

**UNIVERSITI TUNKU ABDUL RAHMAN**

**Faculty of Information and Communication Technology (FICT)**

**UCCD2303 DATABASE TECHNOLOGY /**

**UCCD2203 DATABASE SYSTEMS**

**Group Assignment Mark sheet**

February 2025 Trimester

|  |  |
| --- | --- |
| Group Number (e.g. G999): | G087 |
| Group leader to provide the OneDrive folder hyperlink (editor mode) which contains the zip file and a group presentation video in mp4 format: | https://utarict-my.sharepoint.com/:f:/g/personal/chaijierong\_1utar\_my/EotRKuJ4PoBPoezC5AwkYywBokHGHhkIABBCTWMFhO8kog?e=CdnNnk |
| Group leader name: | Chai Jie Rong |
| Group leader email address: | chaijierong@1utar.my |
| Submission date (dd-Mon-yy): | 29/4/25 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Member Contribution Declaration Sheet** | | | | | |
|  | | **Member 1** | **Member 2** | **Member 3** | **Member 4** |
| **Member Name** (ascending order, group leader with \* ): | | **Chai Jie Rong** | **Ching Yin Kean** | **Tan Yi Xuan** | **Thau Zhe Jun** |
| Student ID (e.g. 2299999) | | **2207093** | **2207267** | **2204266** | **2206252** |
| Program CS / IA, IB / DE: | | **IA** | **IA** | **IA** | **IA** |
| **Member’s signature** | |  | A signature of a person  AI-generated content may be incorrect. |  |  |
|  | | Contribution of each member (Total 100%) | | | |
|  | | **%** | **%** | **%** | **%** |
| PART 1: (Group Assessment – 50 marls) | |  |  |  |  |
| **1.** | Scope of Work (5 marks). | **100** | **0** | **0** | **0** |
| **2.** | ER model (10 marks). | **0** | **0** | **100** | **0** |
| **3.** | Redesign and EER (10 marks). | **0** | **0** | **0** | **100** |
| **4.** | Data Dictionary (10 marks) | **0** | **100** | **0** | **0** |
| **5.** | Tables and records (5 marks) | **100** | **0** | **0** | **0** |
| **6.** | Script (10 marks) | **25** | **25** | **25** | **25** |

**Table of Contents:**

1. **Scope of Work ---------------------- 4-5**
2. **ER model --------------------------- 6**
3. **Redesign and EER ---------------- 7**
4. **Data Dictionary ---------------------- 8-18**
5. **Tables and Records ----------- 19-23**
6. **Scripts ---------------------------------- 24-39**

**1.Scope of Work**

**Introduction**

The proposed of this project is to design and development of a comprehensive Point of Sales (POS) customize for The Foodie, which is a fast-food restaurant chain under the Grand Wooly conglomerate. As The Foodie specialize in high-demand item such as burgers and fried chicken. Hence, the POS system must be stronger, scalable and user-friendly to ensure smooth daily operations across multiple service channels like counter service, self-order kiosks, delivery and self-pickup. The POS system aim to automate and streamline the full cycle of customer transactions, order management, and sales reporting. By combining key operational modules into a centralized database system, The Foodie can be increased speed of service, reduced human error, enhanced inventory linkage and real-time analytics for better decision-making.

Key Functionalities of the Point of Sales System:

1. Inventory Tracking

-Creation, updating and tracking of inventory records. Each product will be linked to a specific category and will contain details such as the product name, stock quantity, and unit price.

2. Sales Tracking

-Every product sold recorded through a structured sales tracking process. Sales information will include details like product name, quantity sold, price per item, and total sale amount

3. Transaction Processing

-record and manage the full transaction process. It will capture the order details, calculate the total price, apply any discount or promotions and record the selected payment method

4. Customer Management

-maintain a customer database that stores essential customer information which includes customer name, contact information and order history.

5. Staff Role and Sales Association

-Each sale will be linked to a staff member, allowing management to monitor staff performance and accountability.

6. Product Categorization

-All product will be organized under specific categories such as beverages, snacks, or combo meals.

7.Payment Management

-The system will support multiple payment methods, like cash, debit/credit card, and QR payments. Each payment will be recorded with its corresponding order ID, payment type, and payment timestamp.

8. Real-Time Stock Deduction

-Stock levels will automatically updated in real time as items are sold. When an order is completed, the system will deduct the purchased quantities form the respective product stock.

9. Order History & Receipt Generation

-For each transaction, the system will generate a detailed receipt including itemized product, quantities, total amount and payment method.

10. Dashboard & Analytics

-An administrative dashboard can be added to visualize key business metrics which may include the total daily sales, top-selling products, staff performance, and low-stock alerts.

The system design includes database management alongside well-defined entities and relationship definitions and constraints to achieve operational excellence. The system design will implement database-driven modules through relational tables which follow business rule assumptions. For example, it is assumed that each supplier may provide multiple items, each of which may belong to a specific category and be stored in multiple locations at the same time. The system establishes well-organized relationships between customers and transactions and inventory that ensure data consistency and reduces redundancy. The system will create multiple user roles which will provide different access permissions to guarantee data protection and operational performance maintenance. The inventory managers will achieve total access to stock management functionality which includes inventory level visibility and automatic restocking notifications. The transaction data along with customer records and promotional offers will be handled by sales managers who share responsibility with purchase managers for managing supplier relationships and purchase orders. Analyzed data will be accessible to analysts who can only view information for creating reports that help business decisions.

**2.ER model**

A diagram of a company

AI-generated content may be incorrect.

**3. Redesign and EER**

A screenshot of a computer

AI-generated content may be incorrect.

**4. Data Dictionary**

**Table Name: Customer**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| customer\_id | PK | NUMBER | N/A | 10 | Unique ID for each customer |
| cusName | NN | VARCHAR2 | N/A | 100 | Customer's name |
| cusPhoneNumber | NN | VARCHAR2 | N/A | 15 | Customer's phone number |
| email | NN | VARCHAR2 | N/A | 100 | Customer's email address |

**Table Name: Order**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| order\_id | PK | NUMBER | N/A | 10 | Unique ID for each order |
| customer\_id | FK | NUMBER | N/A | 10 | ID of the customer who placed the order |
| Employee\_id | FK | NUMBER | N/A | 10 | ID of the employee who handled the order |
| orderType | NN | VARCHAR2 | N/A | 50 | Type of order (e.g., dine-in, takeout) |
| order\_DateTime | NN | DATE | DD-MON-YYYY HH24:MI:SS | N/A | Date and time when the order was placed |
| orderStatus | NN | VARCHAR2 | N/A | 50 | Status of the order (e.g., pending, completed) |
| total | NN | NUMBER | N/A | 10,2 | Total amount for the order |

**Table Name: Order\_Detail**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| OrderDetail\_id | PK,FK | NUMBER | N/A | 10 | ID of the order details |
| order\_id | PK, FK | NUMBER | N/A | 10 | ID of the order |
| product\_id | PK, FK | NUMBER | N/A | 10 | ID of the product |
| productName | NN | VARCHAR2 | N/A | 100 | Name of the product |
| price | NN | NUMBER | N/A | 10,2 | Price of the product |
| quantity | NN | NUMBER | N/A | 10 | Quantity of the product ordered |
| subtotal | NN | NUMBER | N/A | 10,2 | Subtotal for the order detail |
| pickupTime | NN | DATE | DD-MON-YYYY HH24:MI:SS | N/A | Time when the order is scheduled to be picked up |

**Table Name: Product**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| product\_id | PK | NUMBER | N/A | 10 | Unique ID for each product |
| productName | NN | VARCHAR2 | N/A | 100 | Name of the product |
| stockQuantity | NN | NUMBER | N/A | 10 | Quantity of the product in stock |
| price | NN | NUMBER | N/A | 10,2 | Price of the product |

**Table Name: Order\_Items**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| order\_id | PK, FK | NUMBER | N/A | 10 | ID of the order |
| OrderDetail\_id | PK, FK | NUMBER | N/A | 10 | ID of the order details |
| orderQuantity | NN | NUMBER | N/A | 10 | Quantity of the product in the order |

**Table Name: Payment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| payment\_id | PK | NUMBER | N/A | 10 | Unique ID for each payment |
| order\_id | FK | NUMBER | N/A | 10 | ID of the order being paid for |
| discount\_id | FK | NUMBER | N/A | 10 | Unique ID for each discount |
| paymentMethod | NN | VARCHAR2 | N/A | 50 | Method of payment (e.g., cash, card) |
| paymentStatus | NN | VARCHAR2 | N/A | 50 | Status of the payment (e.g., completed, pending) |

