|  |  |
| --- | --- |
| ILS DELIVERABLE 3-Group 24  26 May 2025 | Group Members  R.A Nthathakane 223185940 K.E Ntseno 223021060 T. Tuwe 222062627  Les Maisons café |

Table of Contents

[Les Maisons – Company Narrative and Business Rules 1](#_Toc199191629)

[4 Unique Business Rules 1](#_Toc199191630)

[Les Maisons – ERD Diagram & Assumptions 2](#_Toc199191631)

[Les Maisons – SQL Script & Data 5](#_Toc199191632)

[Query 1 Output: 25](#_Toc199191633)

[Query 2 Output: 26](#_Toc199191634)

[Query 3 Output: 27](#_Toc199191635)

[Query 4 Output: 29](#_Toc199191636)

[Query 5 output: 30](#_Toc199191637)

# Les Maisons – Company Narrative and Business Rules

Les Maisons is a student-focused lifestyle café and cultural lounge that spans multiple Universities in South Africa, with flagship branches in university cities like Johannesburg and Cape Town. The brand is known for its cozy, home-inspired interiors, curated live entertainment, and creative engagement with the student community. Les Maisons serves as more than just a café; it is a dynamic, multi-purpose space designed to support student productivity, self-expression, and socialization. The venue includes a premium coffee bar, exclusive tea selections, and artisanal menu options, including vegan and traditional offerings. Each branch runs a variety of unique weekly events such as Jazz Night, Games Night, Karaoke Night, and Comedy Night. Students use their university cards to access the facilities, and those who frequently visit can benefit from a points-based loyalty program.

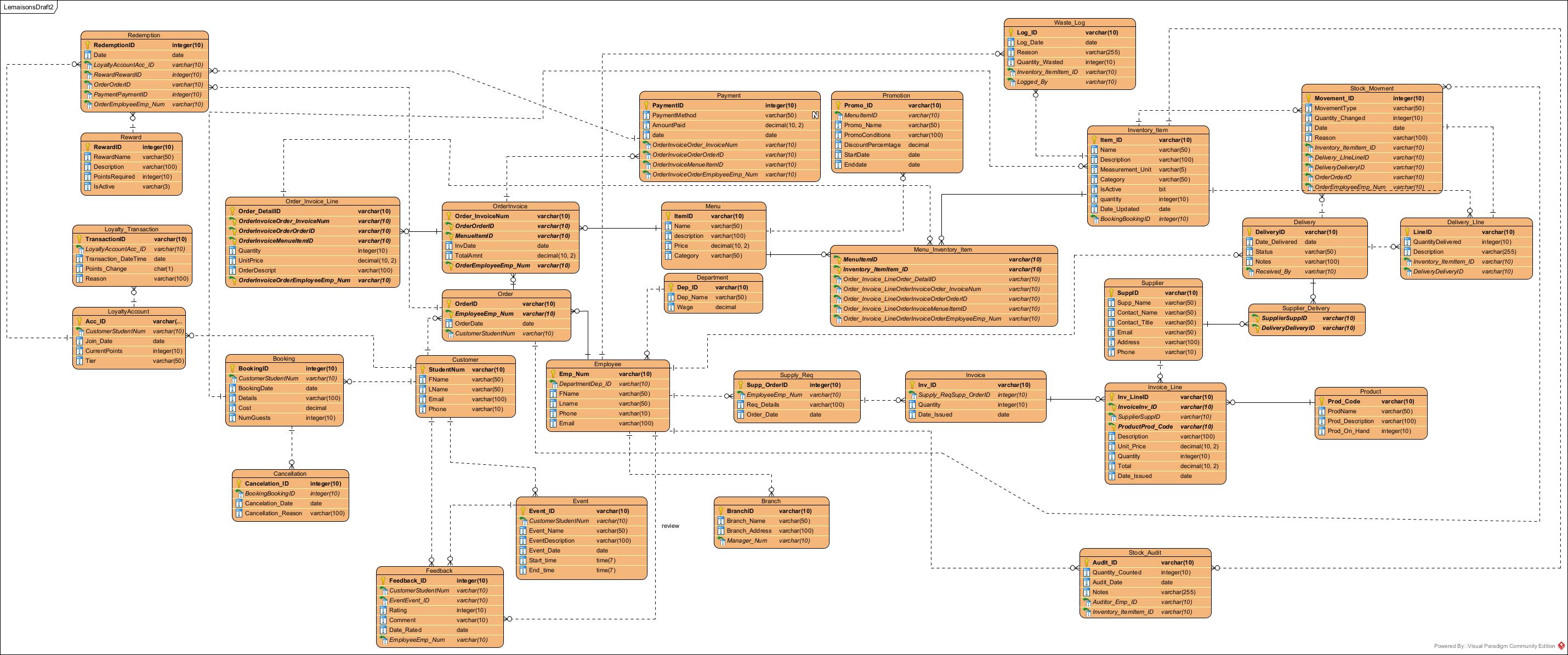
Each branch of Les Maisons must have at least one active manager responsible for overseeing operations, and each branch maintains a unique menu tailored to the preferences of its local clientele. Menu items consist of one or more ingredients, and every recipe used in the preparation of these items must be formally registered and stored within the system to ensure consistency and availability tracking. Orders can only be placed by registered individuals, thereby ensuring that all transactions are traceable and eligible for loyalty rewards. Each supplier must be assigned to supply at least one inventory item to a specific branch. Deliveries are meticulously recorded by shipment and are verified against received inventory to maintain accountability and accuracy.

