join. Ensure that your scenario reflects the reason for such requirement. Display only the relevant records and fields.

- Grab wishes to find the managers of each employee, to easily find out which manager to contact in the situation where disciplinary action needs to be taken.
- **CODE:**

```
SELECT MANAGER.EMP_FIRST_NAME AS MANAGER,
LISTAGG(EMP.EMPLOYEE_ID, ', ') WITHIN GROUP(ORDER BY
EMP.EMPLOYEE_ID) AS Employees
FROM EMPLOYEE EMP, EMPLOYEE MANAGER
WHERE EMP.MANAGED_BY = MANAGER.EMPLOYEE_ID
GROUP BY MANAGER.EMP_FIRST_NAME
```

- **RESULTS:**



```
SELECT MANAGER.EMP_FIRST_NAME AS MANAGER, LISTAGG(EMP.EMPLOYEE_ID, ', ') WITHIN GROUP(ORDER BY EMP.EMPLOYEE_ID) AS Employees
FROM EMPLOYEE EMP, EMPLOYEE MANAGER
WHERE EMP.MANAGED_BY = MANAGER.EMPLOYEE_ID
GROUP BY MANAGER.EMP_FIRST_NAME
;
```

Results | Explain | Describe | Saved SQL | History

| MANAGER | EMPLOYEES |
|---|---|
| Aisha | A02, A08, A12, A19, A26, A34, A39, A45, A50 |
| Emily | A04, A09, A15, A22, A29, A35, A42, A46 |
| Justin | A05, A11, A16, A23, A31, A38, A43, A47 |

# 5.0 Teamwork and Presentation

## 5.1 Members and Roles

| No. | Student Full Name | Student ID | Roles | Signature |
|---|---|---|---|---|
| **1** | Aisha Sofia Binti Najidi | 20065231 | **PROJECT MANAGER, CEO** | |
| **2** | Emily Teng Jie Qi | 20054607 | **DATABASE DESIGNER** | |
| **3** | Justin Phang Sheng Xuan | 20066502 | **DATABASE TEAM LEADER** | |

## 5.2 Project Plan

| TASK | ASSIGNMENT | START | END | PROGRESS |
|---|---|---|---|---|
| • Constructing case scenario<br>• List Business requirements<br>• Create Business rules<br>• Record presentation | All Members<br>All Members<br>Emily, Justin<br>Aisha | 10/06/22 | 15/06/22 | 100% |
| • Flow Chart Diagram<br>• ERD<br>• RDM<br>• Progress Report | Justin<br>Justin<br>Emily<br>Aisha | 18/06/22 | 22/06/22 | 100% |
| • Create Script<br>• Create Views and Index<br>• Create Check constraints<br>• Populate database<br>• Progress Report | Aisha<br>Justin<br>Emily<br>Justin<br>Aisha | 25/06/22 | 29/06/22 | 100% |
| • User queries<br>• SQL statements and results<br>• Project progress report | Aisha<br>Emily, Justin<br>Aisha | 02/07/22 | 06/07/22 | 100% |
| • Presentation | All Members | 12/07/2022 | 12/07/2022 | - |

# References

Grab. (2021, August 15). *Grab launches grabfood beta and unveils vision to be an everyday app*. Grab SG. Retrieved June 15, 2022, from https://www.grab.com/sg/press/business/grab-launches-grabfood-beta-and-unveils-vision-to-be-an-everyday-app/

Grab. (2019, October 25). *Terms of Service: Transport, Delivery and Logistics*. Grab MY. https://www.grab.com/my/terms-policies/transport-delivery-logistics/