TIS1101 Database Fundamentals

Assignment 2

Title: FoodPanda Food Delivery

Prepared by:

Group	Name	ID	Email
Leader	Muhammad Uzair Bin Abdul Razak	1191303163	1191303163@student.mmu.edu.my
Member	Iman Aisyah binti Zailani	1191302815	1191302815@student.mmu.edu.my
Member	Kishen Kumar A/L Sivalingam	1191101423	1191101423@student.mmu.edu.my

1.0 Updated ERD

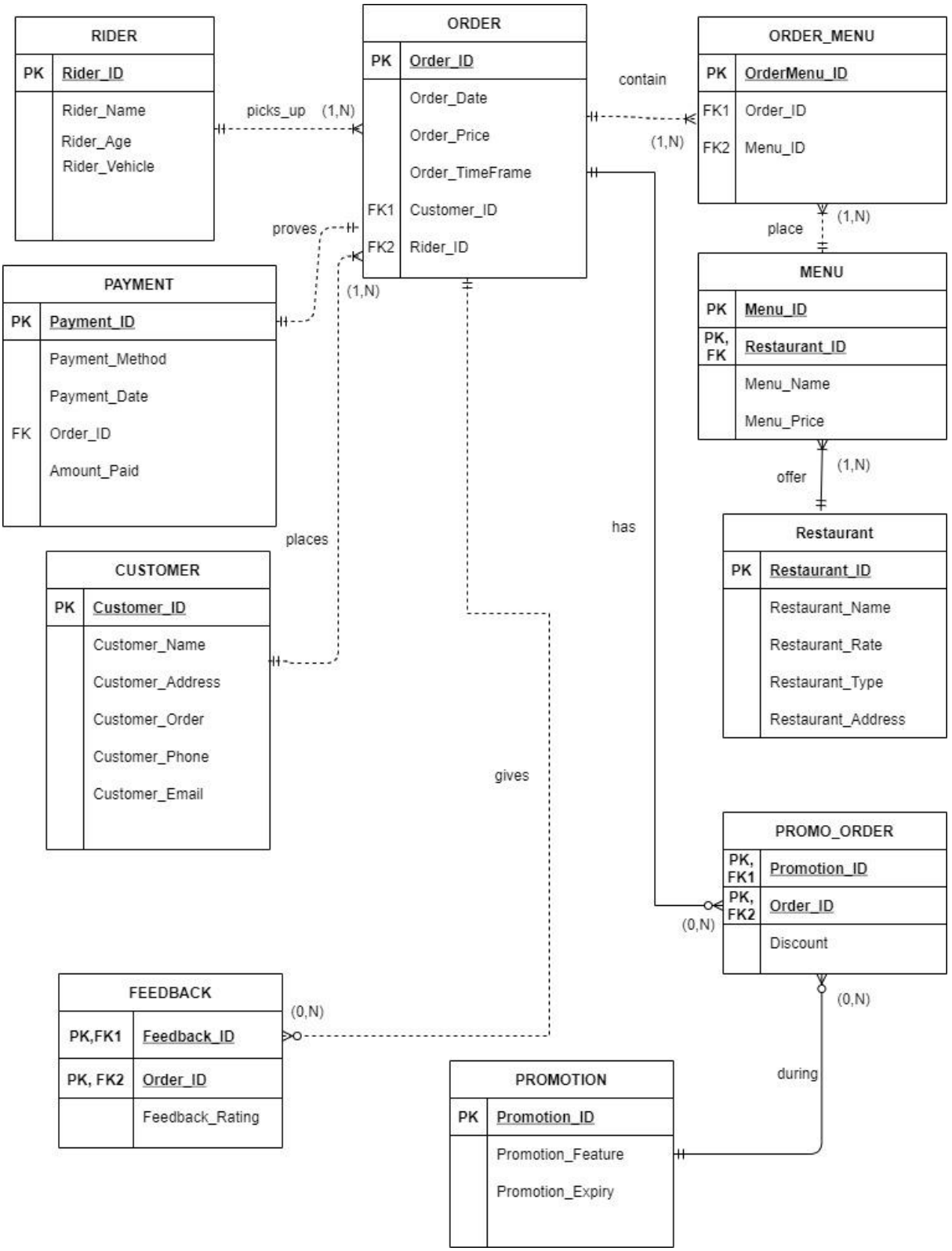


Figure 1.0: Updated Entity Relation Diagram

2.0 Data Dictionary

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK/FK	FK REFERENCED TABLE
RIDER	RIDER_ID	Rider ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	RIDER_NAME	Rider name	VARCHAR2(30)	Xxxxxxx		Y		
	RIDER_AGE	Rider age	NUMBER(5)	99	18-99	Y		
	RIDER_VEHICLE	Rider vehicle type	VARCHAR2(10)	Xxxxxxx				
ORDER	ORDER_ID	Order ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	ORDER_DATE	Date of the order	DATE	dd-mm-yyyy		Y		
	ORDER_PRICE	Price of the order	NUMBER(7,2)	99,999.99		Y		
	ORDER_TIMEFRAME	Time frame of the order	CHAR(10)	Xxxxxxx				
	CUSTOMER_ID	Customer ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	CUSTOMER
	Rider_ID	Rider ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	RIDER
ORDER_MENU	ORDERMENU_ID	Order Menu ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	ORDER_ID	Order ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	ORDER
	MENU_ID	Menu ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	MENU
MENU	MENU_ID	Menu ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	RESTAURANT_ID	Restaurant ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	RESTAURANT
	MENU_NAME	Menu name	VARCHAR2(30)	Xxxxxxx				
	MENU_PRICE	Menu price	NUMBER(7,2)	99,999.99				
RESTAURANT	RESTAURANT_ID	Restaurant ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	RESTAURANT_NAME	Restaurant name	VARCHAR2(30)	Xxxxxxx		Y		
	RESTAURANT_RATE	Restaurant rate	NUMBER(5,1)	9.9	0.0-5.0			
	RESTAURANT_TYPE	Type of the restaurant	VARCHAR2(20)	Xxxxxxx				
	RESTAURANT_ADDRESS	Restaurant address	VARCHAR(50)	Xxxxxxx				
PROMO_ORDER	PROMOTION_ID	Promotion ID number	VARCHAR2(10)	Xxxxxxx		Y	PK,FK	PROMOTION
	ORDER_ID	Order ID number	VARCHAR2(10)	Xxxxxxx		Y	PK,FK	ORDER
	DISCOUNT	Discount	NUMBER(5,2)	9.99				
PROMOTION	PROMOTION_ID	Promotion ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	PROMOTION_FEATURE	Details of promotion	VARCHAR2(100)	Xxxxxxx				
	PROMOTION_EXPIRY	Expiry date of promotion	DATE	dd-mm-yyyy		Y		
FEEDBACK	FEEDBACK_ID	Feedback ID number	NUMBER(5)	99	100-999	Y	PK	
	ORDER_ID	Order ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	ORDER
	FEEDBACK_RATING	Rating from the feedback	NUMBER(5,1)	9.9	0.0-5.0	Y		
CUSTOMER	CUSTOMER_ID	Customer ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	CUSTOMER_NAME	Customer name	VARCHAR2(30)	Xxxxxxx		Y		
	CUSTOMER_ADDRESS	Customer address	VARCHAR2(50)	Xxxxxxx		Y		
	CUSTOMER_PHONE	Customer phone number	NUMBER(30)			Y		
	CUSTOMER_EMAIL	Customer email	VARCHAR2(30)	Xxxxxxx				
PAYMENT	PAYMENT_ID	Payment ID number	VARCHAR2(10)	Xxxxxxx		Y	PK	
	PAYMENT_METHOD	Method of payment	VARCHAR2(20)	Xxxxxxx				
	PAYMENT_DATE	Date of payment	DATE	dd-mm-yyyy				
	ORDER_ID	Order ID number	VARCHAR2(10)	Xxxxxxx		Y	FK	ORDER
	AMOUNT_PAID	Amount paid from customer	NUMBER(7,2)	99,999.99				