## 4 Unique Business Rules

1. Loyalty points are calculated based on order history and can be redeemed for menu discounts.
2. Student Card Tap-ins: All students must tap their university student cards at entry points to track foot traffic. Entry data is stored per student per day and linked to branch visits.
3. Study Time Periods: Les Maisons enforces a quiet “Study Time” policy from 08:00 to 12:00, every Monday to Wednesday, where no live events or performances are allowed, and only soft background music is permitted.
4. Customer booking Inventory Checklist: All custom table bookings must include a mandatory inventory checklist, covering essentials such as decorations, tableware, and other rentable assets, ensuring preparedness.

# Les Maisons – ERD Diagram & Assumptions

**ERD DIAGRAM:**



Assumptions for Les Maisons Database Design

1. Branch & Staff

* Each branch represents the physical location of Les Maisons.
* A staff member belongs to exactly one branch.
* A department (e.g., Barista, Manager, Technician) determines responsibilities access levels and wages; one role can be assigned to many staff members.
* Staff can perform multiple duties, but only one primary department is recorded in this system.

2. Orders & Payments

* Customers/Students must be registered to place orders.
* Orders can contain multiple OrderDetails, each linked to specific menu items.
* Payments are tied to orders and can be made via cash, card, or loyalty points.
* Promotions can apply discounts to orders and are linked to specific menu items or events.

3. Menu & Recipes

* Each menu item is prepared using one or more ingredients, defined through the MenuRecipe table.
* Ingredients are categorized using Ingredient Type (e.g., dairy, grain, meat).
* All recipe ingredients must be available in Inventory to mark a menu item as “available.”
* Menu availability may differ per branch, though the base menu is centrally managed.

4. Inventory & Supplies

* Each branch maintains its own Inventory.
* Suppliers fulfill Supply Requests which result in Shipments and generate Invoices upon delivery.
* A Delivery logs the actual delivery of items to a branch and links to a specific shipment.

5. Events & Entertainment

* Les Maisons hosts scheduled events (Jazz Night, Movie Night, Karaoke Night, Comedy Night, Games Night), defined in the Event Schedule.
* Each event takes place at a branch.
* Student Performances are only allowed during designated event types and require approval.

6. Bookings & Tables

* Customers can book tables, especially during high-traffic hours or special events.
* Table availability and capacity are managed per branch and are linked to bookings.
* Table bookings generate inventory checks (e.g., tableware or decoration availability).
* Customers can make table cancellations within 24 hours.

7. Financial Tracking

* Each financial transaction is logged for financial audit and credibility purposes.

8. Loyalty Program

* Students/customers enrolled in the Loyalty Program earn points per order.
* Loyalty points can be redeemed for menu discounts or purchases.
* Loyalty data is tied to student profiles and updated in real time.

9. System-Wide Assumptions

* All students tap their cards upon entry for foot traffic tracking.
* Study hours are predefined and enforced (8 a.m. to 12 p.m. Monday–Wednesday) and limit event scheduling.
* All custom bookings must include a pre-approved inventory checklist for setup (decor, lighting, etc.).

### Les Maisons – SQL Script & Data

drop database if exists Lesmaisons

create database LesMaisons

use lesMaisons

-- Branch

CREATE TABLE Branch (

BranchID VARCHAR(10) PRIMARY KEY,

BranchName VARCHAR(100),

Location VARCHAR(100),

Province VARCHAR(50),

ManagerName VARCHAR(100)

);

INSERT INTO Branch VALUES

('BR001', 'Les Maisons UJ', 'Kingsway Ave, JHB', 'Gauteng', 'Realeboha Nthathakane'),

('BR002', 'Les Maisons CPT', 'Bree St, Cape Town', 'Western Cape', 'Keletso Ntseno'),

('BR003', 'Les Maisons DBN', 'Florida Rd, Durban', 'KwaZulu-Natal', 'Tadi Tuwe'),

('BR004', 'Les Maisons PTA', 'Church St, Pretoria', 'Gauteng', 'Gareth Maradze'),

('BR005', 'Les Maisons NW', 'Dr James Moroka Dr, Mahikeng', 'North West', 'Putsoeli Ntseno');

-- Department

CREATE TABLE Department (

Dep\_Num VARCHAR(10) PRIMARY KEY,

Dep\_Name VARCHAR(50),

Wage DECIMAL(10,2),

BranchID VARCHAR(10),

FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)

);

INSERT INTO Department VALUES

('DEP01', 'Kitchen', 8000.00, 'BR001'),

('DEP02', 'Front Desk', 7000.00, 'BR001'),

('DEP03', 'Marketing', 6000.00, 'BR002'),

('DEP04', 'Maintenance', 6500.00, 'BR003'),

('DEP05', 'Management', 10000.00, 'BR004');

