# Cafe Management System

**ORACLE SQL DATABASE PROJECT**

**Prova Paul | Roll : 2007037**

**CSE 3110: Database Systems Laboratory**

## Introduction

### ➢ Project Overview

The cafe management system's Oracle SQL database project is structured to support comprehensive management of daily operations through a series of interconnected tables. It manages employees, customers, menu items, orders, and payments, ensuring efficient tracking of transactions and inventory. Additionally, it facilitates the handling of supplier details and order fulfillment, linking every purchase to stock levels and financial flows.

### ➢ Importance of Database in Cafe Management System

The database in a cafe management system centralizes critical operational data across tables for employees, customers, menu items, orders, and suppliers, ensuring efficient management of daily activities and resources. It facilitates precise tracking of transactions and inventory, enabling optimized supply chain management and real-time financial oversight. By integrating these elements, the database supports strategic decision-making and enhances overall business efficiency and customer satisfaction.

## Project Objective

- Design a comprehensive relational database schema that effectively organizes and stores various aspects of cafe-related data. This includes information on employees, menu items, customers, orders, inventory, and suppliers. The schema aims to ensure easy data integration and maintain consistency across the platform, supporting complex queries for reliable data retrieval and management.
- Implementing a system for efficient operation of order handling and menu management with real-time updates and user-friendly interfaces.
- Implementing a system for managing employee details, schedule within the cafe management system which is essential for various operations.
- Besides utilizing SQL queries to retrieve, update, and manage data in the cafe management system is imperative for diverse functionalities. They enable functionalities such as managing menu items, employee details, supplier information, and customer records. Efficient execution ensures an up-to-date and accurate database, supporting dynamic needs like order tracking, inventory management, and customer engagement.

## Database Design

### ➢ Overviews of Database Schema:

I've created the following tables to systematically organize and manage essential data for efficient operations and streamlined management within the cafe.

**1.Employees Table:** Stores detailed information about individual employees, including their employee ID, first name, last name, position, and hire date.
**2.Customers Table:** Contains data about cafe customers, including customer ID, name, email address, and phone number.
**3.MenuItems Table:** Includes information about menu items offered by the cafe, such as item ID, name, description, and price.
**4.Orders Table:** Records details about customer orders, including order ID, customer ID (linked to the Customers table), order date, and total amount.
**5.OrderDetails Table:** Captures specifics about each order, linking to the Orders table, including order detail ID, order ID (linked to the Orders table), menu item ID (linked to the MenuItems table), quantity, and prices.
**6.Suppliers Table:** Tracks information about suppliers providing inventory items to the cafe, including supplier ID, name, contact name, and phone number.
**7.Inventory Table:** Manages inventory levels of items supplied by suppliers, including inventory ID, menu item ID (linked to the MenuItems table), supplier ID (linked to the Suppliers table), quantity, and restock date.
**8.Payments Table:** Logs payment details for orders, including payment ID, order ID (linked to the Orders table), payment date, amount paid, and payment method.

## ➢ Rationale Behind the System Architecture:

The architecture of the cafe management system was formulated to holistically capture and effectively manage various aspects of cafe operations. Each module was meticulously crafted to cater to specific operational facets, guaranteeing seamless handling of diverse tasks—from inventory management and sales tracking to customer engagement and staff scheduling.