**Table Name: Receipt**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| receipt\_id | PK | NUMBER | N/A | 10 | Unique ID for each receipt |
| order\_id | FK | NUMBER | N/A | 10 | ID of the order |
| payment\_id | FK | NUMBER | N/A | 10 | ID of the payment |
| discount\_id | FK | NUMBER | N/A | 10 | ID of the discount applied |
| Employee\_id | FK | NUMBER | N/A | 10 | ID of the employee who generated the receipt |
| issueDate | NN | DATE | DD-MON-YYYY HH24:MI:SS | N/A | Date and time the receipt was issued |
| totalAmount | NN | NUMBER | N/A | 10,2 | Total amount before discount |
| discountApplied | NN | NUMBER | N/A | 10,2 | Discount applied to the order |
| finalAmount | NN | NUMBER | N/A | 10,2 | Amount after discount |

**Table Name: Discount**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| discount\_id | PK | NUMBER | N/A | 10 | Unique ID for each discount |
| discountName | NN | VARCHAR2 | N/A | 100 | Name of the discount |
| discountType | NN | VARCHAR2 | N/A | 50 | Type of discount (e.g., percentage, flat) |
| discountValue | NN | NUMBER | N/A | 10,2 | Value of the discount |
| startDate | NN | DATE | DD-MON-YYYY | N/A | Start date of the discount |
| endDate | NN | DATE | DD-MON-YYYY | N/A | End date of the discount |

**Table Name: Employee**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| Employee\_id | PK | NUMBER | N/A | 10 | Unique ID for each employee |
| Supervisor\_id | FK | NUMBER | N/A | 10 | ID of the supervisor of the employee |
| EmpName | NN | VARCHAR2 | N/A | 100 | Employee's name |
| role | NN | VARCHAR2 | N/A | 50 | Role of the employee (e.g., cashier, cook) |

**Table Name: Delivery**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| delivery\_id | PK | NUMBER | N/A | 10 | Unique ID for each delivery |
| order\_id | FK | NUMBER | N/A | 10 | ID of the order being delivered |
| Employee\_id | FK | NUMBER | N/A | 10 | ID of the employee delivering the order |
| address | NN | VARCHAR2 | N/A | 200 | Delivery address |
| delivery\_DateTime | NN | DATE | DD-MON-YYYY HH24:MI:SS | N/A | Date and time of delivery |
| cusPhoneNumber | NN | VARCHAR2 | N/A | 15 | Customer's phone number for delivery |
| deliveryFee | NN | NUMBER | N/A | 10,2 | Fee for the delivery service |
| deliveryStatus | NN | VARCHAR2 | N/A | 50 | Status of the delivery (e.g., pending, completed) |

**Table Name: Location**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| delivery\_id | FK | NUMBER | N/A | 10 | ID of the related delivery |
| order\_id | PK,FK | NUMBER | N/A | 10 | Unique ID for each order |
| address | NN | VARCHAR2 | N/A | 200 | Location address |

**Table Name: Supplier**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| Supplier\_id | PK | NUMBER | N/A | 10 | Unique ID for each supplier |
| supplierName | NN | VARCHAR2 | N/A | 100 | Name of the supplier |
| telephoneNum | NN | VARCHAR2 | N/A | 15 | Telephone number of the supplier |

**Table Name: Supplier\_Product**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| Supplier\_id | PK, FK | NUMBER | N/A | 10 | ID of the supplier |
| product\_id | PK, FK | NUMBER | N/A | 10 | ID of the product |
| supply\_price | NN | NUMBER | N/A | 10,2 | Price at which the supplier provides the product |
| supply\_date | NN | DATE | DD-MON-YYYY | N/A | Date the product was supplied |
| supply\_quantity | NN | NUMBER | N/A | 10 | Quantity of the product supplied |

**Table Name: Food\_Beverages**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| product\_id | PK, FK | NUMBER | N/A | 10 | ID of the product |
| exp\_date | NN | DATE | DD-MON-YYYY | N/S | Date the food was expired |
| storage\_requirement | NN | VARCHAR2 | N/A | 100 | Storage requirement for the product |

**Table Name: Merchandise**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| product\_id | PK, FK | NUMBER | N/A | 10 | ID of the product |
| Merch\_Color | NN | VARCHAR2 | N/A | 50 | Color of the merchandise |
| Merch\_Quantity | NN | NUMBER | N/A | 10 | Quantity of the merchandise |

**Table Name: Supervisor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Constraints** | **Data Type** | **Data Format** | **Field Size** | **Description** |
| Employee\_id | PK, FK | NUMBER | N/A | 10 | References employee |
| Supervisor\_id | FK | NUMBER | N/A | 10 | ID of the supervisor |
| Super\_PhoneNumber | NN | VARCHAR2 | N/A | 15 | Phone number of the supervisor |

**Table Name: Delivery Rider**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Constraints** | **Data Type** | **Data Format** | **Field Size** | **Description** |
| Employee\_id | PK, FK | NUMBER | N/A | 10 | References employee |
| Rider\_Vehicle | NN | VARCHAR2 | Alphanumeric | 50 | Vehicle used by the delivery rider |
| Rider\_License | NN | VARCHAR2 | Alphanumeric | 50 | Delivery rider's license number |

**Table Name: Cleaner**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| Employee\_id | PK, FK | NUMBER | Numeric | 10 | References employee |
| Clean\_ShiftTime | NN | VARCHAR2 | Alphanumeric | 20 | Shift time for cleaning duties |
| Clean\_Name | NN | VARCHAR2 | Alphanumeric | 100 | Name of the cleaner |

**Table Name: Service**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute Name | Constraints | Data Type | Data Format | Field Size | Description |
| Employee\_id | PK, FK | NUMBER | Numeric | 10 | References employee |
| customer\_id | PK, FK | NUMBER | Numeric | 10 | References customer |
| service\_status | NN | VARCHAR2 | Alphanumeric | 20 | Status of service (e.g., completed, pending) |

**5.Table and record**

**Table name: Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| customer\_id | cusName | cusPhoneNumber | email |
| 1 | Anthony Davis | 011-11110000 | [davis@gmail.com](mailto:davis@gmail.com) |
| 2 | Lebron James | 012-22220000 | [james@gmail.com](mailto:james@gmail.com) |
| 3 | Kobe Bryant | 012-33330000 | [bryant@gmail.com](mailto:bryant@gmail.com) |
| 4 | Paul George | 013-44440000 | [george@gmail.com](mailto:george@gmail.com) |
| 5 | Stephen Curry | 016-55550000 | [curry@gmail.com](mailto:curry@gmail.com) |
| 6 | Ja Morant | 016-66660000 | [morant@gmail.com](mailto:morant@gmail.com) |
| 7 | Micheal Jordan | 013-77770000 | [jordan@gmail.com](mailto:jordan@gmail.com) |
| 8 | Klay Thompson | 012-88880000 | [thompson@gmail.com](mailto:thompson@gmail.com) |
| 9 | Kevin Durant | 012-99990000 | [durant@gmail.com](mailto:durant@gmail.com) |
| 10 | Allen Iverson | 011-00001111 | [iverson@gmail.com](mailto:iverson@gmail.com) |

**Table name: Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| Employee\_id | Supervisor\_id | EmpName | role |
| 1 | 4 | Magic Johnson | Cashier |
| 2 | 4 | Cameron Anthony | Manager |
| 3 | 4 | Shaque Oneal | Service |
| 4 |  | Adams | Supervisor |
| 5 | 4 | Dwyane Wade | Delivery |
| 6 | 4 | Peter Black | Chef |
| 7 | 4 | Sarah Lee | Chef |
| 8 | 4 | John Hill | Delivery |

**Table name: Product**

|  |  |  |  |
| --- | --- | --- | --- |
| product\_id | productName | stockQuantity | price |
| 1 | Cheeseburger | 100 | 10 |
| 2 | Fried Chicken | 80 | 4 |
| 3 | French Fries | 150 | 6.5 |
| 4 | Soft Drink | 200 | 3.5 |
| 5 | Ice Cream | 60 | 2 |
| 6 | Chicken Nuggets | 70 | 6 |
| 7 | Fish Burger | 80 | 7.5 |
| 8 | Beef Burger | 40 | 8 |
| 9 | Onion Rings | 120 | 2.5 |

**Table name: Order**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| order\_id | customer\_id | employee\_id | orderType | order\_DateTime | orderStatus |
| 1 | 2 | 3 | Dine-In | 2025-04-05 11:00:00 | Completed |
| 2 | 3 | 1 | Takeaway | 2025-04-05 11:30:00 | Completed |
| 3 | 5 | 5 | Delivery | 2025-04-05 12:00:00 | Delivered |
| 4 | 7 | 8 | Delivery | 2025-04-05 12:15:00 | Delivered |
| 5 | 1 | 3 | Dine-In | 2025-04-05 12:45:00 | Completed |
| 6 | 4 | 1 | Pickup | 2025-04-05 13:00:00 | Completed |
| 7 | 9 | 5 | Delivery | |  | | --- | |  |  |  | | --- | | 2025-04-05 13:15:00 | | Pending |
| 8 | 10 | 5 | Delivery | 2025-04-05 13:30:00 | Delivered |
| 9 | 8 | 2 | Takeaway | |  | | --- | |  |  |  | | --- | | 2025-04-05 14:00:00 | | Completed |
| 10 | 6 | 8 | Delivery | |  | | --- | |  |  |  | | --- | | 2025-04-05 14:15:00 | | Pending |