-- Employee

CREATE TABLE Employee (

Emp\_Num VARCHAR(10) PRIMARY KEY,

Dep\_Num VARCHAR(10),

FName VARCHAR(50),

LName VARCHAR(50),

Phone VARCHAR(20),

Email VARCHAR(100),

FOREIGN KEY (Dep\_Num) REFERENCES Department(Dep\_Num)

);

INSERT INTO Employee VALUES

('EMP001', 'DEP01', 'Thobeka', 'Olifant', '0711111111', 'thobeka@lesmaisons.co.za'),

('EMP002', 'DEP02', 'Matthew', 'Olifant', '0722222222', 'matthew@lesmaisons.co.za'),

('EMP003', 'DEP03', 'Oratile', 'Selepe', '0733333333', 'oratile@lesmaisons.co.za'),

('EMP004', 'DEP04', 'Nthatisi', 'Dlamini', '0744444444', 'nthatisi@lesmaisons.co.za'),

('EMP005', 'DEP05', 'Moleboheng', 'Mokoena', '0755555555', 'moleboheng@lesmaisons.co.za');

-- Customer

CREATE TABLE Customer (

StudentNum VARCHAR(10) PRIMARY KEY,

FName VARCHAR(50),

LName VARCHAR(50),

Email VARCHAR(100),

Phone VARCHAR(15)

);

INSERT INTO Customer VALUES

('223185940', 'Josh', 'Zerkaa', 'josh@student.uj.ac.za', '0712345678'),

('223021060', 'Harry', 'Mandela', 'harry@student.uj.ac.za', '0723456789'),

('222062627', 'Karabo', 'Dlamini', 'karabo@student.uj.ac.za', '0734567890'),

('220000001', 'Lebo', 'Mokoena', 'lebo@student.uj.ac.za', '0745678901'),

('220000002', 'Sipho', 'Zulu', 'sipho@student.uj.ac.za', '0756789012');

--Loyalty Account

CREATE TABLE LoyaltyAccount (

Acc\_ID VARCHAR(10) PRIMARY KEY,

CustomerStudentNum VARCHAR(10),

Join\_Date DATE,

CurrentPoints INT,

Tier VARCHAR(50),

FOREIGN KEY (CustomerStudentNum) REFERENCES Customer(StudentNum)

);

INSERT INTO LoyaltyAccount VALUES

('ACC001', '223185940', '2024-01-01', 200, 'Gold'),

('ACC002', '223021060', '2024-01-15', 150, 'Silver'),

('ACC003', '222062627', '2024-02-01', 100, 'Bronze'),

('ACC004', '220000001', '2024-03-01', 250, 'Gold'),

('ACC005', '220000002', '2024-04-01', 50, 'Bronze');

--Loyalty Transaction

CREATE TABLE Loyalty\_Transaction (

TransactionID VARCHAR(10) PRIMARY KEY,

LoyaltyAccountAcc\_ID VARCHAR(10),

Transaction\_DateTime DATE,

Points\_Change INT,

Reason VARCHAR(100),

FOREIGN KEY (LoyaltyAccountAcc\_ID) REFERENCES LoyaltyAccount(Acc\_ID)

);

INSERT INTO Loyalty\_Transaction VALUES

('LT001', 'ACC001', '2024-05-01', 20, 'Order Points'),

('LT002', 'ACC002', '2024-05-02', 15, 'Event Participation'),

('LT003', 'ACC003', '2024-05-03', 10, 'Referral'),

('LT004', 'ACC004', '2024-05-04', -30, 'Reward Redemption'),

('LT005', 'ACC005', '2024-05-05', 5, 'Feedback Bonus');

--Reward

CREATE TABLE Reward (

RewardID INT PRIMARY KEY,

RewardName VARCHAR(50),

Description VARCHAR(100),

PointsRequired INT,

IsActive VARCHAR(3)

);

INSERT INTO Reward VALUES

(1, 'Free Coffee', 'One small café latte', 20, 'Yes'),

(2, 'Merch T-Shirt', 'Custom Les Maisons shirt', 100, 'Yes'),

(3, 'Free Sandwich', 'One vegan sandwich', 50, 'Yes'),

(4, 'Discount Voucher', 'R50 voucher on next meal', 60, 'Yes'),

(5, 'Event Ticket', 'Free entry to one event', 40, 'Yes');

--Redemption

CREATE TABLE Redemption (

RedemptionID INT PRIMARY KEY,

Date DATE,

RewardRewardID INT,

LoyaltyAccountAcc\_ID VARCHAR(10),

Emp\_Num VARCHAR(10),

FOREIGN KEY (RewardRewardID) REFERENCES Reward(RewardID),

FOREIGN KEY (LoyaltyAccountAcc\_ID) REFERENCES LoyaltyAccount(Acc\_ID),

FOREIGN KEY (Emp\_Num) REFERENCES Employee(Emp\_Num)

);

INSERT INTO Redemption VALUES

(1, '2024-05-01', 1, 'ACC001', 'EMP001'),

(2, '2024-05-02', 2, 'ACC002', 'EMP002'),

(3, '2024-05-03', 3, 'ACC003', 'EMP003'),

(4, '2024-05-04', 4, 'ACC004', 'EMP004'),

(5, '2024-05-05', 5, 'ACC005', 'EMP005');