Figure 2.0: Data Dictionary

3.0 Creation of Tables

1. Customer Table

```
CREATE TABLE Customer
(
Customer_ID varchar2(10) NOT NULL,
Customer_Name varchar2(30),
Customer_Address varchar2(50),
Customer_Order varchar2(100),
Customer_Phone number(30),
Customer_Email varchar2(30),
PRIMARY KEY (Customer_ID)
);
```

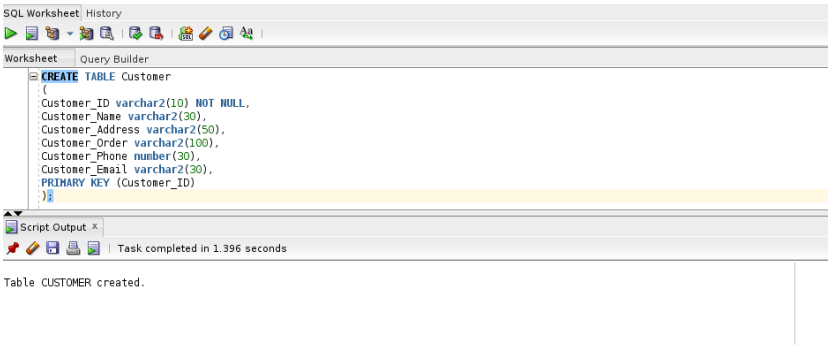


Figure 3.1: Prove of Customer table created

2. Rider Table

```
Create Table Rider
(
    Rider_ID VARCHAR2(10) NOT NULL,
    Rider_Name VARCHAR2(30),
    Rider_Age NUMBER(5),
    Rider_Vehicle VARCHAR2(10),
    PRIMARY KEY (Rider_ID)
);
```

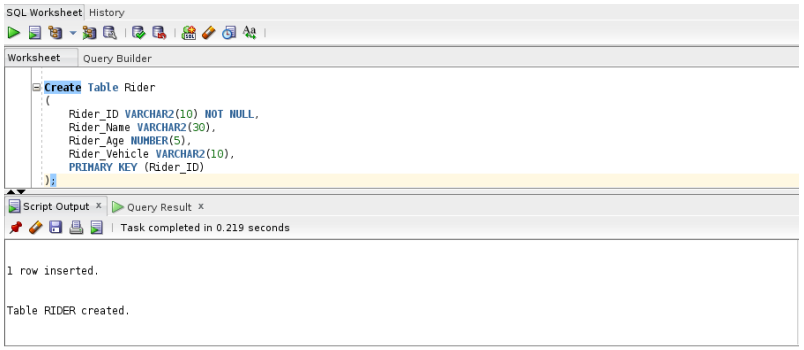


Figure 3.2: Prove of Rider table created

3. Restaurant Table

```
CREATE TABLE Restaurant
(
    Restaurant_ID VARCHAR2(10) NOT NULL,
    Restaurant_Name VARCHAR2(30),
    Restaurant_Rate NUMBER(5,1) CHECK (Restaurant_Rate BETWEEN
0.0 AND 5.0),
    Restaurant_Type VARCHAR2(20),
    Restaurant_Address VARCHAR(50),
    PRIMARY KEY (Restaurant_ID)
);
```

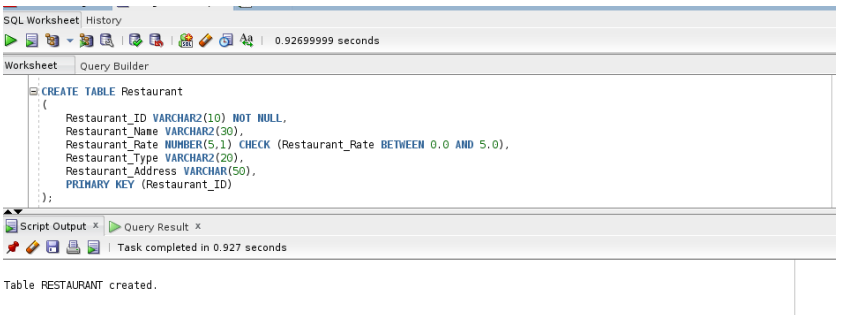


Figure 3.3: Prove of Restaurant table created

4. Order Table

```
Create Table Ordr
(
   Order_ID VARCHAR2(10) PRIMARY KEY NOT NULL,
   Order_Date DATE,
   Order_Price NUMBER(7,2),
   Order_Timeframe CHAR(10),
   Customer_ID VARCHAR2(10),
   Rider_ID VARCHAR2(10),
   FOREIGN KEY (Customer_ID) REFERENCES Customer,
   FOREIGN KEY (Rider_ID) REFERENCES Rider
);
```

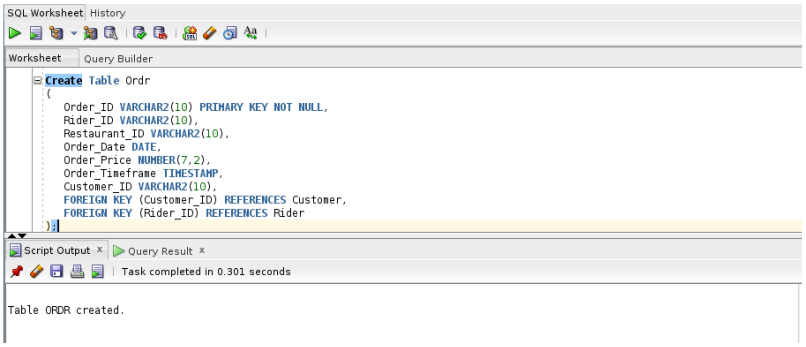


Figure 3.4: Prove of Order table created

5. Menu Table

```
CREATE TABLE Menu
(
  Menu_ID VARCHAR2(10) PRIMARY KEY NOT NULL,
  Restaurant_ID VARCHAR2(10),
  Menu_Name VARCHAR2(30),
  Menu_Price NUMBER(7,2),
  FOREIGN KEY (Restaurant_ID) REFERENCES Restaurant
);
```

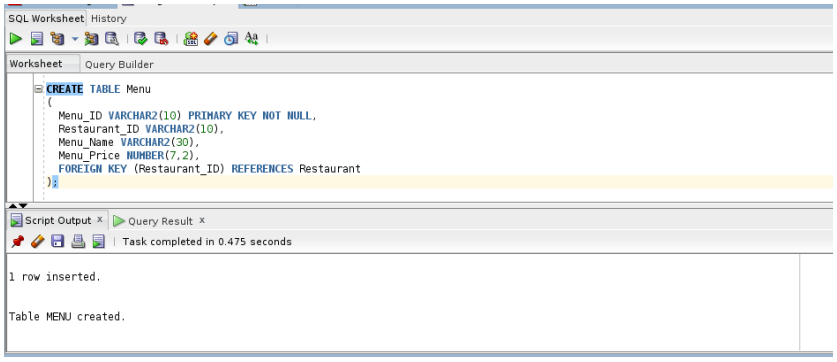


Figure 3.5: Prove of Menu table created

6. Order Menu Table

```
CREATE TABLE Order_Menu (
   OrderMenu_ID VARCHAR2(10) PRIMARY KEY NOT NULL,
   Order_ID VARCHAR2(10),
   Menu_ID VARCHAR2(10),
   FOREIGN KEY (Order_ID) REFERENCES Ordr,
   FOREIGN KEY (Menu_ID) REFERENCES Menu
);
```