**Table name: Order\_Detail**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| orderDetail\_id | order\_id | product\_id | productName | price | quantity | subtotal | pickupTime |
| 1 | 1 | 1 | Cheeseburger | 10 | 2 | 20 |  |
| 2 | 2 | 1 | Cheeseburger | 10 | 1 | 10 | 2025-04-05 11:39:00 |
| 3 | 3 | 3 | French Fries | 6.5 | 3 | 19.5 |  |
| 4 | 4 | 5 | Ice Cream | 2 | 2 | 4 |  |
| 5 | 5 | 6 | Chicken Nuggets | 6 | 3 | 18 |  |
| 6 | 6 | 6 | Chicken Nuggets | 6 | 4 | 24 | 2025-04-05 13:05:00 |
| 7 | 7 | 7 | Fish Burger | 7.5 | 2 | 15 |  |
| 8 | 8 | 8 | Beef Burger | 8 | 1 | 8 |  |
| 9 | 9 | 2 | Fried Chicken | 4 | 3 | 12 | 2025-04-05 14:10:00 |
| 10 | 10 | 2 | Fried Chicken | 4 | 4 | 16 |  |

**Table name: Payment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| payment\_id | order\_id | discount\_id | paymentMethod | paymentStatus |
| 1 | 1 | 5 | Credit Card | Paid |
| 2 | 2 |  | Cash | Paid |
| 3 | 3 | 5 | Online Payment | Paid |
| 4 | 4 |  | Debit Card | Pending |
| 5 | 5 | 5 | Cash | Paid |
| 6 | 6 | 5 | Credit Card | Paid |
| 7 | 7 |  | Online Payment | Paid |
| 8 | 8 |  | Debit Card | Paid |
| 9 | 9 | 5 | Cash | Pending |
| 10 | 10 |  | Credit Card | Paid |

**Table name: Delivery**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| delivery\_id | order\_id | employee\_id | address | delivery\_DateTime | cusPhoneNum | deliveryFee | deliveryStatus |
| 1 | 3 | 5 | 23 Main Street | 2025-04-05 12:30:00 | 016-55550000 | 5 | Delivered |
| 2 | 4 | 8 | 456 Park Avenue | 2025-04-05 13:45:00 | 013-77770000 | 6 | Pending |
| 3 | 7 | 5 | 789 Oak Lane | 2025-04-05 14:35:00 | 012-99990000 | 7 | Pending |
| 4 | 8 | 5 | 87 Elme Street | 2025-04-05 14:00:00 | 011-00001111 | 5 | Delivered |
| 5 | 10 | 8 | 54 apple Drive | 2025-04-05 15:00:00 | 016-66660000 | 5 | Pending |

**Table name: Service**

|  |  |  |
| --- | --- | --- |
| Employee\_id | customer\_id | service\_status |
| 1 | 3 | Completed |
| 1 | 4 | Completed |
| 2 | 8 | Completed |
| 3 | 2 | Completed |
| 3 | 1 | Completed |
| 5 | 5 | Delivered |
| 5 | 9 | Pending |
| 5 | 10 | Delivered |
| 8 | 7 | Completed |
| 8 | 6 | Pending |

**Table name: Order\_Items**

|  |  |  |
| --- | --- | --- |
| product\_id | OrderDetail\_id | orderQuantity |
| 1 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 9 | 3 |
| 2 | 10 | 4 |
| 3 | 3 | 3 |
| 5 | 4 | 2 |
| 6 | 5 | 3 |
| 6 | 6 | 4 |
| 7 | 7 | 2 |
| 8 | 8 | 1 |

**Receipt**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| receipt\_id | order\_id | payment\_id | discount\_id | employee\_id | issueDate | totalAmount | discountApplied | finalAmount |
| 1 | 1 | 1 | 5 | 3 |  | 20 | 8% | 18.4 |
| 2 | 2 | 2 |  | 1 |  | 10 |  | 10 |
| 3 | 3 | 3 | 5 | 5 |  | 19.5 | 8% | 17.94 |
| 4 | 4 | 4 |  | 8 |  | 4 |  | 4 |
| 5 | 5 | 5 | 5 | 3 |  | 18 | 8% | 16.56 |
| 6 | 6 | 6 | 5 | 1 |  | 24 | 8% | 22.08 |
| 7 | 7 | 7 |  | 5 |  | 15 |  | 15 |
| 8 | 8 | 8 |  | 5 |  | 8 |  | 8 |
| 9 | 9 | 9 | 5 | 2 |  | 12 | 8% | 11.04 |
| 10 | 10 | 10 |  | 8 |  | 16 |  | 16 |

**Discount**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| discount\_id | discountName | discountType | discountValue | startDate | endDate |
| 1 | |  | | --- | | New Year Promo |  |  | | --- | |  | | Percentage | 15% | 2025-01-10 | 2025-02-10 |
| 2 | |  | | --- | | Member Special Offer |  |  | | --- | |  | | Percentage | 10% | |  | | --- | | 2025-06-30 |  |  | | --- | |  | | |  | | --- | | 2025-08-20 |  |  | | --- | |  | |
| 3 | |  | | --- | | Flash Sale |  |  | | --- | |  | | Percentage | 20% | 2025-04-30 | 2025-05-05 |
| 4 | |  | | --- | | Bulk Purchase Discount |  |  | | --- | |  | | Fixed Amount | RM5 | |  | | --- | | 2025-12-31 |  |  | | --- | |  | | |  | | --- | | 2026-01-01 |  |  | | --- | |  | |
| 5 | |  | | --- | | Student Discount |  |  | | --- | |  | | Percentage | 8% | 2025-01-01 | 2025-11-01 |

**Supplier**

|  |  |  |
| --- | --- | --- |
| supplier\_id | supplierName | telephoneNum |
| 1 | Alice Tan | 014-1234567 |
| 2 | Liew June | 014-2345678 |
| 3 | Xian Lee | 014-3456789 |
| 4 | Jun You | 014-4567890 |
| 5 | Jun Xian | 014-5678901 |

**Supplier product**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| supplier\_id | product\_id | supply\_price | supply\_date | supply\_quantity |
| 1 | 4 | 1.7 | 1-4-2024 | 700 |
| 2 | 1 | 5 | 1-4-2024 | 500 |
| 2 | 7 | 4 | 1-4-2024 | 450 |
| 2 | 8 | 4.5 | 1-4-2024 | 600 |
| 3 | 2 | 1.5 | 1-4-2024 | 1000 |
| 3 | 6 | 3 | 1-4-2024 | 700 |
| 4 | 3 | 2.5 | 1-4-2024 | 2500 |
| 4 | 9 | 0.9 | 1-4-2024 | 2500 |
| 5 | 5 | 0.7 | 1-4-2024 | 3000 |

**6. Script**

-- The Foodie POS System

ALTER SESSION SET "\_oracle\_script" = true;

DROP TABLE Service CASCADE CONSTRAINTS;

DROP TABLE Delivery CASCADE CONSTRAINTS;

DROP TABLE Payment CASCADE CONSTRAINTS;

DROP TABLE Order\_Items CASCADE CONSTRAINTS;

DROP TABLE Order\_Detail CASCADE CONSTRAINTS;

DROP TABLE Orders CASCADE CONSTRAINTS;

DROP TABLE Supplier\_Product CASCADE CONSTRAINTS;

DROP TABLE Supplier CASCADE CONSTRAINTS;

DROP TABLE Discount CASCADE CONSTRAINTS;

DROP TABLE Product CASCADE CONSTRAINTS;

DROP TABLE Employee CASCADE CONSTRAINTS;

DROP TABLE Customer CASCADE CONSTRAINTS;

DROP TABLE Receipt CASCADE CONSTRAINTS;

-----------drop sequence

DROP SEQUENCE customer\_seq;

DROP SEQUENCE employee\_seq;

DROP SEQUENCE order\_seq;

DROP SEQUENCE order\_detail\_seq;

DROP SEQUENCE payment\_seq;

DROP SEQUENCE product\_seq;

DROP SEQUENCE receipt\_seq;

DROP SEQUENCE supplier\_seq;

DROP SEQUENCE delivery\_seq;

DROP SEQUENCE discount\_seq;

-- Drop users if exist

DROP USER customer\_user CASCADE;

DROP USER employee\_user CASCADE;

DROP USER supplier\_user CASCADE;

---------Drop role

DROP ROLE customer\_role;

DROP ROLE employee\_role;

DROP ROLE supplier\_role;