--Order

CREATE TABLE [Order] (

OrderID VARCHAR(10) PRIMARY KEY,

Emp\_Num VARCHAR(10),

OrderDate DATE,

CustomerStudentNum VARCHAR(10),

FOREIGN KEY (Emp\_Num) REFERENCES Employee(Emp\_Num),

FOREIGN KEY (CustomerStudentNum) REFERENCES Customer(StudentNum)

);

INSERT INTO [Order] VALUES

('O001', 'EMP001', '2024-05-01', '223185940'),

('O002', 'EMP002', '2024-05-02', '223021060'),

('O003', 'EMP003', '2024-05-03', '222062627'),

('O004', 'EMP004', '2024-05-04', '220000001'),

('O005', 'EMP005', '2024-05-05', '220000002');

--Order Invoice Line

CREATE TABLE Order\_Invoice\_Line (

OrderInvoiceNum VARCHAR(10) PRIMARY KEY,

OrderID VARCHAR(10),

MenuItemID VARCHAR(10),

Qty INT,

UnitPrice DECIMAL(10,2),

OrderDesc VARCHAR(255),

Emp\_Num VARCHAR(10),

FOREIGN KEY (OrderID) REFERENCES [Order](OrderID),

FOREIGN KEY (MenuItemID) REFERENCES Menu(MenuItemID),

FOREIGN KEY (Emp\_Num) REFERENCES Employee(Emp\_Num)

);

INSERT INTO Order\_Invoice\_Line VALUES

('INV001', 'O001', 'M001', 2, 35.00, '2 Café Lattes', 'EMP001'),

('INV002', 'O002', 'M002', 1, 50.00, '1 Vegan Sandwich', 'EMP002'),

('INV003', 'O003', 'M003', 3, 25.00, '3 Muffins', 'EMP003'),

('INV004', 'O004', 'M004', 1, 30.00, '1 Chai Latte', 'EMP004'),

('INV005', 'O005', 'M005', 2, 45.00, '2 Quiches', 'EMP005');

--Order Invoice

CREATE TABLE Order\_Invoice (

InvoiceID VARCHAR(10) PRIMARY KEY,

OrderID VARCHAR(10),

SubTotal DECIMAL(10,2),

Discount DECIMAL(10,2),

Total DECIMAL(10,2),

FOREIGN KEY (OrderID) REFERENCES [Order](OrderID)

);

INSERT INTO Order\_Invoice VALUES

('IV001', 'O001', 70.00, 0.00, 70.00),

('IV002', 'O002', 50.00, 5.00, 45.00),

('IV003', 'O003', 75.00, 0.00, 75.00),

('IV004', 'O004', 30.00, 0.00, 30.00),

('IV005', 'O005', 90.00, 10.00, 80.00);

--Inventory Item

CREATE TABLE Inventory\_Item (

InventoryID VARCHAR(10) PRIMARY KEY,

ItemName VARCHAR(100),

ItemType VARCHAR(50),

QuantityOnHand INT,

UnitCost DECIMAL(10,2),

BranchID VARCHAR(10),

FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)

);

INSERT INTO Inventory\_Item VALUES

('I001', 'Milk', 'Dairy', 100, 15.00, 'BR001'),

('I002', 'Bread', 'Bakery', 80, 8.00, 'BR001'),

('I003', 'Chocolate Chips', 'Bakery', 120, 12.00, 'BR002'),

('I004', 'Chai Spice', 'Spices', 90, 6.50, 'BR003'),

('I005', 'Eggs', 'Dairy', 150, 5.00, 'BR004');

--Menu

CREATE TABLE Menu (

MenuItemID VARCHAR(10) PRIMARY KEY,

Name VARCHAR(100),

Description VARCHAR(255),

Price DECIMAL(10,2),

Category VARCHAR(50)

);

INSERT INTO Menu VALUES

('M001', 'Café Latte', 'Hot espresso with milk', 35.00, 'Beverage'),

('M002', 'Vegan Sandwich', 'Grilled tofu and avocado', 50.00, 'Meal'),

('M003', 'Muffin', 'Chocolate chip muffin', 25.00, 'Snack'),

('M004', 'Chai Latte', 'Spiced tea with milk', 30.00, 'Beverage'),

('M005', 'Quiche', 'Spinach and cheese quiche', 45.00, 'Meal');

--Menu Inventory Item

CREATE TABLE Menu\_Inventory\_Item (

MenuItemID VARCHAR(10),

InventoryID VARCHAR(10),

QuantityUsed INT,

PRIMARY KEY (MenuItemID, InventoryID),

FOREIGN KEY (MenuItemID) REFERENCES Menu(MenuItemID),

FOREIGN KEY (InventoryID) REFERENCES Inventory\_Item(InventoryID)

);

INSERT INTO Menu\_Inventory\_Item VALUES

('M001', 'I001', 1), -- Milk for Café Latte

('M002', 'I002', 2), -- Bread for Sandwich

('M003', 'I003', 1), -- Chocolate for Muffin

('M004', 'I004', 1), -- Chai spice

('M005', 'I005', 2); -- Eggs for Quiche

--Promotion

CREATE TABLE Promotion (

PromoID VARCHAR(10) PRIMARY KEY,

PromoName VARCHAR(100),

Description VARCHAR(255),

DiscountRate DECIMAL(5,2),

StartDate DATE,

EndDate DATE

);