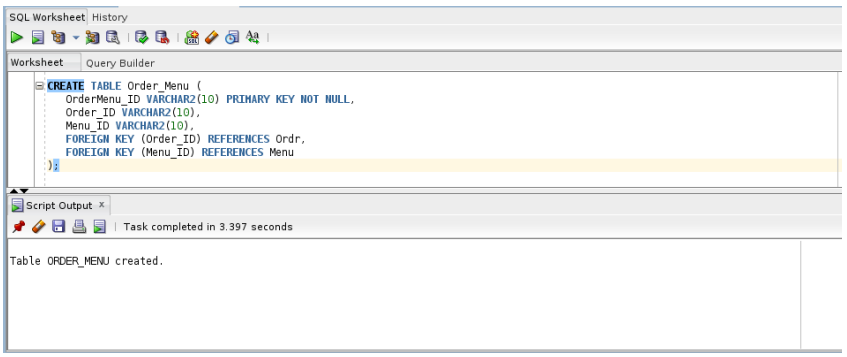


Figure 3.6: Prove of Order Menu table created

7. Promotion Table

```
CREATE TABLE Promotion
(
    Promotion_ID VARCHAR2(10) PRIMARY KEY NOT NULL,
    Promotion_Feature VARCHAR2(100),
    Promotion_Expiry DATE
);
```

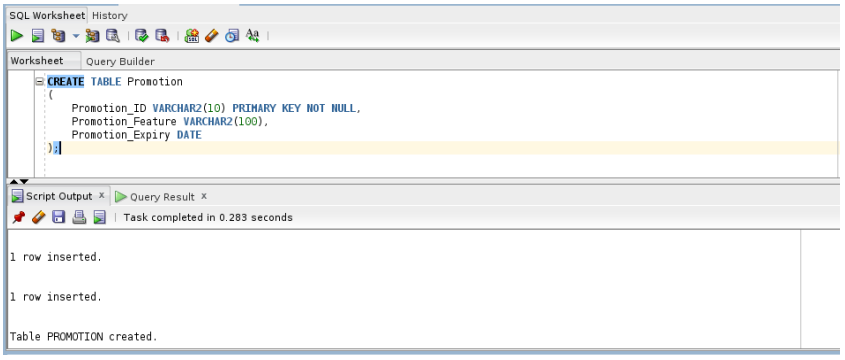


Figure 3.7: Prove of Promotion table created

8. Promo_Order Table

```
CREATE TABLE Promo_Order(
   Promotion_ID VARCHAR2(10) NOT NULL,
   Order_ID VARCHAR2(10) NOT NULL,
   Discount NUMBER(5,2),
   PRIMARY KEY(Promotion_ID,Order_ID),
   FOREIGN KEY (Promotion_ID) REFERENCES Promotion,
   FOREIGN KEY (Order_ID) REFERENCES Ordr
);
```

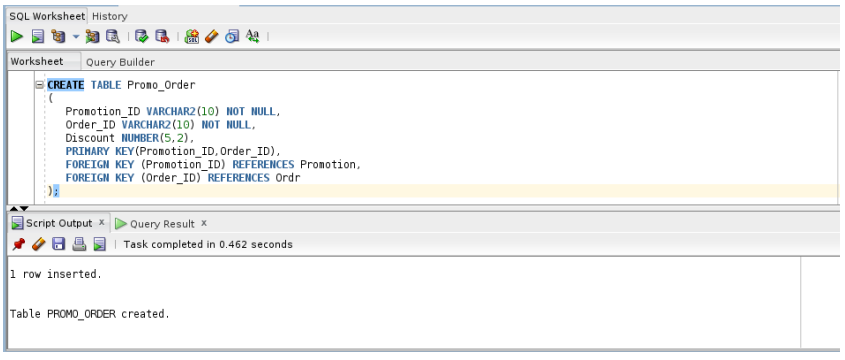


Figure 3.8: Prove of Promo Order table created

9. Payment Table

```
CREATE TABLE Payment
(
  Payment_ID VARCHAR2(10) PRIMARY KEY NOT NULL,
  Payment_Method VARCHAR2(20),
  Payment_Date DATE,
  Order_ID VARCHAR2(10),
  Amount_Paid NUMBER(7,2),
  FOREIGN KEY (Order_ID) REFERENCES Ordr
);
```

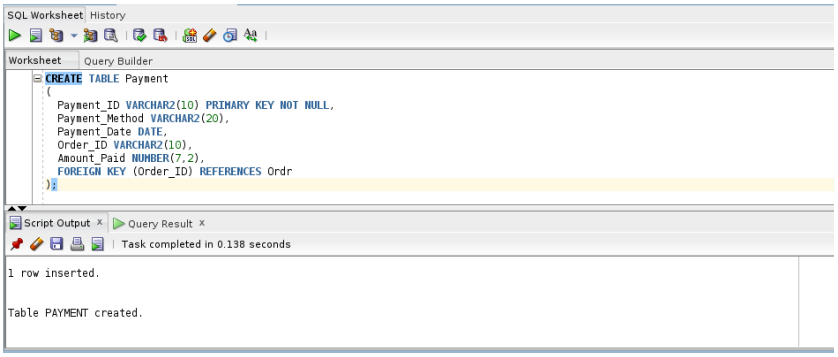


Figure 3.9: Prove of Payment table created

10. Feedback Table

```
CREATE TABLE Feedback
(
  Feedback_ID number(5) PRIMARY KEY NOT NULL,
  Feedback_Rating NUMBER(5,1) CHECK (Feedback_Rating BETWEEN 0.0
AND 5.0),
  Order_ID VARCHAR2(10),
  FOREIGN KEY (Order_ID) REFERENCES Ordr
);
```

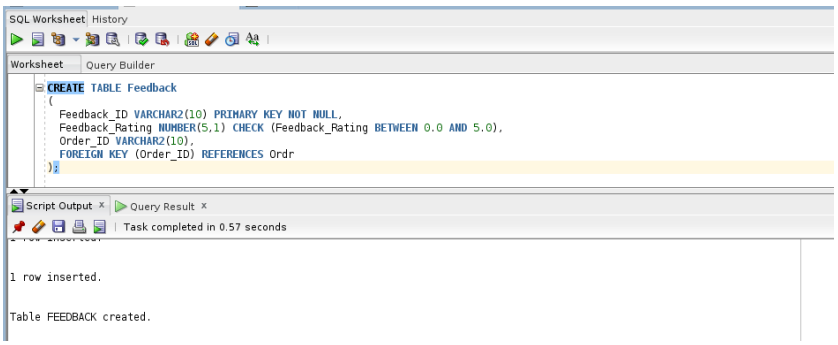


Figure 3.10: Prove of Feedback table created

4.0 Data Insertion

1. Customer Table

```
INSERT INTO Customer values ('CS-0001', 'Sally', '15, Rose Park, Shah Alam', 'cust order', 60197630287, 'sally123@gmail.com');

INSERT INTO Customer values ('CS-0002', 'Noah', 'Lot 7, Taman Mahkota, Subang Jaya', 'cust order', 60172397183, 'n0ah_3@gmail.com');

INSERT INTO Customer values ('CS-0003', 'Ron', '9, Flat A, Presint 9, Putrajaya', 'cust order', 60106501230, 'ronw3asley@gmail.com');

INSERT INTO Customer values ('CS-0004', 'Luna', 'No 01-19, Jalan Pertama, Cheras', 'cust order', 60137650189, 'lovegood30@gmail.com');

INSERT INTO Customer values ('CS-0005', 'Ginny', '05, Seksyen 23, Damansara', 'cust order', 60123458970, 'g_ny650@gmail.com');

SELECT * from Customer;
```