-- Create users and assign roles

CREATE USER customer\_user IDENTIFIED BY Customer123;

CREATE USER employee\_user IDENTIFIED BY Employee123;

CREATE USER supplier\_user IDENTIFIED BY Supplier123;

-- Create roles

CREATE ROLE customer\_role;

CREATE ROLE employee\_role;

CREATE ROLE supplier\_role;

-- Create Tables

-- ========================

-- Customer Table

CREATE TABLE Customer (

    customer\_id NUMBER PRIMARY KEY,

    cusName VARCHAR2(100) NOT NULL,

    cusPhoneNumber VARCHAR2(20),

    email VARCHAR2(100) UNIQUE

);

-- Employee Table

CREATE TABLE Employee (

    employee\_id NUMBER PRIMARY KEY,

    supervisor\_id NUMBER,

    empName VARCHAR2(100) NOT NULL,

    role VARCHAR2(50),

    FOREIGN KEY (supervisor\_id) REFERENCES Employee(employee\_id)

);

-- Product Table

CREATE TABLE Product (

    product\_id NUMBER PRIMARY KEY,

    productName VARCHAR2(100) NOT NULL,

    stockQuantity NUMBER DEFAULT 0,

    price NUMBER(10,2) NOT NULL

);

-- Orders Table

CREATE TABLE Orders (

    order\_id NUMBER PRIMARY KEY,

    customer\_id NUMBER NOT NULL,

    employee\_id NUMBER NOT NULL,

    orderType VARCHAR2(50),

    order\_DateTime TIMESTAMP,

    orderStatus VARCHAR2(50),

    FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),

    FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id)

);

-- Order\_Detail Table

CREATE TABLE Order\_Detail (

    orderDetail\_id NUMBER PRIMARY KEY,

    order\_id NUMBER NOT NULL,

    product\_id NUMBER NOT NULL,

    productName VARCHAR2(100),

    price NUMBER(10,2),

    quantity NUMBER,

    subtotal NUMBER(10,2),

    pickupTime TIMESTAMP,

    FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

    FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

);

-- Payment Table

CREATE TABLE Payment (

    payment\_id NUMBER PRIMARY KEY,

    order\_id NUMBER NOT NULL,

    discount\_id NUMBER,

    paymentMethod VARCHAR2(50),

    paymentStatus VARCHAR2(50),

    FOREIGN KEY (order\_id) REFERENCES Orders(order\_id)

    -- discount\_id FK will be added after Discount table

);

-- Delivery Table

CREATE TABLE Delivery (

    delivery\_id NUMBER PRIMARY KEY,

    order\_id NUMBER NOT NULL,

    employee\_id NUMBER NOT NULL,

    address VARCHAR2(255),

    delivery\_DateTime TIMESTAMP,

    cusPhoneNum VARCHAR2(20),

    deliveryFee NUMBER(10,2),

    deliveryStatus VARCHAR2(50),

    FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

    FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id)

);

-- Service Table

CREATE TABLE Service (

    employee\_id NUMBER NOT NULL,

    customer\_id NUMBER NOT NULL,

    service\_status VARCHAR2(50),

    PRIMARY KEY (employee\_id, customer\_id),

    FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id),

    FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id)

);

-- Order\_Items Table

CREATE TABLE Order\_Items (

    product\_id NUMBER NOT NULL,

    orderDetail\_id NUMBER NOT NULL,

    orderQuantity NUMBER,

    PRIMARY KEY (product\_id, orderDetail\_id),

    FOREIGN KEY (product\_id) REFERENCES Product(product\_id),

    FOREIGN KEY (orderDetail\_id) REFERENCES Order\_Detail(orderDetail\_id)

);

-- Discount Table

CREATE TABLE Discount (

    discount\_id NUMBER PRIMARY KEY,

    discountName VARCHAR2(100) NOT NULL,

    discountType VARCHAR2(50),

    discountValue VARCHAR2(20),

    startDate DATE,

    endDate DATE

);

-- Receipt Table

CREATE TABLE Receipt (

    receipt\_id NUMBER PRIMARY KEY,

    order\_id NUMBER NOT NULL,

    payment\_id NUMBER NOT NULL,

    discount\_id NUMBER,

    employee\_id NUMBER NOT NULL,

    issueDate DATE,

    totalAmount NUMBER(10,2),

    discountApplied VARCHAR2(20),

    finalAmount NUMBER(10,2),

    FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

    FOREIGN KEY (payment\_id) REFERENCES Payment(payment\_id),

    FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id),

    FOREIGN KEY (discount\_id) REFERENCES Discount(discount\_id)

);

-- Supplier Table

CREATE TABLE Supplier (

    supplier\_id NUMBER PRIMARY KEY,

    supplierName VARCHAR2(100) NOT NULL,

    telephoneNum VARCHAR2(20)

);

-- Supplier\_Product Table

CREATE TABLE Supplier\_Product (

    supplier\_id NUMBER NOT NULL,

    product\_id NUMBER NOT NULL,

    supply\_price NUMBER(10,2),

    supply\_date DATE,

    supply\_quantity NUMBER,

    PRIMARY KEY (supplier\_id, product\_id, supply\_date),

    FOREIGN KEY (supplier\_id) REFERENCES Supplier(supplier\_id),

    FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

);

-- Add Discount Foreign Key in Payment (after Discount is created)

ALTER TABLE Payment ADD (

    FOREIGN KEY (discount\_id) REFERENCES Discount(discount\_id)

);

-- Create sequences for primary keys

CREATE SEQUENCE customer\_seq

START WITH 11

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE employee\_seq

START WITH 9

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE product\_seq

START WITH 10

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE order\_seq

START WITH 11

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE order\_detail\_seq

START WITH 11

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE payment\_seq

START WITH 11

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE delivery\_seq

START WITH 6

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE discount\_seq

START WITH 6

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE receipt\_seq

START WITH 11

INCREMENT BY 1

CACHE 10

NOCYCLE;

CREATE SEQUENCE supplier\_seq

START WITH 6

INCREMENT BY 1

CACHE 10

NOCYCLE;

-- ============================

-- Insert Data into Customer

-- ============================

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (1, 'Anthony Davis', '011-11110000', 'davis@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (2, 'Lebron James', '012-22220000', 'james@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (3, 'Kobe Bryant', '012-33330000', 'bryant@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (4, 'Paul George', '013-44440000', 'george@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (5, 'Stephen Curry', '016-55550000', 'curry@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (6, 'Ja Morant', '016-66660000', 'morant@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (7, 'Micheal Jordan', '013-77770000', 'jordan@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (8, 'Klay Thompson', '012-88880000', 'thompson@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (9, 'Kevin Durant', '012-99990000', 'durant@gmail.com');

INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email) VALUES (10, 'Allen Iverson', '011-00001111', 'iverson@gmail.com');

-- ============================

-- Insert Data into Employee

-- ============================

-- Insert Supervisor (employee\_id = 4)

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (4, NULL, 'Adams', 'Supervisor');

-- Insert other Employees that refer to supervisor\_id = 4

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (1, 4, 'Magic Johnson', 'Cashier');

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (2, 4, 'Cameron Anthony', 'Manager');

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (3, 4, 'Shaque Oneal', 'Service');

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (5, 4, 'Dwyane Wade', 'Delivery');

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (6, 4, 'Peter Black', 'Chef');

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (7, 4, 'Sarah Lee', 'Chef');

INSERT INTO Employee (employee\_id, supervisor\_id, empName, role) VALUES (8, 4, 'John Hill', 'Delivery');

-- ============================

-- Insert Data into Product

-- ============================

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (1, 'Cheeseburger', 100, 10);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (2, 'Fried Chicken', 80, 4);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (3, 'French Fries', 150, 6.5);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (4, 'Soft Drink', 200, 3.5);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (5, 'Ice Cream', 60, 2);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (6, 'Chicken Nuggets', 70, 6);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (7, 'Fish Burger', 80, 7.5);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (8, 'Beef Burger', 40, 8);

INSERT INTO Product (product\_id, productName, stockQuantity, price) VALUES (9, 'Onion Rings', 120, 2.5);

-- ============================

-- Insert Data into Discount

-- ============================

INSERT INTO Discount (discount\_id, discountName, discountType, discountValue, startDate, endDate)

VALUES (1, 'New Year Promo', 'Percentage', '15%', TO\_DATE('2025-01-10', 'YYYY-MM-DD'), TO\_DATE('2025-02-10', 'YYYY-MM-DD'));

INSERT INTO Discount (discount\_id, discountName, discountType, discountValue, startDate, endDate)

VALUES (2, 'Member Special Offer', 'Percentage', '10%', TO\_DATE('2025-06-30', 'YYYY-MM-DD'), TO\_DATE('2025-08-20', 'YYYY-MM-DD'));

INSERT INTO Discount (discount\_id, discountName, discountType, discountValue, startDate, endDate)