INSERT INTO Promotion VALUES

('P001', 'Winter Warmers', 'Hot drinks 10% off', 10.00, '2024-06-01', '2024-07-01'),

('P002', 'Snack Attack', 'Buy 2 Muffins Get 1 Free', 33.33, '2024-05-15', '2024-06-15'),

('P003', 'Lunch Combo', 'Free drink with Sandwich', 20.00, '2024-05-01', '2024-05-31'),

('P004', 'Monday Madness', 'All items 5% off', 5.00, '2024-05-06', '2024-05-27'),

('P005', 'Student Special', 'Extra 15% off with Student Card', 15.00, '2024-04-01', '2024-08-01');

--Product

CREATE TABLE Product (

ProductID VARCHAR(10) PRIMARY KEY,

ProductName VARCHAR(100),

Description VARCHAR(255),

Price DECIMAL(10,2),

StockLevel INT

);

INSERT INTO Product VALUES

('PR001', 'Branded T-Shirt', 'Cotton shirt with logo', 120.00, 50),

('PR002', 'Reusable Mug', 'Stainless steel travel mug', 80.00, 40),

('PR003', 'Canvas Tote Bag', 'Eco-friendly shopping bag', 70.00, 60),

('PR004', 'Sticker Pack', 'Set of 10 logo stickers', 25.00, 100),

('PR005', 'Notebook', 'A5 Journal with branded cover', 90.00, 35);

--Delivery

CREATE TABLE Delivery (

DeliveryID VARCHAR(10) PRIMARY KEY,

DeliveryDate DATE,

SupplierName VARCHAR(100),

Emp\_Num VARCHAR(10),

FOREIGN KEY (Emp\_Num) REFERENCES Employee(Emp\_Num)

);

INSERT INTO Delivery VALUES

('D001', '2024-05-01', 'Fresh Milk Co.', 'EMP001'),

('D002', '2024-05-02', 'GreenFarm Produce', 'EMP002'),

('D003', '2024-05-03', 'Urban Bakery', 'EMP003'),

('D004', '2024-05-04', 'Spice Central', 'EMP004'),

('D005', '2024-05-05', 'Eggworks Ltd.', 'EMP005');

--Delivery Line

CREATE TABLE Delivery\_Line (

DeliveryLineID VARCHAR(10) PRIMARY KEY,

DeliveryID VARCHAR(10),

InventoryID VARCHAR(10),

QuantityReceived INT,

UnitCost DECIMAL(10,2),

FOREIGN KEY (DeliveryID) REFERENCES Delivery(DeliveryID),

FOREIGN KEY (InventoryID) REFERENCES Inventory\_Item(InventoryID)

);

INSERT INTO Delivery\_Line VALUES

('DL001', 'D001', 'I001', 10, 15.00), -- Milk

('DL002', 'D002', 'I002', 20, 8.00), -- Bread

('DL003', 'D003', 'I003', 30, 12.00), -- Chocolate

('DL004', 'D004', 'I004', 15, 6.50), -- Chai

('DL005', 'D005', 'I005', 40, 5.00); -- Eggs

--Payment

CREATE TABLE Payment (

PaymentID INT PRIMARY KEY,

PaymentMethod VARCHAR(50),

AmountPaid DECIMAL(10,2),

OrderInvoiceNum VARCHAR(10),

FOREIGN KEY (OrderInvoiceNum) REFERENCES Order\_Invoice\_Line(OrderInvoiceNum)

);

INSERT INTO Payment VALUES

(1, 'Card', 70.00, 'INV001'),

(2, 'Cash', 50.00, 'INV002'),

(3, 'Card', 75.00, 'INV003'),

(4, 'Mobile Wallet', 30.00, 'INV004'),

(5, 'Cash', 90.00, 'INV005');

--Booking

CREATE TABLE Booking (

BookingID INT PRIMARY KEY,

CustomerStudentNum VARCHAR(10),

BookingDate DATE,

BookingSlot VARCHAR(100),

Cost DECIMAL(10,2),

NumGuests INT,

FOREIGN KEY (CustomerStudentNum) REFERENCES Customer(StudentNum)

);

INSERT INTO Booking VALUES

(101, '223185940', '2024-05-10', '10:00-12:00', 0.00, 1),

(102, '223021060', '2024-05-11', '14:00-16:00', 50.00, 2),

(103, '222062627', '2024-05-12', '16:00-18:00', 75.00, 3),

(104, '220000001', '2024-05-13', '18:00-20:00', 0.00, 1),

(105, '220000002', '2024-05-14', '12:00-14:00', 30.00, 2);

--Cancellation

CREATE TABLE Cancellation (

CancellationID INT PRIMARY KEY,

BookingID INT,

CancellationDate DATE,

CancellationReason VARCHAR(255),

FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)

);

INSERT INTO Cancellation VALUES

(1, 102, '2024-05-10', 'Double booked slot'),

(2, 103, '2024-05-11', 'Attendee fell ill'),

(3, 104, '2024-05-12', 'Switched to event ticket'),

(4, 105, '2024-05-13', 'Venue closed'),

(5, 101, '2024-05-14', 'Changed schedule');