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CUSTOMER_ORDER	CUSTOMER_PHONE	CUSTOMER_EMAIL
CS-0001	Sally	15, Rose Park, Shah Alam	cust order	60197630287	sally123@gmail.com
CS-0002	Noah	Lot 7, Taman Mahkota, Subang Jaya	cust order	60172397183	n0ah_3@gmail.com
CS-0003	Ron	9, Flat A, Presint 9, Putrajaya	cust order	60106501230	ronw3asley@gmail.com
CS-0004	Luna	No 01-19, Jalan Pertama, Cheras	cust order	60137650189	lovegood30@gmail.com
CS-0005	Ginny	05, Seksyen 23, Damansara	cust order	60123458970	g_ny650@gmail.com

Figure 4.1: Customer table after data insertion

2. Rider Table

```
INSERT INTO Rider values('FP0001', 'Lisa', 25, 'Car');
INSERT INTO Rider values('FP0002', 'Ming', 20, 'Motorbike');
INSERT INTO Rider values('FP0003', 'Yusof', 30, 'Car');
INSERT INTO Rider values('FP0004', 'Kumar', 23, 'Car');
INSERT INTO Rider values('FP0005', 'Naquib', 27, 'Motorbike');

SELECT * FROM Rider;
```

RIDER_ID	RIDER_NAME	RIDER_AGE	RIDER_VEHICLE
FP0001	Lisa	25	Car
FP0002	Ming	20	Motorbike
FP0003	Yusof	30	Car
FP0004	Kumar	23	Car
FP0005	Naquib	27	Motorbike

Figure 4.2: Rider table after data insertion

3. Restaurant Table

```
INSERT INTO Restaurant values('FRP 1101', 'McDonalds', 4.8, 'Fast Food', 'Bandar Sri Damansara, Kuala Lumpur');
INSERT INTO Restaurant values('FRP 1102', 'Dominos Pizza', 4.8, 'Pizza', 'Cyberjaya, Selangor');
INSERT INTO Restaurant values('FRP 1103', 'Secret Recipe', 2.5, 'Dessert', 'Tesco Semenyih, Selangor');
INSERT INTO Restaurant values('FRP 1104', 'Sushi Zanmai', 3.0, 'Asian', 'NU Sentral, Kuala Lumpur');
INSERT INTO Restaurant values('FRP 1105', 'KFC', 4.0, 'Fast Food', 'The Curve, Selangor');
INSERT INTO Restaurant values('FRP 1106', 'Tealive', 3.9, 'Beverage', 'IOI City Mall, Selangor');
INSERT INTO Restaurant values('FRP 1107', 'Starbucks', 4.0, 'Beverage', 'TTDI, Kuala Lumpur');
```

RESTAURANT_ID	RESTAURANT_NAME	RESTAURANT_RATE	RESTAURANT_TYPE	RESTAURANT_ADDRESS
FRP 1101	McDonalds	4.8	Fast Food	Bandar Sri Damansara, Kuala Lumpur
FRP 1102	Dominos Pizza	4.8	Pizza	Cyberjaya, Selangor
FRP 1103	Secret Recipe	2.5	Dessert	Tesco Semenyih, Selangor
FRP 1104	Sushi Zanmai	3	Asian	NU Sentral, Kuala Lumpur
FRP 1105	KFC	4	Fast Food	The Curve, Selangor
FRP 1106	Tealive	3.9	Beverage	IOI City Mall, Selangor
FRP 1107	Starbucks	4	Beverage	TTDI, Kuala Lumpur

Figure 4.3: Restaurant table after data insertion

4. Order Table

```
INSERT INTO Ordr
values('#b1bx-2pd8','01-Jan-2021',9.76,'11:50:32','CS-0001',
'FP0001');
INSERT INTO Ordr
values('#k3fy-5fg3','03-Jan-2021',11.80,'10:47:34','CS-0002','FP0
002');
INSERT INTO Ordr
values('#m3kz-7zl5','05-Jan-2021',8.30,'09:23:55','CS-0003','FP00
03');
INSERT INTO Ordr
values('#r3nf-9ew9','07-Jan-2021',7.75,'12:30:42','CS-0004','FP00
04');
INSERT INTO Ordr
values('#q3ar-1hj2','11-Jan-2021',5.50,'07:13:27','CS-0005','FP00
05');
INSERT INTO Ordr VALUES('#m4tv-9jt4', '12-FEB-2021', 16.80,
'13:30:45', 'CS-0002','FP0001');
INSERT INTO Ordr VALUES('#f3ku-0rd7', '24-FEB-2021', 20.20,
'15:00:30', 'CS-0003','FP0005');
INSERT INTO Ordr VALUES('#h1qw-7vg8', '09-MAR-2021', 16.30,
'11:23:44', 'CS-0004','FP0003');
INSERT INTO Ordr VALUES('#n7rs-3ju6', '21-MAR-2021', 12.90,
'18:45:53', 'CS-0002','FP0001');
```

ORDER_ID	ORDER_DATE	ORDER_PRICE	ORDER_TIMEFRAME	CUSTOMER_ID	RIDER_ID
#b1bx-2pd8	01-JAN-21	9.76	11:50:32	CS-0001	FP0001
#k3fy-5fg3	03-JAN-21	11.8	10:47:34	CS-0002	FP0002
#m3kz-7zl5	05-JAN-21	8.3	09:23:55	CS-0003	FP0003
#r3nf-9ew9	07-JAN-21	7.75	12:30:42	CS-0004	FP0004
#q3ar-1hj2	11-JAN-21	5.5	07:13:27	CS-0005	FP0005
#m4tv-9jt4	12-FEB-21	16.8	13:30:45	CS-0002	FP0001
#f3ku-0rd7	24-FEB-21	20.2	15:00:30	CS-0003	FP0005
#h1qw-7vg8	09-MAR-21	16.3	11:23:44	CS-0004	FP0003
#n7rs-3ju6	21-MAR-21	12.9	18:45:53	CS-0002	FP0001