VALUES (3, 'Flash Sale', 'Percentage', '20%', TO\_DATE('2025-04-30', 'YYYY-MM-DD'), TO\_DATE('2025-05-05', 'YYYY-MM-DD'));

INSERT INTO Discount (discount\_id, discountName, discountType, discountValue, startDate, endDate)

VALUES (4, 'Bulk Purchase Discount', 'Fixed Amount', 'RM5', TO\_DATE('2025-04-04', 'YYYY-MM-DD'), TO\_DATE('2026-04-06', 'YYYY-MM-DD'));

INSERT INTO Discount (discount\_id, discountName, discountType, discountValue, startDate, endDate)

VALUES (5, 'Student Discount', 'Percentage', '8%', TO\_DATE('2025-01-01', 'YYYY-MM-DD'), TO\_DATE('2025-11-01', 'YYYY-MM-DD'));

-- ============================

-- Insert Data into Supplier

-- ============================

INSERT INTO Supplier (supplier\_id, supplierName, telephoneNum) VALUES (1, 'Alice Tan', '014-1234567');

INSERT INTO Supplier (supplier\_id, supplierName, telephoneNum) VALUES (2, 'Liew June', '014-2345678');

INSERT INTO Supplier (supplier\_id, supplierName, telephoneNum) VALUES (3, 'Xian Lee', '014-3456789');

INSERT INTO Supplier (supplier\_id, supplierName, telephoneNum) VALUES (4, 'Jun You', '014-4567890');

INSERT INTO Supplier (supplier\_id, supplierName, telephoneNum) VALUES (5, 'Jun Xian', '014-5678901');

-- ============================

-- Insert Data into Orders

-- ============================

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (1, 2, 3, 'Dine-In', TO\_TIMESTAMP('2025-04-05 11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Completed');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (2, 3, 1, 'Takeaway', TO\_TIMESTAMP('2025-04-05 11:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'Completed');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (3, 5, 5, 'Delivery', TO\_TIMESTAMP('2025-04-05 12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Delivered');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (4, 7, 8, 'Delivery', TO\_TIMESTAMP('2025-04-05 12:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'Delivered');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (5, 1, 3, 'Dine-In', TO\_TIMESTAMP('2025-04-05 12:45:00', 'YYYY-MM-DD HH24:MI:SS'), 'Completed');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (6, 4, 1, 'Pickup', TO\_TIMESTAMP('2025-04-05 13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Completed');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (7, 9, 5, 'Delivery', TO\_TIMESTAMP('2025-04-05 13:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'Pending');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (8, 10, 5, 'Delivery', TO\_TIMESTAMP('2025-04-05 13:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'Delivered');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (9, 8, 2, 'Takeaway', TO\_TIMESTAMP('2025-04-05 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Completed');

INSERT INTO Orders (order\_id, customer\_id, employee\_id, orderType, order\_DateTime, orderStatus)

VALUES (10, 6, 8, 'Delivery', TO\_TIMESTAMP('2025-04-05 14:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'Pending');

-- ============================

-- Insert Data into Order\_Detail

-- ============================

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (1, 1, 1, 'Cheeseburger', 10, 2, 20, NULL);

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (2, 2, 1, 'Cheeseburger', 10, 1, 10, TO\_TIMESTAMP('2025-04-05 11:39:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (3, 3, 3, 'French Fries', 6.5, 3, 19.5, NULL);

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (4, 4, 5, 'Ice Cream', 2, 2, 4, NULL);

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (5, 5, 6, 'Chicken Nuggets', 6, 3, 18, NULL);

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (6, 6, 6, 'Chicken Nuggets', 6, 4, 24, TO\_TIMESTAMP('2025-04-05 13:05:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (7, 7, 7, 'Fish Burger', 7.5, 2, 15, NULL);

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (8, 8, 8, 'Beef Burger', 8, 1, 8, NULL);

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (9, 9, 2, 'Fried Chicken', 4, 3, 12, TO\_TIMESTAMP('2025-04-05 14:10:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO Order\_Detail (orderDetail\_id, order\_id, product\_id, productName, price, quantity, subtotal, pickupTime)

VALUES (10, 10, 2, 'Fried Chicken', 4, 4, 16, NULL);

-- ============================

-- Insert Data into Payment

-- ============================

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (1, 1, 5, 'Credit Card', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (2, 2, NULL, 'Cash', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (3, 3, 5, 'Online Payment', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (4, 4, NULL, 'Debit Card', 'Pending');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (5, 5, 5, 'Cash', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (6, 6, 5, 'Credit Card', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (7, 7, NULL, 'Online Payment', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (8, 8, NULL, 'Debit Card', 'Paid');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (9, 9, 5, 'Cash', 'Pending');

INSERT INTO Payment (payment\_id, order\_id, discount\_id, paymentMethod, paymentStatus)

VALUES (10, 10, 4, 'Credit Card', 'Paid');

-- ============================

-- Insert Data into Delivery

-- ============================

INSERT INTO Delivery (delivery\_id, order\_id, employee\_id, address, delivery\_DateTime, cusPhoneNum, deliveryFee, deliveryStatus)

VALUES (1, 3, 5, '23 Main Street', TO\_TIMESTAMP('2025-04-05 12:30:00', 'YYYY-MM-DD HH24:MI:SS'), '016-55550000', 5, 'Delivered');

INSERT INTO Delivery (delivery\_id, order\_id, employee\_id, address, delivery\_DateTime, cusPhoneNum, deliveryFee, deliveryStatus)

VALUES (2, 4, 8, '456 Park Avenue', TO\_TIMESTAMP('2025-04-05 13:45:00', 'YYYY-MM-DD HH24:MI:SS'), '013-77770000', 6, 'Pending');

INSERT INTO Delivery (delivery\_id, order\_id, employee\_id, address, delivery\_DateTime, cusPhoneNum, deliveryFee, deliveryStatus)

VALUES (3, 7, 5, '789 Oak Lane', TO\_TIMESTAMP('2025-04-05 14:35:00', 'YYYY-MM-DD HH24:MI:SS'), '012-99990000', 7, 'Pending');

INSERT INTO Delivery (delivery\_id, order\_id, employee\_id, address, delivery\_DateTime, cusPhoneNum, deliveryFee, deliveryStatus)

VALUES (4, 8, 5, '87 Elme Street', TO\_TIMESTAMP('2025-04-05 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), '011-00001111', 5, 'Delivered');

INSERT INTO Delivery (delivery\_id, order\_id, employee\_id, address, delivery\_DateTime, cusPhoneNum, deliveryFee, deliveryStatus)

VALUES (5, 10, 8, '54 Apple Drive', TO\_TIMESTAMP('2025-04-05 15:00:00', 'YYYY-MM-DD HH24:MI:SS'), '016-66660000', 5, 'Pending');

-- ============================

-- Insert Data into Service

-- ============================

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (1, 3, 'Completed');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (1, 4, 'Completed');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (2, 8, 'Completed');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (3, 2, 'Completed');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (3, 1, 'Completed');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (5, 5, 'Delivered');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (5, 9, 'Pending');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (5, 10, 'Delivered');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (8, 7, 'Completed');

INSERT INTO Service (employee\_id, customer\_id, service\_status) VALUES (8, 6, 'Pending');

-- ============================

-- Insert Data into Order\_Items

-- ============================

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (1, 1, 2);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (1, 2, 1);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (2, 9, 3);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (2, 10, 4);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (3, 3, 3);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (5, 4, 2);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (6, 5, 3);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (6, 6, 4);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (7, 7, 2);

INSERT INTO Order\_Items (product\_id, orderDetail\_id, orderQuantity) VALUES (8, 8, 1);

-- ============================

-- Insert Data into Receipt

-- ============================

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (1, 1, 1, 5, 3, NULL, 20, '8%', 18.4);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (2, 2, 2, NULL, 1, NULL, 10, NULL, 10);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (3, 3, 3, 5, 5, NULL, 19.5, '8%', 17.94);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (4, 4, 4, NULL, 8, NULL, 4, NULL, 4);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (5, 5, 5, 5, 3, NULL, 18, '8%', 16.56);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (6, 6, 6, 5, 1, NULL, 24, '8%', 22.08);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (7, 7, 7, NULL, 5, NULL, 15, NULL, 15);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (8, 8, 8, NULL, 5, NULL, 8, NULL, 8);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (9, 9, 9, 5, 2, NULL, 12, '8%', 11.04);

INSERT INTO Receipt (receipt\_id, order\_id, payment\_id, discount\_id, employee\_id, issueDate, totalAmount, discountApplied, finalAmount)

VALUES (10, 10, 10, 4, 8, NULL, 16, NULL, 11);

-- ============================

-- Insert Data into Supplier\_Product

-- ============================

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (1, 4, 1.7, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 700);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (2, 1, 5, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 500);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (2, 7, 4, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 450);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (2, 8, 4.5, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 600);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (3, 2, 1.5, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 1000);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (3, 6, 3, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 700);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (4, 3, 2.5, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 2500);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (4, 9, 0.9, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 2500);