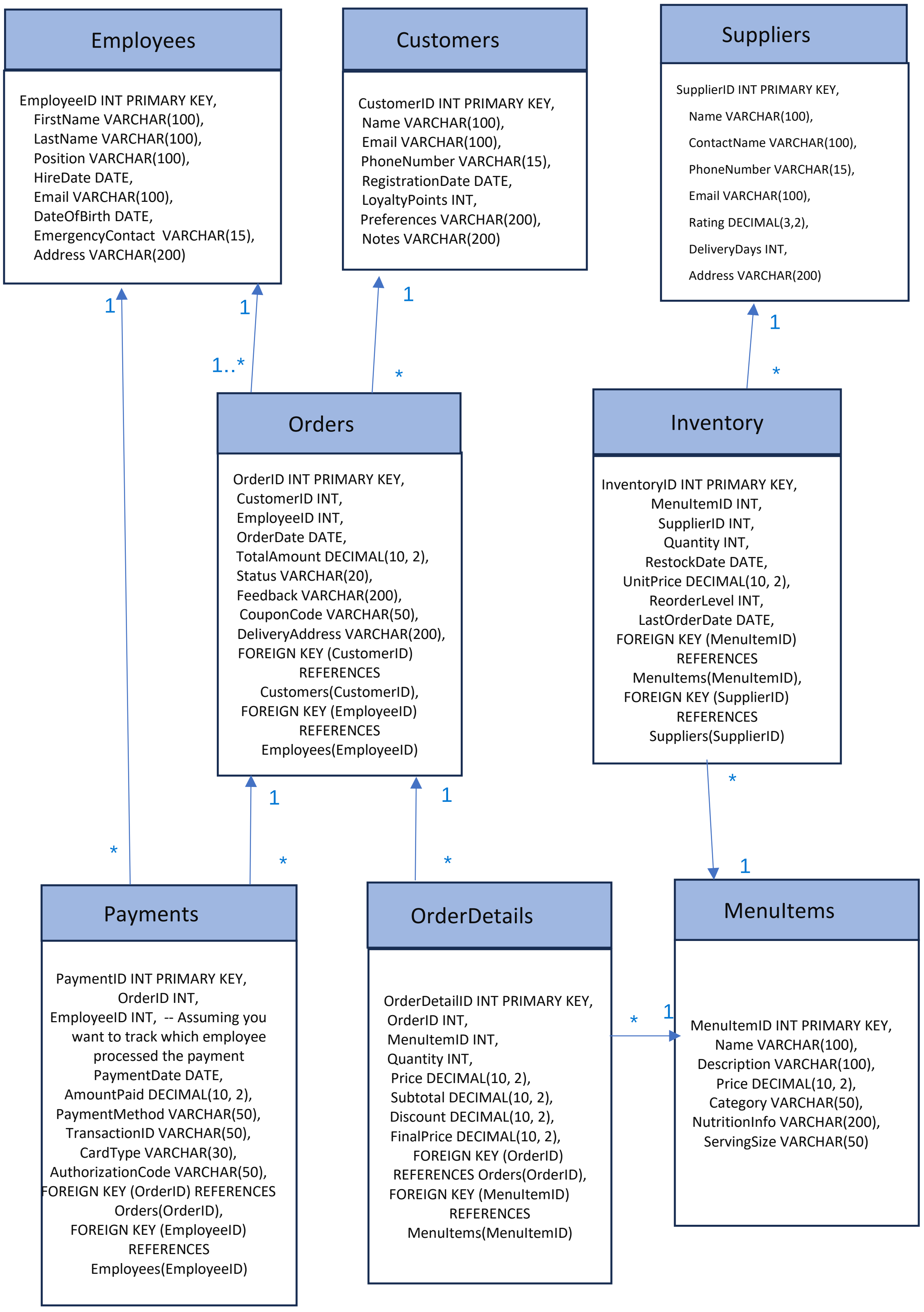
## ➢ Table Relationships:

The table relationships in my cafe management system project are designed to establish connections between different entities, such as employees, customers, menu items, orders,suppliers, inventory, and payments. These relationships ensure data integrity and enable efficient data retrieval and management within the database.

```sql
-- Creating the Employees table
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Position VARCHAR(100),
    HireDate DATE,
    Email VARCHAR(100),
    DateOfBirth DATE,
    EmergencyContact VARCHAR(15),
    Address VARCHAR(200)
);
-- Creating the Customers table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    PhoneNumber VARCHAR(15),
    RegistrationDate DATE,
    LoyaltyPoints INT,
    Preferences VARCHAR(200),
    Notes VARCHAR(200)
);
-- Creating the MenuItems table
CREATE TABLE MenuItems (
    MenuItemID INT PRIMARY KEY,
    Name VARCHAR(100),
    Description VARCHAR(100),
    Price DECIMAL(10, 2),
    Category VARCHAR(50),
    NutritionInfo VARCHAR(200),
    ServingSize VARCHAR(50)
);


-- Creating the Orders table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    EmployeeID INT,
    OrderDate DATE,
    TotalAmount DECIMAL(10, 2),
    Status VARCHAR(20),
    Feedback VARCHAR(200),
    CouponCode VARCHAR(50),
    DeliveryAddress VARCHAR(200),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
-- Creating the OrderDetails table
CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    MenuItemID INT,
    Quantity INT,
    Price DECIMAL(10, 2),
    Subtotal DECIMAL(10, 2),
    Discount DECIMAL(10, 2),
    FinalPrice DECIMAL(10, 2),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (MenuItemID) REFERENCES MenuItems(MenuItemID)
);
-- Creating the Suppliers table
CREATE TABLE Suppliers (
    SupplierID INT PRIMARY KEY,
    Name VARCHAR(100),
    ContactName VARCHAR(100),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100),
    Rating DECIMAL(3,2),
    DeliveryDays INT,
    Address VARCHAR(200)
);
-- Creating the Inventory table
CREATE TABLE Inventory (
    InventoryID INT PRIMARY KEY,
    MenuItemID INT,
    SupplierID INT,
    Quantity INT,
    RestockDate DATE,
    UnitPrice DECIMAL(10, 2),
    ReorderLevel INT,
    LastOrderDate DATE,
    FOREIGN KEY (MenuItemID) REFERENCES MenuItems(MenuItemID),
    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
);
-- Creating the Payments table
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY,
    OrderID INT,
    EmployeeID INT,
    PaymentDate DATE,
    AmountPaid DECIMAL(10, 2),
    PaymentMethod VARCHAR(50),
    TransactionID VARCHAR(50),
    CardType VARCHAR(30),
    AuthorizationCode VARCHAR(50),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

- **Entity-Relationship Diagram:**