Figure 4.4: Order table after data insertion

5. Menu Table

```
INSERT INTO Menu VALUES ('MFP 1000', 'FRP 1107', 'Green Tea Creme Frappuchino', 18.55);
INSERT INTO Menu VALUES ('MFP 1010', 'FRP 1101', 'Spicy Chicken McDeluxe', 9.95);
INSERT INTO Menu VALUES ('MFP 1020', 'FRP 1106', 'Original Pearl Milk Tea', 8.55);
INSERT INTO Menu VALUES ('MFP 1030', 'FRP 1105', 'Snack Plate Combo', 15.30);
INSERT INTO Menu VALUES ('MFP 1040', 'FRP 1103', 'Marble Cheese Cake', 11.30);
INSERT INTO Menu VALUES ('MFP 1050', 'FRP 1102', 'Beef Pepperoni Pizza', 20.20);
INSERT INTO Menu VALUES ('MFP 1060', 'FRP 1103', 'Kimchi Fried Rice With Egg', 12.90);
INSERT INTO Menu VALUES ('MFP 1070', 'FRP 1107', 'Very Berry Strawberry', 11.55);
INSERT INTO Menu VALUES ('MFP 1080', 'FRP 1104', 'Salmon Mentai Roll', 16.80);
INSERT INTO Menu VALUES ('MFP 1090', 'FRP 1105', '3-PC Chicken Combo Set', 16.30);
```

MENU_ID	RESTAURANT_ID	MENU_NAME	MENU_PRICE
MFP 1000	FRP 1107	Green Tea Creme Frappuchino	18.55
MFP 1010	FRP 1101	Spicy Chicken McDeluxe	9.95
MFP 1020	FRP 1106	Original Pearl Milk Tea	8.55
MFP 1030	FRP 1105	Snack Plate Combo	15.3
MFP 1040	FRP 1103	Marble Cheese Cake	11.3
MFP 1050	FRP 1102	Beef Pepperoni Pizza	20.2
MFP 1060	FRP 1103	Kimchi Fried Rice With Egg	12.9
MFP 1070	FRP 1107	Very Berry Strawberry	11.55
MFP 1080	FRP 1104	Salmon Mentai Roll	16.8
MFP 1090	FRP 1105	3-PC Chicken Combo Set	16.3

Figure 4.5: Restaurant table after data insertion

6. Order Menu Table

```
INSERT INTO Order_Menu VALUES('OM 5000','#b1bx-2pd8','MFP 1000');
INSERT INTO Order_Menu VALUES('OM 5001','#k3fy-5fg3','MFP 1010');
INSERT INTO Order_Menu VALUES('OM 5002','#m3kz-7zl5','MFP 1020');
INSERT INTO Order_Menu VALUES('OM 5003','#r3nf-9ew9','MFP 1030');
INSERT INTO Order_Menu VALUES('OM 5004','#q3ar-1hj2','MFP 1040');
INSERT INTO Order_Menu VALUES('OM 5005','#m4tv-9jt4','MFP 1080');
INSERT INTO Order_Menu VALUES('OM 5006','#f3ku-0rd7','MFP 1050');
INSERT INTO Order_Menu VALUES('OM 5007','#h1qw-7vg8','MFP 1090');
INSERT INTO Order_Menu VALUES('OM 5008','#n7rs-3ju6','MFP 1060');
```

ORDERMENU_ID	ORDER_ID	MENU_ID
OM 5000	#b1bx-2pd8	MFP 1000
OM 5001	#k3fy-5fg3	MFP 1010
OM 5002	#m3kz-7zl5	MFP 1020
OM 5003	#r3nf-9ew9	MFP 1030
OM 5004	#q3ar-1hj2	MFP 1040
OM 5005	#m4tv-9jt4	MFP 1080
OM 5006	#f3ku-0rd7	MFP 1050
OM 5007	#h1qw-7vg8	MFP 1090
OM 5008	#n7rs-3ju6	MFP 1060

Figure 4.6: Order menu table after data insertion

7. Promotion Table

```
INSERT INTO Promotion VALUES('PFP 9001', '20% Off on chicken type vendor', '15-Sep-2021');
INSERT INTO Promotion VALUES('PFP 9002', 'Buy 1 Free 1 in Tealive', '30-Mar-2021');
INSERT INTO Promotion VALUES('PFP 9003', '50% Off for buying more than RM 100', '28-Feb-2021');
INSERT INTO Promotion VALUES('PFP 9004', '50% Off for celebrating the festive season', '20-May-2021');
INSERT INTO Promotion VALUES('PFP 9005', '20% Off for buying food from any fast food restaurant', '31-Aug-2021');

SELECT * FROM Promotion;
```

PROMOTION_ID	PROMOTION_FEATURE	PROMOTION_EXPIRY
PFP 9001	20% Off on chicken type vendor	15-SEP-21
PFP 9002	Buy 1 Free 1 in Tealive	30-MAR-21
PFP 9003	50% Off for buying more than RM 100	28-FEB-21
PFP 9004	50% Off for celebrating the festive season	20-MAY-21
PFP 9005	20% Off for buying food from any fast food restaurant	31-AUG-21

Figure 4.7: Promotion table after data insertion

8. Promo_Order Table

```
INSERT INTO Promo_Order VALUES ('PFP 9001','#b1bx-2pd8',0.20);
INSERT INTO Promo_Order VALUES ('PFP 9002','#k3fy-5fg3',0.50);
INSERT INTO Promo_Order VALUES ('PFP 9003','#m3kz-7zl5',0.50);
INSERT INTO Promo_Order VALUES ('PFP 9004','#r3nf-9ew9',0.50);
INSERT INTO Promo_Order VALUES ('PFP 9005','#q3ar-1hj2',0.20);
```

PROMOTION_ID	ORDER_ID	DISCOUNT
PFP 9001	#b1bx-2pd8	0.2
PFP 9002	#k3fy-5fg3	0.5
PFP 9003	#m3kz-7zl5	0.5
PFP 9004	#r3nf-9ew9	0.5
PFP 9005	#q3ar-1hj2	0.2

Figure 4.8: Promotion table after data insertion

9. Payment Table

```
INSERT INTO Payment VALUES('P-15011','Cash On Delivery','01-Jan-2021','#b1bx-2pd8',9.76);
INSERT INTO Payment VALUES('P-15012','Debit Card','03-Jan-2021','#k3fy-5fg3',11.80);
INSERT INTO Payment VALUES('P-15013','Paypal','05-Jan-2021','#m3kz-7zl5',8.30);
INSERT INTO Payment VALUES('P-15014','Credit Card','07-Jan-2021','#r3nf-9ew9',7.75);
INSERT INTO Payment VALUES('P-15015','Cash On Delivery','11-Jan-2021','#q3ar-1hj2',5.50);
INSERT INTO Payment VALUES('P-15016','Debit Card','12-Feb-2021','#m4tv-9jt4',16.80);
INSERT INTO Payment VALUES('P-15017','Debit Card','24-Feb-2021','#f3ku-0rd7',20.20);
INSERT INTO Payment VALUES('P-15018','Grab Wallet','09-Mar-2021','#h1qw-7vg8',16.30);
INSERT INTO Payment VALUES('P-15019','Cash On Delivery','21-Mar-2021','#n7rs-3ju6',20.00);
```

PAYMENT_ID	PAYMENT_METHOD	PAYMENT_DATE	ORDER_ID	AMOUNT_PAID
P-15011	Cash On Delivery	01-JAN-21	#b1bx-2pd8	9.76
P-15012	Debit Card	03-JAN-21	#k3fy-5fg3	11.8
P-15013	Paypal	05-JAN-21	#m3kz-7zl5	8.3
P-15014	Credit Card	07-JAN-21	#r3nf-9ew9	7.75
P-15015	Cash On Delivery	11-JAN-21	#q3ar-1hj2	5.5
P-15016	Debit Card	12-FEB-21	#m4tv-9jt4	16.8
P-15017	Debit Card	24-FEB-21	#f3ku-0rd7	20.2
P-15018	Grab Wallet	09-MAR-21	#h1qw-7vg8	16.3
P-15019	Cash On Delivery	21-MAR-21	#n7rs-3ju6	20