INSERT INTO Supplier\_Product (supplier\_id, product\_id, supply\_price, supply\_date, supply\_quantity)

VALUES (5, 5, 0.7, TO\_DATE('2024-04-01', 'YYYY-MM-DD'), 3000);

----------Grant Privilege

-- CustomerRole: customers usually view products and their own orders

GRANT SELECT ON Product TO customer\_role;

GRANT SELECT, INSERT ON Customer TO customer\_role;

GRANT SELECT, INSERT ON Orders TO customer\_role;

GRANT SELECT ON Order\_Items TO customer\_role;

GRANT SELECT ON Receipt TO customer\_role;

GRANT SELECT ON Payment TO customer\_role;

GRANT SELECT ON Delivery TO customer\_role;

GRANT SELECT ON Discount TO customer\_role;

-- EmployeeRole: employees manage products, orders, and services

GRANT SELECT, INSERT, UPDATE, DELETE ON Product TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Orders TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Order\_Detail TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Order\_Items TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Payment TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Delivery TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Service TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Receipt TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Discount TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Customer TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Supplier TO employee\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Supplier\_Product TO employee\_role;

-- SupplierRole: suppliers manage their products

GRANT SELECT, INSERT, UPDATE, DELETE ON Supplier TO supplier\_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Supplier\_Product TO supplier\_role;

GRANT SELECT ON Product TO supplier\_role;

----------Grant Roles to Users

GRANT customer\_role TO customer\_user;

GRANT employee\_role TO employee\_user;

GRANT supplier\_role TO supplier\_user;

-------------Commit

COMMIT;

**PART 2: (Individual Assessment – 50 marks)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Member |  | 1 | 2 | 3 | 4 |
| **Student Name: (to fill)** |  | **Chai Jie Rong** | **Ching Yin Kean** | **Tan Yi Xuan** | **Thau Zhe Jun** |
| **Student ID; (to fill)** |  | **2207093** | **2207267** | **2204266** | **2206252** |
| Two Queries (10 marks) | 10 |  |  |  |  |
| Two Stored Procedure (10 marks) | 10 |  |  |  |  |
| Two Functions (10 marks) | 10 |  |  |  |  |
| Presentation (20 marks) | 20 |  |  |  |  |
| Total Individual Assessment (50 marks) |  |  |  |  |  |
| Group Assessment (50 marls) |  |  |  |  |  |
| **Total marks (100 marks)** |  |  |  |  |  |

**Member 1: Chai Jie Rong**

/\*

INDIVIDUAL ASSIGNMENT SUBMISSION

STUDENT NAME : Chai Jie Rong

STUDENT ID : 2207093

GROUP NUMBER : G087

PROGRAMME : IA

Submission date and time: 29-04-25

\*/

--Query1

SELECT

    sp.supplier\_id,

    s.supplierName,

    sp.product\_id,

    p.productName,

    sp.supply\_price,

    sp.supply\_quantity,

    sp.supply\_date

FROM Supplier\_Product sp

JOIN Supplier s ON sp.supplier\_id = s.supplier\_id

JOIN Product p ON sp.product\_id = p.product\_id

ORDER BY sp.supply\_date DESC;

--Query2

SELECT

    s.supplierName,

    p.productName,

    sp.supply\_quantity,

    sp.supply\_date

FROM Supplier\_Product sp

JOIN Supplier s ON sp.supplier\_id = s.supplier\_id

JOIN Product p ON sp.product\_id = p.product\_id

WHERE s.supplierName = 'Alice Tan'

ORDER BY sp.supply\_date DESC;

--Procedure1

SET SERVEROUTPUT ON

-- add

BEGIN

    add\_supplier\_with\_product(

        'Zhe Jun',

        '012-3456789',

        1,

        5.00,

        TO\_DATE('2025-05-01', 'YYYY-MM-DD'),

        100

    );

END;

/

--check procedure 1

SELECT \* FROM Supplier ORDER BY supplier\_id DESC;

SELECT \* FROM Supplier\_Product ORDER BY supply\_date DESC;

--Procedure2

--add

BEGIN

    update\_supplier\_contact(

        1,

        '018-7654321'

    );

END;

/

--check procedure 2

SELECT \* FROM Supplier WHERE supplier\_id = 1;

--Function1

CREATE OR REPLACE FUNCTION get\_product\_stock(p\_product\_id INT)

RETURN INT

IS

    v\_stock INT;

BEGIN

    SELECT stockQuantity

    INTO v\_stock

    FROM Product

    WHERE product\_id = p\_product\_id;

    RETURN v\_stock;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        RETURN 0;

END;

/

--test fucntion 1

SELECT get\_product\_stock(1) FROM dual;

--Function2

CREATE OR REPLACE FUNCTION get\_total\_supplied\_quantity(p\_product\_id INT)

RETURN NUMBER

IS

    v\_total\_supplied NUMBER;

BEGIN

    SELECT COALESCE(SUM(supply\_quantity), 0)

    INTO v\_total\_supplied

    FROM Supplier\_Product

    WHERE product\_id = p\_product\_id;

    RETURN v\_total\_supplied;

END;

/

--test function 2

SELECT get\_total\_supplied\_quantity(1) FROM dual;

**Member2 : Ching Yin Kean**

/\*

Format: G087\_ChingYinKean.sql

INDIVIDUAL ASSIGNMENT SUBMISSION

STUDENT NAME : Ching Yin Kean

STUDENT ID : 2207267

GROUP NUMBER : G087

PROGRAMME : IA

Submission date and time: 29-4-25

\*/

/\* Query 1: Show all orders with customer info and total price \*/

SELECT o.order\_id, c.cusName AS "Customer Name", o.orderType, o.orderStatus, 'RM'||TO\_CHAR(SUM(od.subtotal),'999.99') AS "Total Price"

FROM Orders o

JOIN Customer c ON o.customer\_id = c.customer\_id

JOIN Order\_Detail od ON o.order\_id = od.order\_id

GROUP BY o.order\_id, c.cusName, o.orderType, o.orderStatus;

/\* Query 2: Show all products with stock quantity and price \*/

SELECT productName, stockQuantity, '  RM'||TO\_CHAR(price,'999.99') AS "Price"

FROM Product;

/\* Stored Procedure 1: Add a new customer after checking email and phone uniqueness \*/

CREATE OR REPLACE PROCEDURE add\_new\_customer(

    p\_customer\_name VARCHAR2,

    p\_phone\_number VARCHAR2,

    p\_email VARCHAR2)

IS

    v\_count NUMBER;

BEGIN

    SELECT COUNT(\*) INTO v\_count

    FROM Customer

    WHERE email = p\_email;

    IF v\_count > 0 THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Email already exists.');

    END IF;

    SELECT COUNT(\*) INTO v\_count

    FROM Customer

    WHERE cusPhoneNumber = p\_phone\_number;

    IF v\_count > 0 THEN

        RAISE\_APPLICATION\_ERROR(-20002, 'Phone number already exists.');

    END IF;

    INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email)

    VALUES (customer\_seq.NEXTVAL, p\_customer\_name, p\_phone\_number, p\_email);

END add\_new\_customer;

/

/\* Stored Procedure 2: Update the status of an order \*/

CREATE OR REPLACE PROCEDURE update\_order\_status(

    p\_order\_id NUMBER,

    p\_new\_status VARCHAR2)

IS

BEGIN

    UPDATE Orders

    SET orderStatus = p\_new\_status

    WHERE order\_id = p\_order\_id;

    COMMIT;

END update\_order\_status;

/

/\* Function 1: Calculate total amount spent by a customer \*/

CREATE OR REPLACE FUNCTION calculate\_total\_spent(

    p\_customer\_id NUMBER)

RETURN NUMBER

IS

    v\_total\_spent NUMBER(10,2);

BEGIN

    SELECT SUM(od.subtotal)

    INTO v\_total\_spent

    FROM Order\_Detail od

    JOIN Orders o ON od.order\_id = o.order\_id

    WHERE o.customer\_id = p\_customer\_id;

    RETURN NVL(v\_total\_spent, 0);

END calculate\_total\_spent;

/

/\* Function 2: Check if a product is in stock \*/

CREATE OR REPLACE FUNCTION is\_product\_available(

    p\_product\_id NUMBER)

RETURN VARCHAR2

IS

    v\_stock\_quantity NUMBER;

BEGIN

    SELECT stockQuantity

    INTO v\_stock\_quantity

    FROM Product

    WHERE product\_id = p\_product\_id;

    IF v\_stock\_quantity > 0 THEN

        RETURN 'In Stock';

    ELSE

        RETURN 'Out of Stock';

    END IF;

END is\_product\_available;