```
Employees
EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Position VARCHAR(100),
    HireDate DATE,
    Email VARCHAR(100),
    DateOfBirth DATE,
    EmergencyContact  VARCHAR(15),
    Address VARCHAR(200)
```

```
Customers
CustomerID INT PRIMARY KEY,
 Name VARCHAR(100),
 Email VARCHAR(100),
 PhoneNumber VARCHAR(15),
 RegistrationDate DATE,
 LoyaltyPoints INT,
 Preferences VARCHAR(200),
 Notes VARCHAR(200)
```

```
Suppliers
SupplierID INT PRIMARY KEY,
   Name VARCHAR(100),
   ContactName VARCHAR(100),
   PhoneNumber VARCHAR(15),
   Email VARCHAR(100),
   Rating DECIMAL(3,2),
   DeliveryDays INT,
   Address VARCHAR(200)
```

```
Orders
OrderID INT PRIMARY KEY,
 CustomerID INT,
 EmployeeID INT,
 OrderDate DATE,
 TotalAmount DECIMAL(10, 2),
 Status VARCHAR(20),
 Feedback VARCHAR(200),
 CouponCode VARCHAR(50),
 DeliveryAddress VARCHAR(200),
 FOREIGN KEY (CustomerID)
          REFERENCES
     Customers(CustomerID),
 FOREIGN KEY (EmployeeID)
          REFERENCES
     Employees(EmployeeID)
```

```
Inventory
InventoryID INT PRIMARY KEY,
      MenuItemID INT,
       SupplierID INT,
         Quantity INT,
     RestockDate DATE,
  UnitPrice DECIMAL(10, 2),
     ReorderLevel INT,
    LastOrderDate DATE,
 FOREIGN KEY (MenuItemID)
        REFERENCES
   MenuItems(MenuItemID),
  FOREIGN KEY (SupplierID)
        REFERENCES
     Suppliers(SupplierID)
```

```
Payments
 PaymentID INT PRIMARY KEY,
        OrderID INT,
EmployeeID INT,  -- Assuming you
    want to track which employee
        processed the payment
        PaymentDate DATE,
   AmountPaid DECIMAL(10, 2),
 PaymentMethod VARCHAR(50),
   TransactionID VARCHAR(50),
     CardType VARCHAR(30),
AuthorizationCode VARCHAR(50),
FOREIGN KEY (OrderID) REFERENCES
        Orders(OrderID),
   FOREIGN KEY (EmployeeID)
          REFERENCES
      Employees(EmployeeID)
```

```
OrderDetails
OrderDetailID INT PRIMARY KEY,
 OrderID INT,
 MenuItemID INT,
 Quantity INT,
  Price DECIMAL(10, 2),
  Subtotal DECIMAL(10, 2),
  Discount DECIMAL(10, 2),
  FinalPrice DECIMAL(10, 2),
      FOREIGN KEY (OrderID)
  REFERENCES Orders(OrderID),
 FOREIGN KEY (MenuItemID)
          REFERENCES
    MenuItems(MenuItemID)
```