--Event

CREATE TABLE Event (

Event\_ID VARCHAR(10) PRIMARY KEY,

CustomerStudentNum VARCHAR(10),

Event\_Name VARCHAR(100),

Event\_Description VARCHAR(255),

Event\_Date DATE,

FOREIGN KEY (CustomerStudentNum) REFERENCES Customer(StudentNum)

);

INSERT INTO Event VALUES

('EV001', '223185940', 'Jazz Night', 'Live jazz performance', '2024-05-10'),

('EV002', '223021060', 'Movie Night', 'Student film screening', '2024-05-11'),

('EV003', '222062627', 'Games Night', 'Board games and fun', '2024-05-12'),

('EV004', '220000001', 'Karaoke Night', 'Open mic and singing', '2024-05-13'),

('EV005', '220000002', 'Comedy Night', 'Stand-up comedy by students', '2024-05-14');

--Feedback

CREATE TABLE Feedback (

FeedbackID INT PRIMARY KEY,

CustomerStudentNum VARCHAR(10),

EventID VARCHAR(10),

Rating INT,

Comment VARCHAR(255),

Date\_Rated DATE,

FOREIGN KEY (CustomerStudentNum) REFERENCES Customer(StudentNum),

FOREIGN KEY (EventID) REFERENCES Event(Event\_ID)

);

INSERT INTO Feedback VALUES

(1, '223185940', 'EV001', 5, 'Amazing jazz night!', '2024-05-11'),

(2, '223021060', 'EV002', 4, 'Nice movie screening.', '2024-05-12'),

(3, '222062627', 'EV003', 5, 'Had lots of fun.', '2024-05-13'),

(4, '220000001', 'EV004', 3, 'Could be better.', '2024-05-14'),

(5, '220000002', 'EV005', 4, 'Very funny!', '2024-05-15');

--Stock Movement

CREATE TABLE Stock\_Movement (

MovementID VARCHAR(10) PRIMARY KEY,

InventoryID VARCHAR(10),

MovementDate DATE,

QuantityMoved INT,

MovementType VARCHAR(20), -- e.g., 'IN' or 'OUT'

Reason VARCHAR(100),

FOREIGN KEY (InventoryID) REFERENCES Inventory\_Item(InventoryID)

);

INSERT INTO Stock\_Movement VALUES

('SM001', 'I001', '2024-05-01', 50, 'IN', 'Delivery'),

('SM002', 'I002', '2024-05-01', -10, 'OUT', 'Menu Preparation'),

('SM003', 'I003', '2024-05-02', -20, 'OUT', 'Menu Preparation'),

('SM004', 'I004', '2024-05-03', 30, 'IN', 'New Stock'),

('SM005', 'I005', '2024-05-03', -15, 'OUT', 'Broken Eggs');

--Invoice

CREATE TABLE Invoice (

InvoiceID VARCHAR(10) PRIMARY KEY,

CustomerStudentNum VARCHAR(10),

InvoiceDate DATE,

TotalAmount DECIMAL(10,2),

Status VARCHAR(20),

FOREIGN KEY (CustomerStudentNum) REFERENCES Customer(StudentNum)

);

INSERT INTO Invoice VALUES

('INV100', '223185940', '2024-05-01', 150.00, 'Paid'),

('INV101', '223021060', '2024-05-02', 100.00, 'Paid'),

('INV102', '222062627', '2024-05-03', 90.00, 'Pending'),

('INV103', '220000001', '2024-05-04', 120.00, 'Paid'),

('INV104', '220000002', '2024-05-05', 75.00, 'Cancelled');

--Invoice Line

CREATE TABLE Invoice\_Line (

InvoiceLineID VARCHAR(10) PRIMARY KEY,

InvoiceID VARCHAR(10),

Description VARCHAR(100),

Quantity INT,

UnitPrice DECIMAL(10,2),

LineTotal DECIMAL(10,2),

FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID)

);

INSERT INTO Invoice\_Line VALUES

('IL001', 'INV100', 'Branded T-Shirt', 1, 120.00, 120.00),

('IL002', 'INV100', 'Sticker Pack', 1, 30.00, 30.00),

('IL003', 'INV101', 'Tote Bag', 1, 70.00, 70.00),

('IL004', 'INV102', 'Notebook', 1, 90.00, 90.00),

('IL005', 'INV103', 'Reusable Mug', 2, 60.00, 120.00);

--Supplier

CREATE TABLE Supplier (

SupplierID VARCHAR(10) PRIMARY KEY,

SupplierName VARCHAR(100),

ContactPerson VARCHAR(100),

Phone VARCHAR(20),

Email VARCHAR(100),

Address VARCHAR(255)

);

INSERT INTO Supplier VALUES

('SUP001', 'Fresh Milk Co.', 'Sarah M.', '0711000001', 'milk@supplier.co.za', '123 Dairy Lane, JHB'),

('SUP002', 'GreenFarm Produce', 'Jonas K.', '0722000002', 'produce@greenfarm.co.za', '78 Farm Rd, CPT'),