Figure 4.9: Payment table after data insertion

10. Feedback Table

```
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.5,'#b1bx-2pd8');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.2,'#k3fy-5fg3');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,3.5,'#m3kz-7zl5');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,3.1,'#r3nf-9ew9');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,2.0,'#q3ar-1hj2');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.4,'#m4tv-9jt4');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,3.2,'#f3ku-0rd7');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,2.5,'#h1qw-7vg8');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.0,'#n7rs-3ju6');
```

FEEDBACK_ID	FEEDBACK_RATING	ORDER_ID
100	4.5	#b1bx-2pd8
101	4.2	#k3fy-5fg3
102	3.5	#m3kz-7zl5
103	3.1	#r3nf-9ew9
104	2	#q3ar-1hj2
105	4.4	#m4tv-9jt4
106	3.2	#f3ku-0rd7
107	2.5	#h1qw-7vg8
108	4	#n7rs-3ju6

Figure 4.10: Feedback table after data insertion

5.0 Data Manipulation with SQL

5.1: Aggregate Function

1. To display the sum of money paid according to the payment method in descending order according to the total amount paid from each payment method.

   SQL Statement:

   ```
   SELECT Payment_Method, SUM(Amount_Paid) AS Total_Amount
   FROM Payment
   GROUP BY Payment_Method
   ORDER BY Total_Amount;
   ```

PAYMENT_METHOD	TOTAL_AMOUNT
Credit Card	7.75
Paypal	8.3
eWallet	16.3
Cash On Delivery	35.26
Debit Card	48.8

Figure 5.1.1: Table of sum of money spent group by payment methods

2. To display the average feedback from customers rating of each restaurant in ascending order.

   SQL Statement:

   ```
   SELECT Restaurant_Name, AVG(Feedback_Rating) AS Order_AVGRate
   FROM Feedback, Ordr, Order_Menu, Menu, Restaurant
   WHERE Feedback.Order_ID=Ordr.Order_ID AND Ordr.Order_ID=
   Order_Menu.Order_ID AND Order_Menu.Menu_ID = Menu.Menu_ID AND
    Menu.Restaurant_ID = Restaurant.Restaurant_Id
   GROUP BY Restaurant_Name
   ORDER BY Order_AVGRate;
   ```

RESTAURANT_NAME	ORDER_AVGRATE
KFC	2.8
Secret Recipe	3
Dominos Pizza	3.2
Tealive	3.5
McDonalds	4.2
Sushi Zanmai	4.4
Starbucks	4.5

Figure 5.1.2: Table of average restaurant order rating group by restaurant name

3. To display the description of the most expensive menu in the list

   SQL Statement:

   SELECT Menu_ID, Menu_Name, Menu_Price

   FROM Menu

   WHERE Menu_Price

   BETWEEN (SELECT MIN(Menu_Price) FROM Menu) AND 22

   AND Menu_Price = (SELECT MAX(Menu_Price) FROM Menu)

	MENU_ID	MENU_NAME	MENU_PRICE
1	MFP 1050	Beef Pepperoni Pizza	20.2

Figure 5.1.3: Description of the most expensive menu in the list

4. To display the payment id and the number of times it has been used to pay a minimum amount of money where the amount of money has been found in a subset of listed values.

   SQL Statement:

   SELECT Payment_ID, COUNT(PAYMENT_ID), MIN(Amount_Paid)

   FROM Payment

   GROUP BY Payment_ID

   HAVING MIN(Amount_Paid) IN(5.50,7.75,8.30)

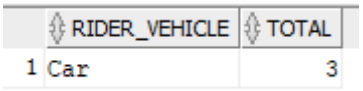
	PAYMENT_ID	COUNT(PAYMENT_ID)	MIN(AMOUNT_PAID)
1	P-15013	1	8.3
2	P-15014	1	7.75
3	P-15015	1	5.5

Figure 5.1.4: The minimum amount of money paid by specific payment_id

5. To display the vehicle type with the number of usage by the riders respectively.

SQL Queries:

```
SELECT Rider_Vehicle, COUNT(*) AS Total
FROM Rider
GROUP BY Rider_Vehicle
HAVING COUNT(*) > 2;
```



	RIDER_VEHICLE	TOTAL
1	Car	3

Figure 5.1.5: Vehicle type with the number of usage by the riders respectively.

5.2 Use of subqueries/nested queries

1. Subqueries with SELECT Statement

   To display the food and beverages available from the list of menu priced equal to or above RM15.00.

   SQL Statement:

```
SELECT * FROM MENU WHERE Menu_ID IN
(SELECT Menu_ID FROM MENU
 WHERE Menu_Price >= 15.00) ;
```

MENU_ID	RESTAURANT_ID	MENU_NAME	MENU_PRICE
MFP 1000	FRP 1107	Green Tea Creme Frappuchino	18.55
MFP 1030	FRP 1105	Snack Plate Combo	15.3
MFP 1050	FRP 1102	Beef Pepperoni Pizza	20.2
MFP 1080	FRP 1104	Salmon Mentai Roll	16.8
MFP 1090	FRP 1105	3-PC Chicken Combo Set	16.3

Figure 5.2.1: Table of Menu where the Menu Price is more than RM15.00

2. Nested queries

   To display the rider's name who delivered orders between 6th January 2021 and 15th January 2021 based on the Order ID.

   SQL Statement:

```
SELECT Rider_Name
 FROM RIDER WHERE Rider_ID IN
 (SELECT Rider_ID FROM Ordr
 WHERE Order_Date BETWEEN '06-Jan-2021' AND '15-Jan-2021' );
```

RIDER_NAME
Kumar
Naquib

Figure 5.2.2: Rider Names From Table Rider who delivered between 6th Jan 2021 to 15th Jan 2021 displayed.

5.3 Trigger and View

1. A virtual order table with an updated column named ‘Reward_Point’.

   SQL Queries:

   ALTER TABLE Ordr ADD Reward_Point NUMBER(3);

   CREATE VIEW virtualOrder AS SELECT * FROM Ordr;

ORDER_ID	ORDER_DATE	ORDER_PRICE	ORDER_TIMEFRAME	CUSTOMER_ID	RIDER_ID	REWARD_POINT
#blbx-2pd8	01-JAN-21	9.76	11:50:32	CS-0001	FP0001	(null)
#k3fy-5fg3	03-JAN-21	11.8	10:47:34	CS-0002	FP0002	(null)
#m3kz-7zl5	05-JAN-21	8.3	09:23:55	CS-0003	FP0003	(null)
#r3nf-9ew9	07-JAN-21	7.75	12:30:42	CS-0004	FP0004	(null)
#q3ar-1hj2	11-JAN-21	5.5	07:13:27	CS-0005	FP0005	(null)
#m4tv-9jt4	12-FEB-21	16.8	13:30:45	CS-0002	FP0001	(null)
#f3ku-0rd7	24-FEB-21	20.2	15:00:30	CS-0003	FP0005	(null)
#hlqw-7vg8	09-MAR-21	16.3	11:23:44	CS-0004	FP0003	(null)
#n7rs-3ju6	21-MAR-21	12.9	18:45:53	CS-0002	FP0001	(null)
#k4ct-6yh6	22-MAR-21	18.55	14:50:04	CS-0002	FP0001	(null)
#t7jy-7njl	23-MAR-21	9.95	18:45:53	CS-0003	FP0005	(null)
#r5ds-1bn3	25-MAR-21	20.2	18:45:53	CS-0002	FP0001	(null)

Figure 5.3.1: Virtual order table after the original has been altered to create new column called ‘Reward_Point’

2. A trigger that simultaneously updates the reward point column in the virtual order table where if a price of an order is more than RM 10.00, the reward point that will be received to the customer will be indicated in the reward point column as 50 whereas less than the stated price, the reward point will be shown are 25.