```
MenuItems
MenuItemID INT PRIMARY KEY,
    Name VARCHAR(100),
  Description VARCHAR(100),
     Price DECIMAL(10, 2),
    Category VARCHAR(50),
 NutritionInfo VARCHAR(200),
   ServingSize VARCHAR(50)
```

➢ **Table Relationships Description:**

1. The Employees table has a foreign key relationship with the Orders table, linking EmployeeID to track which employee took the order.
2. Customers table's CustomerID serves as a primary key, linked to the Orders table's CustomerID, facilitating customer order tracking.
3. MenuItems table's MenuItemID is a primary key referenced by OrderDetails table's MenuItemID, connecting orders to specific menu items.
4. Suppliers table's SupplierID is a primary key referenced by the Inventory table's SupplierID, establishing relationships for inventory management.
5. The OrderID in the Orders table is used in both the OrderDetails and Payments tables to keep track of order details and payments accurately.
6. In the Payments table, the OrderID column connects payments with specific orders.
7. The Inventory table uses MenuItemID and SupplierID from MenuItems and Suppliers to manage inventory effectively.

## SQL Queries and Functionality

- **Examples of SQL queries:**

1. Adding a temporary column to Employees table:
   - ALTER TABLE Employees ADD temporary_column VARCHAR2(50);

2. Dropping the temporary column from Employees table:
   - ALTER TABLE Employees DROP COLUMN temporary_column;

3. Modifying the data type of the HireDate column in Employees table:
   - ALTER TABLE Employees MODIFY HireDate DATE;

4. Renaming the FirstName column to First_Name in Employees table:
   - ALTER TABLE Employees RENAME COLUMN FirstName TO First_Name;

5. Updating the HireDate for a specific EmployeeID in Employees table:
   - UPDATE Employees SET HireDate = DATE '2022-06-29' WHERE EmployeeID = 8;

6. Deleting a row from Employees table based on EmployeeID:
   - DELETE FROM Employees WHERE EmployeeID = 5;

7. Retrieving Email for a specific CustomerID in Customers table:

- SELECT Email FROM Customers WHERE CustomerID = 1;

8. Retrieving Email and Name for CustomerIDs between 1 and 5:
   - SELECT Email, Name FROM Customers WHERE CustomerID BETWEEN 1 AND 5;

9. Ordering Email and Name in ascending order by CustomerID:
   - SELECT Email,Name  FROM Customers ORDER BY CustomerID;

10. Calculating the maximum CustomerID in Customers table:
    - SELECT MAX(CustomerID) FROM Customers;

11. Retrieving CustomerIDs where Email contains 'ma':
    - SELECT CustomerID FROM Customers WHERE Email LIKE '%ma%';

12. Counting the number of employees per Position in Employees table:
    - SELECT COUNT(EmployeeID), Position FROM Employees GROUP BY Position;

13. Counting the number of employees per Position having more than one employee:
    - SELECT COUNT(EmployeeID), Position FROM Employees GROUP BY Position HAVING COUNT(EmployeeID) > 1;

14. Retrieving FirstNames of employees who are Managers:
    - SELECT FirstName FROM Employees WHERE EmployeeID IN (SELECT EmployeeID FROM Employees WHERE Position = 'Manager');

- **Implementation of Advanced SQL Features:**

➢ Insert and Set Default pl/sql Values: Inserts a new employee record with default values using PL/SQL variables and the current system date.

```
SET SERVEROUTPUT ON;
DECLARE
  v_employee_id Employees.EmployeeID%TYPE := 9;
  v_first_name Employees.FirstName%TYPE := 'John';
  v_last_name Employees.LastName%TYPE := 'Doe';
  v_position Employees.Position%TYPE := 'Manager';
  v_hire_date Employees.HireDate%TYPE := SYSDATE;
  v_email Employees.Email%TYPE := 'john.doe@example.com';
  v_dob Employees.DateOfBirth%TYPE := DATE '1985-05-15';
  v_emergency_contact Employees.EmergencyContact%TYPE := '1234567890';
  v_address Employees.Address%TYPE := '123 Maple Street';
```

```
BEGIN
  INSERT INTO Employees VALUES (v_employee_id, v_first_name, v_last_name,
v_position, v_hire_date, v_email, v_dob, v_emergency_contact, v_address);
  DBMS_OUTPUT.PUT_LINE('Insert successful');
END;
/
```

- Row Type: Retrieves and displays details of an employee with EmployeeID 9 using %ROWTYPE.

```
   SET SERVEROUTPUT ON;
DECLARE
  v_employee_row Employees%ROWTYPE;
BEGIN
  SELECT * INTO v_employee_row FROM Employees WHERE EmployeeID = 9;
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_employee_row.FirstName || ' ' ||
v_employee_row.LastName);
END;
/
```

- Cursor and Row Count: Iterates through all employee records using a cursor and displays their IDs and names.