('SUP003', 'Urban Bakery', 'Thabo D.', '0733000003', 'orders@urbanbakery.co.za', '5 Baker St, DBN'),

('SUP004', 'Spice Central', 'Leila H.', '0744000004', 'spice@central.co.za', '99 Spice Market, PTA'),

('SUP005', 'Eggworks Ltd.', 'Sam T.', '0755000005', 'sales@eggworks.co.za', '11 Poultry Rd, NW');

--Supplier Delivery

CREATE TABLE Supplier\_Delivery (

SupplierDeliveryID VARCHAR(10) PRIMARY KEY,

SupplierID VARCHAR(10),

DeliveryDate DATE,

DeliveryNote VARCHAR(255),

Emp\_Num VARCHAR(10),

FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),

FOREIGN KEY (Emp\_Num) REFERENCES Employee(Emp\_Num)

);

INSERT INTO Supplier\_Delivery VALUES

('SD001', 'SUP001', '2024-05-01', 'Milk delivered in full', 'EMP001'),

('SD002', 'SUP002', '2024-05-02', 'Fresh produce received', 'EMP002'),

('SD003', 'SUP003', '2024-05-03', 'Bread delivered late', 'EMP003'),

('SD004', 'SUP004', '2024-05-04', 'Spices packed properly', 'EMP004'),

('SD005', 'SUP005', '2024-05-05', 'Eggs inspected upon arrival', 'EMP005');

--Supply Request

CREATE TABLE Supply\_Request (

RequestID VARCHAR(10) PRIMARY KEY,

SupplierID VARCHAR(10),

InventoryID VARCHAR(10),

RequestedQuantity INT,

RequestDate DATE,

Status VARCHAR(20),

RequestedBy VARCHAR(10),

FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),

FOREIGN KEY (InventoryID) REFERENCES Inventory\_Item(InventoryID),

FOREIGN KEY (RequestedBy) REFERENCES Employee(Emp\_Num)

);

INSERT INTO Supply\_Request VALUES

('SR001', 'SUP001', 'I001', 50, '2024-04-28', 'Approved', 'EMP001'),

('SR002', 'SUP002', 'I002', 60, '2024-04-29', 'Pending', 'EMP002'),

('SR003', 'SUP003', 'I003', 70, '2024-04-30', 'Approved', 'EMP003'),

('SR004', 'SUP004', 'I004', 40, '2024-05-01', 'Rejected', 'EMP004'),

('SR005', 'SUP005', 'I005', 80, '2024-05-02', 'Approved', 'EMP005');

--SQL QUERIES

--1.Goal: Track loyalty point changes over time and classify customers by tier.

--Short Description:

--Summarizes loyalty account activity and classifies customers, potentially recognizing high-value customers beyond their official tier.

--Business Advantages:

--• Helps identify and retain loyal customers.

--• Encourages personalized marketing by understanding high spenders.

--• Aids in loyalty program optimization and customer retention strategies.

WITH PointSummary AS (

SELECT

la.Acc\_ID,

c.FName,

c.LName,

SUM(CASE WHEN lt.Points\_Change > 0 THEN lt.Points\_Change ELSE 0 END) AS TotalPointsEarned,

SUM(CASE WHEN lt.Points\_Change < 0 THEN ABS(lt.Points\_Change) ELSE 0 END) AS TotalPointsRedeemed,

la.CurrentPoints,

la.Tier

FROM LoyaltyAccount la

JOIN Loyalty\_Transaction lt ON la.Acc\_ID = lt.LoyaltyAccountAcc\_ID

JOIN Customer c ON la.CustomerStudentNum = c.StudentNum

GROUP BY la.Acc\_ID, c.FName, c.LName, la.CurrentPoints, la.Tier

)

SELECT

ps.\*,

-- Subquery: Count how many redemptions each customer has made

(SELECT COUNT(\*)

FROM Redemption r

WHERE r.LoyaltyAccountAcc\_ID = ps.Acc\_ID) AS RedemptionCount,

-- Adjust the tier if net points exceed 200

CASE

WHEN (TotalPointsEarned - TotalPointsRedeemed) > 200 THEN 'Platinum (Unofficial)'

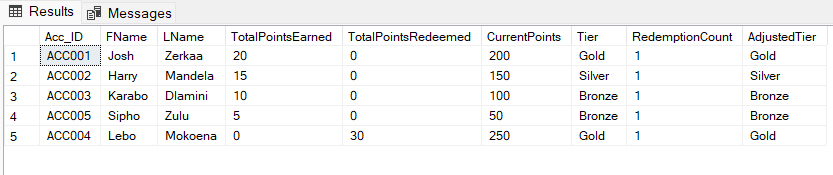
ELSE Tier

END AS AdjustedTier

FROM PointSummary ps

ORDER BY TotalPointsEarned DESC;

## Query 1 Output:



--2. Analyze which weekdays attract the most event bookings.

--Short Description:

--Breaks down event popularity by day of the week between two selected dates, offering insights into peak demand periods.

-- Stored procedure to analyze event activity by weekday within a date range

--Business Advantages:

--• Informs scheduling and staffing decisions for events.

--• Helps with marketing promotions on high/low performing days.

--• Improves planning for event inventory and logistics.