/

--------------------------------------------------------------------------------

-- DEMONSTRATION SECTION (Run Procedures / Functions)

--------------------------------------------------------------------------------

/\* Example Usage for add\_new\_customer \*/

EXEC add\_new\_customer('Jason Lee', '018-77778888', 'jasonlee@gmail.com');

/\* Check if customer was added \*/

SELECT \* FROM Customer;

/\* Example Usage for update\_order\_status \*/

EXEC update\_order\_status(2, 'Completed');

/\* Check if order status was updated \*/

SELECT order\_id, orderStatus

FROM Orders;

/\* Example Usage for calculate\_total\_spent function \*/

SELECT customer\_id, 'RM' || TO\_CHAR(calculate\_total\_spent(customer\_id), '999.99') AS "Total Spent"

FROM Customer;

/\* Example Usage for is\_product\_available function \*/

SELECT product\_id, productName, is\_product\_available(product\_id) AS "Availability"

FROM Product;

**Member3: Tan Yi Xuan**

/\*

Format: G999\_MemberName.sql

INDIVIDUAL ASSIGNMENT SUBMISSION

STUDENT NAME : TAN YI XUAN

STUDENT ID : 22ACB04266

GROUP NUMBER : G087

PROGRAMME : IA

Submission date and time: 29-04-25

SQL script to be submtted by each member, click save as "G999\_MemberName.sql" e.g. G001\_SmithWhite.sql

\*/

-- This script contains 2 Queries, 2 Procedures, and 2 Functions

---------------------------------------------------------------------------Pre-Setup---------------------------------------------------------------------------------------------------

SET SERVEROUTPUT ON

-- ===================================================================================================================================================================================

--

--                                     1. Queries

--

-- ===================================================================================================================================================================================

-----------------------------------------------------------------------------Query 1---------------------------------------------------------------------------------------------------

The query retrieves receipt details for orders placed by customers on a specific date, along with the customer name, order ID, receipt ID, order date time, total amount, discount applied, and the final amount after the discount.

SELECT

    r.receipt\_id,

    r.order\_id,

    c.cusName AS customer\_name,

    o.order\_DateTime,

    r.totalAmount,

    r.discountApplied,

    r.finalAmount

FROM Receipt r

JOIN orders o ON r.order\_id = o.order\_id

JOIN Customer c ON o.customer\_id = c.customer\_id

WHERE o.order\_DateTime >= TO\_DATE('05-04-25', 'DD-MM-YY')

  AND o.order\_DateTime < TO\_DATE('06-04-25', 'DD-MM-YY')

ORDER BY r.receipt\_id;

------------------------------------------------------------------------------------Query 2-----------------------------------------------------------------------------------------------

This query retrieves a list of employees who are directly supervised by a specific supervisor, along with their employee details (employee ID, name, role, and supervisor ID). Display the employee who supervisor name is Adams.

SELECT

    e.employee\_id,

    e.empName,

    e.role,

    e.supervisor\_id

FROM Employee e

JOIN Employee s ON e.supervisor\_id = s.employee\_id

WHERE UPPER(s.empName) = UPPER('Adams')

ORDER BY e.empName;

-- ========================================================================================================================================================================================

--

--                                     2. PROCEDURES

-- ===========================================================================================================================================================================================

------------------------------------------------------------------------Procedure 1---------------------------------------------------------------------------------------------------------

This procedure allows employees to insert a new discount into the Discount table. Before inserting, the system checks whether a discount with the same name already exists. If no existing discount with the same name is found, the new discount is inserted; otherwise, a message is displayed indicating that the discount name already exists.

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE insert\_discount(

    p\_discountName VARCHAR2,

    p\_discountType VARCHAR2,

    p\_discountValue VARCHAR2,

    p\_startDate DATE,

    p\_endDate DATE

)

IS

    v\_count INT;

BEGIN

    SELECT COUNT(\*)

    INTO v\_count

    FROM Discount

    WHERE discountName = p\_discountName;

    COMMIT;

    IF v\_count = 0 THEN

        INSERT INTO Discount (discount\_id, discountName, discountType, discountValue, startDate, endDate)

        VALUES (discount\_seq.NEXTVAL, p\_discountName, p\_discountType, p\_discountValue, p\_startDate, p\_endDate);

        DBMS\_OUTPUT.PUT\_LINE('Discount inserted successfully.');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Discount with this name already exists.');

    END IF;

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error inserting discount promo.');

END;

/

---------------------------------------------------------Demo command to insert a new discount into the discount table--------------------------------------------------------------------

EXECUTE insert\_discount('Hari Raya Promotion', 'Fixed Amount', 'RM10', TO\_DATE('2025-03-10', 'YYYY-MM-DD'), TO\_DATE('2025-06-10', 'YYYY-MM-DD'));

SELECT \* FROM discount;

------------------------------------------Demo command to show an error when insert an existing discount into the discount table---------------------------------------------------------

EXECUTE insert\_discount('New Year Promo', 'Fixed Amount', 'RM10', TO\_DATE('2025-03-10', 'YYYY-MM-DD'), TO\_DATE('2025-06-10', 'YYYY-MM-DD'));

SELECT \* FROM discount;

----------------------------------------------------------------------------------Procedure 2-------------------------------------------------------------------------------------------

This procedure designed to update the endDate of a specific discount in the Discount table. So employee can modify the end date if the end date involve any error.

CREATE OR REPLACE PROCEDURE update\_discount\_enddate(

    p\_discount\_id INT,

    p\_new\_endDate DATE

)

IS

    -- Variable to store the current end date of the discount.

    v\_current\_endDate DATE;

BEGIN

    -- Validate if the new end date is valid (cannot be in the past).

    IF p\_new\_endDate < SYSDATE THEN

        DBMS\_OUTPUT.PUT\_LINE('Error: The new end date cannot be in the past.');

        RETURN;

    END IF;

    BEGIN

        -- Fetch the current end date.

        SELECT endDate INTO v\_current\_endDate

        FROM Discount

        WHERE discount\_id = p\_discount\_id;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            DBMS\_OUTPUT.PUT\_LINE('Error: Discount ID not found.');

            RETURN;

    END;

    -- Update the discount's end date.

    BEGIN

        UPDATE Discount

        SET endDate = p\_new\_endDate

        WHERE discount\_id = p\_discount\_id;

        COMMIT;

        DBMS\_OUTPUT.PUT\_LINE('Success: Discount end date updated successfully.');

    EXCEPTION

        WHEN OTHERS THEN

            ROLLBACK;

            DBMS\_OUTPUT.PUT\_LINE('Error during update: ' || SQLERRM);

    END;

END;

/

------------------------------------------------- Demo command to update a new discount end date into the discount table ---------------------------------------------------------------

EXECUTE update\_discount\_enddate(5, TO\_DATE('2025-12-31', 'YYYY-MM-DD'));

SELECT \* FROM discount;

------------------------------------- Demo command for error when employee update a new discount end date that has passed ---------------------------------------------------------------

EXECUTE update\_discount\_enddate(5, TO\_DATE('2025-3-31', 'YYYY-MM-DD'));

SELECT \* FROM discount;

-- =======================================================================================================================================================================================

--

--                                      3. FUNCTIONS

--

-- =======================================================================================================================================================================================

-------------------------------------------------------------------------------------Function 1--------------------------------------------------------------------------------------------

This function is designed to compute the discount amount for a specific order, identified by its order\_id. It retrieves the discount type and value associated with the order and calculates the discount accordingly.​

CREATE OR REPLACE FUNCTION calculate\_discount\_amount(p\_order\_id INT)

RETURN NUMBER

IS

    v\_discount\_type Discount.discountType%TYPE;

    v\_discount\_value Discount.discountValue%TYPE;

    v\_total\_amount NUMBER;

    v\_discount\_amount NUMBER := 0;

BEGIN

    SELECT d.discountType, d.discountValue, r.totalAmount

    INTO v\_discount\_type, v\_discount\_value, v\_total\_amount

    FROM Receipt r

    LEFT JOIN Discount d ON r.discount\_id = d.discount\_id

    WHERE r.order\_id = p\_order\_id;

    IF v\_discount\_type IS NULL THEN

        DBMS\_OUTPUT.PUT\_LINE('No Discount was Used for Order ' || p\_order\_id || '.');

        RETURN 0;

    ELSIF v\_discount\_type = 'Percentage' THEN

        v\_discount\_amount := v\_total\_amount \* TO\_NUMBER(REPLACE(v\_discount\_value, '%', '')) / 100;

    ELSE

        v\_discount\_amount := TO\_NUMBER(REPLACE(v\_discount\_value, 'RM', ''));

    END IF;

    DBMS\_OUTPUT.PUT\_LINE('The Discount Amount for Order ' || p\_order\_id || ' is RM' || v\_discount\_amount);

    RETURN v\_discount\_amount;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Please Enter A Valid Order ID. Order ID ' || p\_order\_id || ' is invalid.');

        RETURN 0;

END;

/

---------------------------------------Demo for calculate the discount amount that using the percentage as its discount type---------------------------------------------------------

SELECT calculate\_discount\_amount(1) AS discount\_amount FROM dual;

----------------------------------------Demo for calculate the discount amount that use fixed amount as its discount type--------------------------------------------------------------

SELECT calculate\_discount\_amount(10) AS discount\_amount FROM dual;

------------------------------------------------------Demo for display if the order do not used any discount--------------------------------------------------------------------------

SELECT calculate\_discount\_amount(2) AS discount\_amount FROM dual;

---------------------------------------Demo command for calculate the discount amount when enter an invalid order id-------------------------------------------------------------------

SELECT calculate\_discount\_amount(15) AS discount\_amount FROM dual；

---------------------------------------------------------------------Function 2--------------------------------------------------------------------------------------------------------

This function is to list out all the discount based on the categories active, expired and upcoming promotions according the  current date time. The promotion name and the total promotion of that category will be display after run the following command, so it do not in the enter other command to get the output.