```
SET SERVEROUTPUT ON;
DECLARE
  CURSOR employee_cursor IS SELECT * FROM Employees;
  v_employee_row Employees%ROWTYPE;
BEGIN
  OPEN employee_cursor;
  FETCH employee_cursor INTO v_employee_row;
  WHILE employee_cursor%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_row.EmployeeID || ', Name:
' || v_employee_row.FirstName || ' ' || v_employee_row.LastName);
    FETCH employee_cursor INTO v_employee_row;
  END LOOP;
  CLOSE employee_cursor;
END;
/
```

- FOR LOOP / WHILE LOOP / ARRAY with EXTEND() Function: Fetches employee names for IDs 1 to 9 and stores them in an array using a FOR LOOP and the EXTEND() function.

```
SET SERVEROUTPUT ON;
```

```
DECLARE
  TYPE NameArray IS VARRAY(9) OF VARCHAR2(100);
  v_names NameArray := NameArray();
  v_name VARCHAR2(100);
BEGIN
  FOR i IN 1..9 LOOP -- Loop through all employee IDs from 1 to 9
    BEGIN
      SELECT FirstName INTO v_name FROM Employees WHERE EmployeeID = i;
      v_names.EXTEND; -- Extend the array for each new name
      v_names(v_names.LAST) := v_name;
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        NULL;
    END;
  END LOOP;
END;
/
```

➢ IF / ELSEIF / ELSE: Determines the value range of order totals and prints corresponding messages for each order.

```
SET SERVEROUTPUT ON;
DECLARE
  v_order_total Orders.TotalAmount%TYPE;
BEGIN
  FOR rec IN (SELECT * FROM Orders) LOOP
    v_order_total := rec.TotalAmount;
    IF v_order_total > 1000 THEN
      DBMS_OUTPUT.PUT_LINE('Order ID ' || rec.OrderID || ' is a high value 
order.');
    ELSIF v_order_total BETWEEN 500 AND 1000 THEN
      DBMS_OUTPUT.PUT_LINE('Order ID ' || rec.OrderID || ' is a medium value 
order.');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Order ID ' || rec.OrderID || ' is a low value 
order.');
    END IF;
  END LOOP;
END;
/
```

➢ PL/SQL Procedure to Fetch Employee Details by ID: Retrieves and displays details of an employee by EmployeeID using a PL/SQL procedure.

```
SET SERVEROUTPUT ON;
```

```
CREATE OR REPLACE PROCEDURE GetEmployeeDetails (p_emp_id IN
Employees.EmployeeID%TYPE) IS
  v_first_name Employees.FirstName%TYPE;
  v_last_name Employees.LastName%TYPE;
  v_position Employees.Position%TYPE;
  v_hire_date Employees.HireDate%TYPE;
  v_email Employees.Email%TYPE;
  v_dob Employees.DateOfBirth%TYPE;
  v_emergency_contact Employees.EmergencyContact%TYPE;
  v_address Employees.Address%TYPE;
BEGIN
  SELECT FirstName, LastName, Position, HireDate, Email, DateOfBirth,
EmergencyContact, Address
  INTO v_first_name, v_last_name, v_position, v_hire_date, v_email, v_dob,
v_emergency_contact, v_address
  FROM Employees
  WHERE EmployeeID = p_emp_id;

  DBMS_OUTPUT.PUT_LINE('First Name: ' || v_first_name || ', Last Name: ' ||
v_last_name ||
                       ', Position: ' || v_position || ', Hire Date: ' ||
v_hire_date ||
                       ', Email: ' || v_email || ', Date of Birth: ' || v_dob ||
                       ', Emergency Contact: ' || v_emergency_contact || ',
Address: ' || v_address);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || p_emp_id);
END;
/
```

➢ Create a trigger that automatically updates the TotalAmount in the Orders table: This trigger automatically updates the inventory quantity for menu items after an order is placed, ensuring accurate stock management in the café management system.