SQL Queries:

```
CREATE OR REPLACE TRIGGER trigCustomer
INSTEAD OF INSERT ON virtualOrder
FOR EACH ROW
BEGIN

    UPDATE Ordr
    SET Reward_Point = 50
    WHERE Order_Price > 10.00;

    UPDATE Ordr
    SET Reward_Point = 25
    WHERE Order_Price < 10.00;

END;
```

ORDER_ID	ORDER_DATE	ORDER_PRICE	ORDER_TIMEFRAME	CUSTOMER_ID	RIDER_ID	REWARD_POINT
#blbx-2pd8	01-JAN-21	9.76	11:50:32	CS-0001	FP0001	25
#k3fy-5fg3	03-JAN-21	11.8	10:47:34	CS-0002	FP0002	50
#m3kz-7zl5	05-JAN-21	8.3	09:23:55	CS-0003	FP0003	25
#r3nf-9ew9	07-JAN-21	7.75	12:30:42	CS-0004	FP0004	25
#q3ar-1hj2	11-JAN-21	5.5	07:13:27	CS-0005	FP0005	25
#m4tv-9jt4	12-FEB-21	16.8	13:30:45	CS-0002	FP0001	50
#f3ku-0rd7	24-FEB-21	20.2	15:00:30	CS-0003	FP0005	50
#hlqw-7vg8	09-MAR-21	16.3	11:23:44	CS-0004	FP0003	50
#n7rs-3ju6	21-MAR-21	12.9	18:45:53	CS-0002	FP0001	50
#k4ct-6yh6	22-MAR-21	18.55	14:50:04	CS-0002	FP0001	50
#t7jy-7nj1	23-MAR-21	9.95	18:45:53	CS-0003	FP0005	25
#r5ds-1bn3	25-MAR-21	20.2	18:45:53	CS-0002	FP0001	50

Figure 5.3.2: Virtual order table reward point column has been updated by trigger

5.4 Stored Procedure

1. To display the price of the menu after applying the 50% off discount.

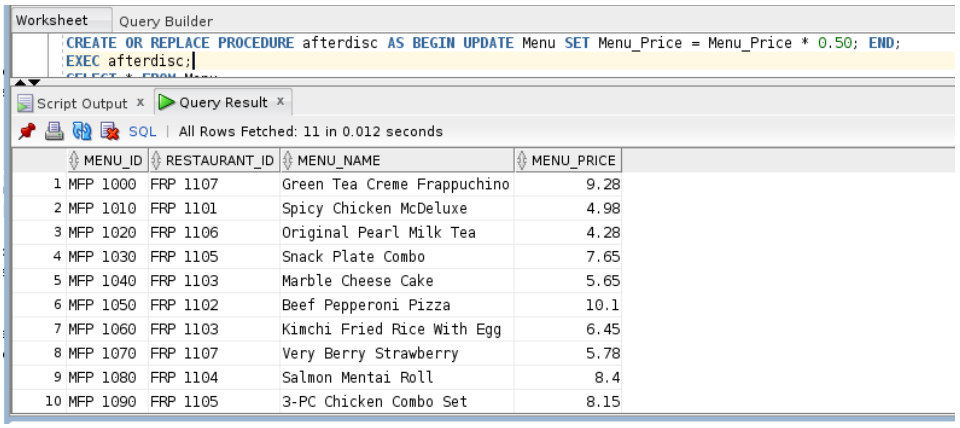
   SQL Statement:

   CREATE OR REPLACE PROCEDURE afterdisc AS BEGIN UPDATE Menu SET Menu_Price = Menu_Price*0.50;

   END;

   EXEC afterdisc;

   SELECT * FROM Menu;



	MENU_ID	RESTAURANT_ID	MENU_NAME	MENU_PRICE
1	MFP 1000	FRP 1107	Green Tea Creme Frappuchino	9.28
2	MFP 1010	FRP 1101	Spicy Chicken McDeluxe	4.98
3	MFP 1020	FRP 1106	Original Pearl Milk Tea	4.28
4	MFP 1030	FRP 1105	Snack Plate Combo	7.65
5	MFP 1040	FRP 1103	Marble Cheese Cake	5.65
6	MFP 1050	FRP 1102	Beef Pepperoni Pizza	10.1
7	MFP 1060	FRP 1103	Kimchi Fried Rice With Egg	6.45
8	MFP 1070	FRP 1107	Very Berry Strawberry	5.78
9	MFP 1080	FRP 1104	Salmon Mentai Roll	8.4
10	MFP 1090	FRP 1105	3-PC Chicken Combo Set	8.15

Figure 5.4.1: Updated Menu Table where Menu Price after applying 50% discount displayed

5.5 Not covered in lecture

1. Ranking riders based on most deliveries in ascending order by displaying the Rider ID and Rider Name.

   SQL Queries:

   SELECT Ordr.Rider_ID, Rider_Name, RANK() OVER(ORDER BY COUNT(*) DESC) AS Rider_Rank
   FROM Ordr, Rider
   WHERE Ordr.Rider_ID = Rider.Rider_ID
   GROUP BY Ordr.Rider_ID, Rider_Name
   ORDER BY COUNT(*) DESC;

RIDER_ID	RIDER_NAME	RIDER_RANK
FP0001	Lisa	1
FP0005	Naquib	2
FP0003	Yusof	3
FP0002	Ming	4
FP0004	Kumar	4

Figure 5.5.1: Result table with the ranking of the riders that delivers most customers orders

2. Auto-incrementing varchar data type for riders by just typing ‘FP’ for the ID whereas their name, age, and type of vehicle they use are to be types normally.