SET SERVEROUTPUT ON;

DECLARE

    -- Cursor for active promotions

    CURSOR active\_promotions\_cursor IS

        SELECT discountName

        FROM Discount

        WHERE SYSDATE BETWEEN startDate AND endDate;

    -- Cursor for expired promotions

    CURSOR expired\_promotions\_cursor IS

        SELECT discountName

        FROM Discount

        WHERE SYSDATE > endDate;

    -- Cursor for upcoming promotions

    CURSOR upcoming\_promotions\_cursor IS

        SELECT discountName

        FROM Discount

        WHERE SYSDATE < startDate;

    -- Variables to hold promotion names

    v\_discountName Discount.discountName%TYPE;

    -- Counters for each category

    v\_active\_count   NUMBER := 0;

    v\_expired\_count  NUMBER := 0;

    v\_upcoming\_count NUMBER := 0;

BEGIN

    -- Process active promotions

    DBMS\_OUTPUT.PUT\_LINE('--- Active Promotions ---');

    OPEN active\_promotions\_cursor;

    LOOP

        FETCH active\_promotions\_cursor INTO v\_discountName;

        EXIT WHEN active\_promotions\_cursor%NOTFOUND;

        v\_active\_count := v\_active\_count + 1;

        DBMS\_OUTPUT.PUT\_LINE('Promotion Name: ' || v\_discountName);

    END LOOP;

    CLOSE active\_promotions\_cursor;

    DBMS\_OUTPUT.PUT\_LINE('Total Active Promotions: ' || v\_active\_count);

    DBMS\_OUTPUT.PUT\_LINE(''); -- Blank line for separation

    -- Process expired promotions

    DBMS\_OUTPUT.PUT\_LINE('--- Expired Promotions ---');

    OPEN expired\_promotions\_cursor;

    LOOP

        FETCH expired\_promotions\_cursor INTO v\_discountName;

        EXIT WHEN expired\_promotions\_cursor%NOTFOUND;

        v\_expired\_count := v\_expired\_count + 1;

        DBMS\_OUTPUT.PUT\_LINE('Promotion Name: ' || v\_discountName);

    END LOOP;

    CLOSE expired\_promotions\_cursor;

    DBMS\_OUTPUT.PUT\_LINE('Total Expired Promotions: ' || v\_expired\_count);

    DBMS\_OUTPUT.PUT\_LINE(''); -- Blank line for separation

    -- Process upcoming promotions

    DBMS\_OUTPUT.PUT\_LINE('--- Upcoming Promotions ---');

    OPEN upcoming\_promotions\_cursor;

    LOOP

        FETCH upcoming\_promotions\_cursor INTO v\_discountName;

        EXIT WHEN upcoming\_promotions\_cursor%NOTFOUND;

        v\_upcoming\_count := v\_upcoming\_count + 1;

        DBMS\_OUTPUT.PUT\_LINE('Promotion Name: ' || v\_discountName);

    END LOOP;

    CLOSE upcoming\_promotions\_cursor;

    DBMS\_OUTPUT.PUT\_LINE('Total Upcoming Promotions: ' || v\_upcoming\_count);

END;

/

**Member4: Thau Zhe Jun**

/\*

Format: G087\_ThauZheJun.sql

INDIVIDUAL ASSIGNMENT SUBMISSION

STUDENT NAME : THAU ZHE JUN

STUDENT ID : 22ACB06252

GROUP NUMBER : G087

PROGRAMME : IA

Submission date and time: 29-04-25

\*/

-- This script contains 2 Queries, 2 Procedures, and 2 Functions

Query 1

// The query is to search specify products with specify product name and details about the product

SELECT product\_id, productName, price, stockQuantity

FROM Product

WHERE productName LIKE '%Burger%' OR productName LIKE '%Drink%' OR productName LIKE '%Fries%' OR productName LIKE '%Ring%'

ORDER BY productName;

Query 2

// This query is to view all pending deliveries with the rider assigned

SELECT d.delivery\_id, d.order\_id, e.empName AS rider\_name, d.address, d.delivery\_DateTime, d.deliveryFee, d.deliveryStatus

FROM Delivery d

JOIN Employee e ON d.employee\_id = e.employee\_id

WHERE d.deliveryStatus = 'Pending'

ORDER BY d.delivery\_DateTime;

Procedure 1

// This procedure is to add in new customer account

CREATE OR REPLACE PROCEDURE create\_customer(

    p\_cusName VARCHAR2,

    p\_cusPhone VARCHAR2,

    p\_email VARCHAR2

)

IS

BEGIN

    INSERT INTO Customer (customer\_id, cusName, cusPhoneNumber, email)

    VALUES (customer\_seq.NEXTVAL, p\_cusName, p\_cusPhone, p\_email);

    COMMIT;

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Data already exists.');

END;

/

// to create a new customer:

BEGIN

create\_customer('Alex','0124223322','alex88@gmail.com');

END;

/

Procedure 2

// The procedure is to create a new delivery order and assign to rider

CREATE OR REPLACE PROCEDURE create\_delivery\_order(

    p\_order\_id INT,

    p\_employee\_id INT,

    p\_address VARCHAR2,

    p\_delivery\_datetime DATE,

    p\_cusPhoneNum VARCHAR2,

    p\_deliveryFee NUMBER

)

IS

BEGIN

    INSERT INTO Delivery (

        delivery\_id,

        order\_id,

        employee\_id,

        address,

        delivery\_DateTime,

        cusPhoneNum,

        deliveryFee,

        deliveryStatus

    ) VALUES (

        delivery\_seq.NEXTVAL,

        p\_order\_id,

        p\_employee\_id,

        p\_address,

        p\_delivery\_datetime,

        p\_cusPhoneNum,

        p\_deliveryFee,

        'Pending'

    );

    COMMIT;

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        RAISE\_APPLICATION\_ERROR(-20040, 'Duplicate Delivery ID.');

    WHEN OTHERS THEN

        RAISE\_APPLICATION\_ERROR(-20041, 'Unexpected error occurred while creating delivery order.');

END;

/

//to create new order and assign to rider

 BEGIN

 create\_delivery\_order(6, 5, '101 Sunway Avenue', TO\_DATE('29-APR-2025 10:30','DD-MM-YYYY HH24:MI'),'012-4320000', 5.00);

 END;

 /

Function 1

// This function is to calculate the total sales till now

CREATE OR REPLACE FUNCTION get\_total\_sales\_till\_now

RETURN NUMBER

IS

    v\_total NUMBER := 0;

BEGIN

    SELECT COALESCE(SUM(finalAmount), 0)

    INTO v\_total

    FROM Receipt;  -- No WHERE clause

    RETURN v\_total;

END;

/

//to view the total sales until now

DECLARE

    v\_total\_sales NUMBER;

BEGIN

    v\_total\_sales := get\_total\_sales\_till\_now;

    DBMS\_OUTPUT.PUT\_LINE('Total sales till now: RM' || v\_total\_sales);

END;

/

Function 2

// This function is to check rider's total deliveries today

CREATE OR REPLACE FUNCTION rider\_deliveries\_today(p\_employee\_id INT)

RETURN NUMBER

IS

    v\_total\_deliveries NUMBER;

BEGIN

    SELECT COUNT(\*)

    INTO v\_total\_deliveries

    FROM Delivery

    WHERE employee\_id = p\_employee\_id

      AND TRUNC(delivery\_DateTime) = TRUNC(SYSDATE)

      AND deliveryStatus = 'Delivered';

    RETURN v\_total\_deliveries;

END;

/

// to view the rider's total deliveries today

DECLARE

    v\_count NUMBER;

BEGIN

    v\_count := rider\_deliveries\_today(5);

    DBMS\_OUTPUT.PUT\_LINE('Total deliveries today by rider 5: '|| v\_count);

END;

/