CREATE PROCEDURE sp\_WeeklyEventStats\_WithTopEvent

@StartDate DATE,

@EndDate DATE

AS

BEGIN

SELECT

DATENAME(WEEKDAY, e.Event\_Date) AS Weekday,

COUNT(e.Event\_ID) AS TotalEvents,

AVG(f.Rating) AS AvgRating,

STRING\_AGG(e.Event\_Name, ', ') AS SampleEvents,

-- Subquery: Find the top-rated event for the weekday

(SELECT TOP 1 e2.Event\_Name

FROM Event e2

JOIN Feedback f2 ON e2.Event\_ID = f2.EventID

WHERE DATENAME(WEEKDAY, e2.Event\_Date) = DATENAME(WEEKDAY, e.Event\_Date)

AND e2.Event\_Date BETWEEN @StartDate AND @EndDate

ORDER BY f2.Rating DESC) AS TopRatedEvent

FROM Event e

LEFT JOIN Feedback f ON e.Event\_ID = f.EventID

WHERE e.Event\_Date BETWEEN @StartDate AND @EndDate

GROUP BY DATENAME(WEEKDAY, e.Event\_Date)

ORDER BY TotalEvents DESC;

END;

EXEC sp\_WeeklyEventStats\_WithTopEvent @StartDate = '2024-05-01', @EndDate = '2024-05-31';

### Query 2 Output:

A screenshot of a computer screen

AI-generated content may be incorrect.

--3.Goal: Compare employee wages to department budgets and flag anomalies.

--Short Description:

--Provides a budget overview for departments and checks if certain employees are being paid less than expected.

--Business Advantages:

--• Ensures fairness and transparency in compensation.

--• Helps HR and Finance monitor wage distribution efficiency.

--• Prevents underpayment of critical staff.

CREATE VIEW vw\_DepartmentBudget AS

SELECT

d.Dep\_Num,

d.Dep\_Name,

d.Wage AS AvgDepartmentWage,

COUNT(e.Emp\_Num) AS EmployeeCount,

(d.Wage \* COUNT(e.Emp\_Num)) AS TotalDepartmentBudget,

b.BranchName

FROM Department d

JOIN Employee e ON d.Dep\_Num = e.Dep\_Num

JOIN Branch b ON d.BranchID = b.BranchID

GROUP BY d.Dep\_Num, d.Dep\_Name, d.Wage, b.BranchName;

select \* from vw\_DepartmentBudget

### Query 3 Output:

A screenshot of a computer

AI-generated content may be incorrect.

--4.creates an inventory audit report that correlates supply requests with deliveries and stock movements.

--Short Description:

--Provides a detailed audit linking supply requests to actual inventory received and recorded movements.

--Business Advantages:

--• Enhances transparency in inventory management.

--• Detects discrepancies between requested and received stock.

--• Supports auditing and compliance processes.

CREATE PROCEDURE sp\_InventoryAuditTrail

AS

BEGIN

SELECT

sr.RequestID,

sr.RequestedQuantity,

dl.QuantityReceived,

sm.QuantityMoved,

sm.MovementType,

sr.Status,

d.DeliveryDate

FROM Supply\_Request sr

LEFT JOIN Inventory\_Item i ON sr.InventoryID = i.InventoryID

LEFT JOIN Delivery\_Line dl ON dl.InventoryID = i.InventoryID

LEFT JOIN Delivery d ON d.DeliveryID = dl.DeliveryID

LEFT JOIN Stock\_Movement sm ON sm.InventoryID = i.InventoryID

WHERE sr.Status = 'Approved' AND sm.MovementType = 'IN'

ORDER BY sr.RequestDate;

END;

EXEC sp\_InventoryAuditTrail;

### Query 4 Output:

A screenshot of a computer

AI-generated content may be incorrect.

--5. Flag Gold-tier customers inactive for 60+ days.

--Short Description:

--Flags high-tier but inactive customers to prompt re-engagement strategies or tier reviews.

--Business Advantages:

--• Enables proactive re-engagement campaigns to retain premium customers.

--• Helps reduce loyalty program costs by identifying inactive users.

--• Encourages more frequent purchases through targeted reminders.

CREATE PROCEDURE sp\_FlagInactiveGoldCustomers

@InactivityDays INT = 60

AS

BEGIN

SELECT

c.FName,

c.LName,

la.Join\_Date,

MAX(o.OrderDate) AS LastOrderDate,

DATEDIFF(DAY, MAX(o.OrderDate), GETDATE()) AS DaysSinceLastOrder,

DATEADD(DAY, @InactivityDays, MAX(o.OrderDate)) AS TierExpirationDate

FROM LoyaltyAccount la

JOIN Customer c ON la.CustomerStudentNum = c.StudentNum

LEFT JOIN [Order] o ON c.StudentNum = o.CustomerStudentNum

WHERE la.Tier = 'Gold'

GROUP BY c.FName, c.LName, la.Join\_Date

HAVING DATEDIFF(DAY, MAX(o.OrderDate), GETDATE()) >= @InactivityDays;

END;

EXEC sp\_FlagInactiveGoldCustomers @InactivityDays = 60;

### Query 5 output:

A screenshot of a computer

AI-generated content may be incorrect.