SQL Queries:

```
CREATE SEQUENCE Rider_Sequence
MINVALUE 6
MAXVALUE 999
START WITH 6
INCREMENT BY 1
NOCACHE;

CREATE OR REPLACE TRIGGER trgRider
BEFORE INSERT ON Rider
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
   BEGIN
      :NEW.Rider_ID := :NEW.Rider_ID ||
TRIM(TO_CHAR(Rider_Sequence.nextval, '0000'));
   END;
```

RIDER_ID	RIDER_NAME	RIDER_AGE	RIDER_VEHICLE
FP0001	Lisa	25	Car
FP0002	Ming	20	Motorbike
FP0003	Yusof	30	Car
FP0004	Kumar	23	Car
FP0005	Naquib	27	Motorbike

Figure 5.5.2: Before using the Auto-Increment

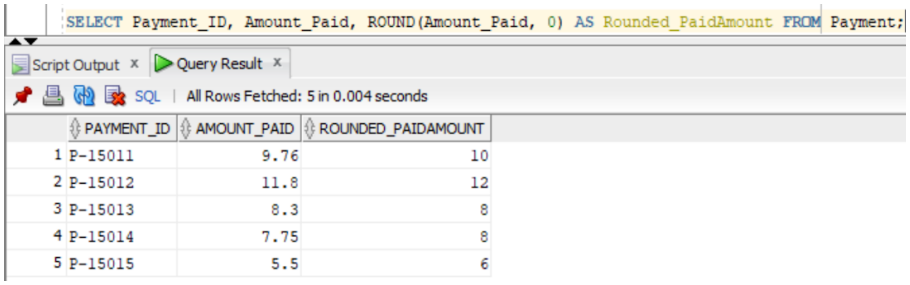
RIDER_ID	RIDER_NAME	RIDER_AGE	RIDER_VEHICLE
FP0001	Lisa	25	Car
FP0002	Ming	20	Motorbike
FP0003	Yusof	30	Car
FP0004	Kumar	23	Car
FP0005	Naquib	27	Motorbike
FP0006	Steven	25	Motorbike

Figure 5.5.3: After using Auto-Increment by entering Rider_ID ‘FP’

3. Rounding off the amount paid for the orders and organizing the rounded paid amount in a new table called 'Rounded_PaidAmount'.

   SQL Statement:

   SELECT Payment_ID, Amount_Paid, ROUND(Amount_Paid, 0) AS Rounded_PaidAmount FROM Payment;



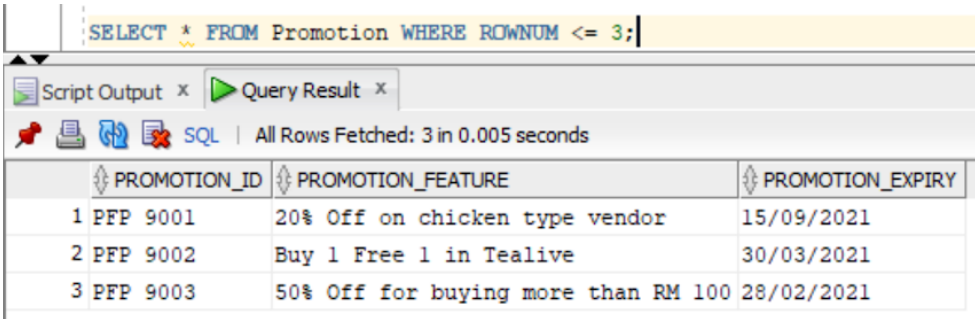
	PAYMENT_ID	AMOUNT_PAID	ROUNDED_PAIDAMOUNT
1	P-15011	9.76	10
2	P-15012	11.8	12
3	P-15013	8.3	8
4	P-15014	7.75	8
5	P-15015	5.5	6

Figure 5.5.4: After using Round() to round off the amount paid orders.

4. To display the promotions offered from row 1 until row 3 from the Promotion Table.

   SQL Statement:

   SELECT * FROM Promotion WHERE ROWNUM <= 3;



	PROMOTION_ID	PROMOTION_FEATURE	PROMOTION_EXPIRY
1	PFP 9001	20% Off on chicken type vendor	15/09/2021
2	PFP 9002	Buy 1 Free 1 in Tealive	30/03/2021
3	PFP 9003	50% Off for buying more than RM 100	28/02/2021

Figure 5.5.5: Promotions from row 1-3 displayed.

5. Auto-incrementing number data type will be used when we create the sequence and it will start the value as 100 for Feedback_ID and it automatically increments the value by 1 after the insertion of new data.
   SQL Statement:
   CREATE SEQUENCE seq_feedback
   MINVALUE 100
   START WITH 100
   INCREMENT BY 1
   CACHE 10;

INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.5,'#b1bx-2pd8');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.2,'#k3fy-5fg3');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,3.5,'#m3kz-7zl5');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,3.1,'#r3nf-9ew9');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,2.0,'#q3ar-1hj2');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.4,'#m4tv-9jt4');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,3.2,'#f3ku-0rd7');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,2.5,'#h1qw-7vg8');
INSERT INTO Feedback (Feedback_ID, Feedback_Rating, Order_ID)
Values(seq_feedback.nextval,4.0,'#n7rs-3ju6');

	FEEDBACK_ID	FEEDBACK_RATING	ORDER_ID
1	100	4.5	#blbx-2pd8
2	101	4.2	#k3fy-5fg3
3	102	3.5	#m3kz-7z15
4	103	3.1	#r3nf-9ew9
5	104	2	#q3ar-1hj2
6	105	4.4	#m4tv-9jt4
7	106	3.2	#f3ku-0rd7
8	107	2.5	#hlqw-7vg8
9	108	4	#n7rs-3ju6

Figure 5.5.6: After using Auto-Increment by using the SQL sequence

5.6 Query with a *group by* and *having* clauses

1. To display the number of deliveries that must be more than 1 which has been made by some riders and also it is counted in ascending order.

   SQL Statement:

```
SELECT
   Rider_Name,
   COUNT(*) AS num_deliveries
FROM
   Rider r INNER JOIN Ordr o
   ON r.Rider_ID = o.Rider_ID
GROUP BY
   Rider_Name
HAVING
   COUNT(*) > 1
ORDER BY
   num_deliveries;
```

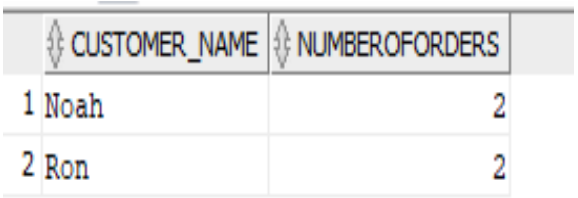
	RIDER_NAME	NUM_DELIVERIES
1	Yusof	2
2	Naquib	2
3	Lisa	3

Figure 5.6.1: Number of deliveries made by specific riders

2. To display the number of orders that must be more than 1 which has been made by some customers and the date of order must be in the range of 2-Jan-2021 till 25-Feb-2021

SQL Statement:

```
SELECT Customer.Customer_Name, COUNT(Ordr.Order_ID) AS NumberOfOrders
FROM Ordr
LEFT JOIN Customer ON Ordr.Customer_ID = Customer.Customer_ID
WHERE Order_Date BETWEEN '2-Jan-2021' AND '25-Feb-2021'
GROUP BY Customer_Name
HAVING COUNT(Ordr.Order_ID) > 1;
```



	CUSTOMER_NAME	NUMBEROFORDERS
1	Noah	2
2	Ron	2

Figure 5.6.2: Number of orders made by specific customers

3. To display the total price of the orders made by the customers respectively, The total price must be less than RM20 and also the customer name must contain the character ‘n’.

SQL Statement:

```
SELECT c.Customer_Name, SUM (o.Order_Price)
FROM Customer c, Ordr o
WHERE c.Customer_ID = o.Customer_ID
AND Customer_Name LIKE '%n%'
AND Order_Price IS NOT NULL
Group By c.Customer_Name
Having SUM(o.Order_Price) < 20
```

	CUSTOMER_NAME	SUM(O.ORDER_PRICE)
1	Ginny	5.5

Figure 5.6.3: The total price of the orders made by the customers respectively

Contribution

Name (Student ID)	Contribution Percentage
Muhammad Uzair Bin Abdul Razak (1191303163)	100
Iman Aisyah binti Zailani (1191302815)	100
Kishen Kumar A/L Sivalingam (1191101423)	100