```
➢ SET SERVEROUTPUT ON;
➢ CREATE OR REPLACE TRIGGER UpdateOrderAndInventory
➢ AFTER INSERT ON OrderDetails
➢ FOR EACH ROW
➢ DECLARE
➢     v_new_inventory INT;
➢ BEGIN
➢     UPDATE Inventory
➢     SET Quantity = Quantity - :NEW.Quantity
```

```
    WHERE MenuItemID = :NEW.MenuItemID;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'No inventory record found for
MenuItemID: ' || :NEW.MenuItemID);
    END IF;

    SELECT Quantity INTO v_new_inventory FROM Inventory WHERE MenuItemID =
:NEW.MenuItemID;
    DBMS_OUTPUT.PUT_LINE('Updated Inventory for MenuItemID ' ||
:NEW.MenuItemID || ': ' || v_new_inventory);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No inventory record found for MenuItemID ' ||
:NEW.MenuItemID);
END;
/
```

## Targeted Audience

- This Cafe Management system project is designed to cater to a diverse range of users within the cafe management community, accommodating various interests and requirements associated with running a cafe. The primary target groups include:

### 1.Cafe Owners and Managers:

- Manage stock easily: With Oracle, it's very easy to keep track of what's in my cafe in real-time. This helps to avoid running out of stuff and wasting money.
- Smart decisions: Use Oracle's tools to figure out the best times to schedule the staff, which items on his menu are selling the most, and how to price things right. It helps him run the cafe better and make more money.

### 2.Cafe Staff (Baristas, Servers, Kitchen Staff):

- Smooth operations: Oracle's system is easy to use, making it simple for staff to take orders, know what ingredients are available, and serve customers quickly.
- Mobile-friendly: You can use Oracle on your phone or tablet, so you're always in the loop even during busy times.

### 3.Customers:

- Personal touch: Oracle remembers your preferences, what you've ordered before, and any loyalty points you've earned. This means you get recommendations tailored just for you and even special offers to make you happy.

• Easy access: With Oracle, you can easily check out the menu, see what's new, and get rewards for being a loyal customer.

### 4.Suppliers:

• Smooth communication: Oracle helps you know when the café needs more supplies, so you can deliver at the right time. This means no more guessing games and fewer chances of running out of stock.

•Plan better: With Oracle, you can see what the café needs in advance, so you can plan your production and deliveries more efficiently.

### 5.Financial Advisors and Accountants:

• Keep track of money: Oracle helps you see exactly how much money the café is making and spending. This makes it easier to plan for the future and make smart financial decisions.

• Stay compliant: Oracle makes sure you're following all the rules and regulations, so you don't get in trouble with the authorities.

### 6.Regulatory Authorities:

• Secure data: Oracle keeps all the café's information safe and secure, so you can trust that everything is being handled properly.

• Easy audits: With Oracle's reporting tools, you can quickly find the information you need for audits and compliance checks.

### 7.Technology Partners and Developers:

• Customize with ease: Oracle's flexible system lets developers create new features and integrations quickly, so the café can stay ahead of the game.

• Reliable support: With access to Oracle's tools and resources, developers can keep the cafe's system running smoothly and make sure it grows with the business.

## Conclusion

### ➢ Enhanced Customer Experience and Engagement

The cafe management database plays a pivotal role in enhancing customer experience and engagement. Through features such as customer preferences tracking and loyalty points management, it enables cafes to personalize services and rewards for their patrons. Moreover, the database's ability to streamline order processing and optimize service delivery enhances overall customer satisfaction, leading to positive reviews and word-of-mouth referrals.

### ➢ Empowering Data-Driven Decision Making

The cafe management database empowers cafe owners and managers to make informed, data-driven decisions. By providing comprehensive insights into sales trends, inventory levels, and customer preferences, it enables strategic planning and resource allocation. Additionally, the database facilitates performance analysis and benchmarking, allowing cafes to identify areas for improvement and implement targeted strategies for growth.

## ➢ Reflection on Progress and Challenges

As we reflect on the development journey of our cafe management system, we've encountered both notable achievements and significant challenges. We successfully navigated through mastering intricate SQL functionalities and implementing advanced database management techniques, which enhanced the system's robustness. However, challenges such as ensuring data consistency from diverse sources and scaling the system to accommodate burgeoning data volumes required innovative solutions, including the deployment of rigorous data validation methods and optimizing database performance.

## ➢ Importance of the Database in Cafe Operations

The cafe management database holds paramount importance in facilitating efficient cafe operations. It serves as a vital tool for cafe owners, staff, and patrons alike. By offering a centralized repository of cafe-related data, it streamlines various aspects of cafe management, from employee scheduling to inventory tracking. This database not only enhances operational efficiency but also fosters better customer service and satisfaction. Its ability to provide real-time insights and streamline decision-making processes makes it an indispensable asset in the cafe industry.

## Reference

1. https://github.com/shahidul034/Database-Lab-Tutorial
2. https://www.geeksforgeeks.org/dbms/