```
-- Create Tables for Restaurant Database

-- Customers Table
CREATE TABLE Customers (
    customer_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(20),
    address VARCHAR(255),
    loyalty_points NUMERIC(8,2) DEFAULT 0,
    join_date DATE DEFAULT CURRENT_DATE,
    birth_date DATE,
    dietary_preferences TEXT,
    allergy_information TEXT,
    last_visit_date DATE,
    marketing_opt_in BOOLEAN DEFAULT TRUE
);

-- Tables Table (for restaurant seating)
CREATE TABLE Tables (
    table_id SERIAL PRIMARY KEY,
    table_number VARCHAR(10) NOT NULL UNIQUE,
    capacity INTEGER NOT NULL,
    section VARCHAR(20),
    is_accessible BOOLEAN DEFAULT FALSE,
    status VARCHAR(20) DEFAULT 'Available'
        CHECK (status IN ('Available', 'Reserved', 'Occupied',
'Maintenance'))
);

-- Reservations Table
CREATE TABLE Reservations (
    reservation_id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES Customers(customer_id),
    reservation_date DATE NOT NULL,
    reservation_time TIME NOT NULL,
    party_size INTEGER NOT NULL,
    table_id INTEGER REFERENCES Tables(table_id),
    special_requests TEXT,
    status VARCHAR(20) DEFAULT 'Confirmed'
        CHECK (status IN ('Confirmed', 'Canceled', 'Completed')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    reserved_by VARCHAR(50),
    contact_phone VARCHAR(20),
    CONSTRAINT valid_party_size CHECK (party_size > 0)
);

-- Menu Categories Table
CREATE TABLE Menu_Categories (
    category_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description TEXT,
    display_order INTEGER
);

-- Menu Table
```

```
CREATE TABLE Menu (
    dish_id SERIAL PRIMARY KEY,
    category_id INTEGER REFERENCES Menu_Categories(category_id),
    dish_name VARCHAR(100) NOT NULL,
    description TEXT,
    price NUMERIC(6,2) NOT NULL,
    is_vegetarian BOOLEAN DEFAULT FALSE,
    is_vegan BOOLEAN DEFAULT FALSE,
    is_gluten_free BOOLEAN DEFAULT FALSE,
    calories INTEGER,
    preparation_time INTEGER, -- minutes
    is_available BOOLEAN DEFAULT TRUE,
    is_seasonal BOOLEAN DEFAULT FALSE,
    image_path VARCHAR(255)
);

-- Suppliers Table
CREATE TABLE Suppliers (
    supplier_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    contact_person VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20),
    address VARCHAR(255),
    preferred_status BOOLEAN DEFAULT FALSE,
    lead_time_days INTEGER
);

-- Ingredients Table
CREATE TABLE Ingredients (
    ingredient_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    unit VARCHAR(20) NOT NULL,
    quantity_on_hand NUMERIC(10,2) DEFAULT 0,
    reorder_level NUMERIC(10,2),
    supplier_id INTEGER REFERENCES Suppliers(supplier_id),
    cost_per_unit NUMERIC(8,2),
    storage_location VARCHAR(50)
);

-- Junction Table for Menu Items and Ingredients
CREATE TABLE Menu_Item_Ingredients (
    dish_id INTEGER REFERENCES Menu(dish_id),
    ingredient_id INTEGER REFERENCES Ingredients(ingredient_id),
    quantity NUMERIC(10,2) NOT NULL,
    PRIMARY KEY (dish_id, ingredient_id)
);

-- Staff Table
CREATE TABLE Staff (
    employee_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    position VARCHAR(50) NOT NULL,
    hourly_rate NUMERIC(8,2),
    hire_date DATE DEFAULT CURRENT_DATE,
    email VARCHAR(100),
    phone VARCHAR(20),
```

```
    emergency_contact VARCHAR(100),
    address VARCHAR(255),
    social_security VARCHAR(11), -- Consider encryption for production
    tax_withholding INTEGER
);

-- Shifts Table
CREATE TABLE Shifts (
    shift_id SERIAL PRIMARY KEY,
    employee_id INTEGER REFERENCES Staff(employee_id),
    shift_date DATE NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    role VARCHAR(50),
    section_assigned VARCHAR(50),
    status VARCHAR(20) DEFAULT 'Scheduled'
        CHECK (status IN ('Scheduled', 'Completed', 'Absent', 'Late'))
);

-- Orders Table
CREATE TABLE Orders (
    order_id SERIAL PRIMARY KEY,
    table_id INTEGER REFERENCES Tables(table_id),
    customer_id INTEGER REFERENCES Customers(customer_id),
    server_id INTEGER REFERENCES Staff(employee_id),
    order_date DATE DEFAULT CURRENT_DATE,
    order_time TIME DEFAULT CURRENT_TIME,
    status VARCHAR(20) DEFAULT 'Placed'
        CHECK (status IN ('Placed', 'Preparing', 'Served',
'Completed')),
    special_instructions TEXT,
    total_amount NUMERIC(10,2) DEFAULT 0,
    discount_amount NUMERIC(10,2) DEFAULT 0,
    tax_amount NUMERIC(10,2) DEFAULT 0,
    final_amount NUMERIC(10,2) DEFAULT 0
);

-- Order Items Table
CREATE TABLE Order_Items (
    order_item_id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES Orders(order_id),
    dish_id INTEGER REFERENCES Menu(dish_id),
    quantity INTEGER NOT NULL DEFAULT 1,
    price NUMERIC(10,2) NOT NULL,
    modifications TEXT,
    sent_to_kitchen_time TIMESTAMP,
    completed_time TIMESTAMP,
    CONSTRAINT valid_quantity CHECK (quantity > 0)
);

-- Payments Table
CREATE TABLE Payments (
    payment_id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES Orders(order_id),
    payment_date DATE DEFAULT CURRENT_DATE,
    payment_time TIME DEFAULT CURRENT_TIME,
    amount NUMERIC(10,2) NOT NULL,
    tip_amount NUMERIC(10,2) DEFAULT 0,
```

```
    payment_method VARCHAR(50) NOT NULL,
    card_last_four VARCHAR(4),
    is_split BOOLEAN DEFAULT FALSE,
    transaction_id VARCHAR(100),
    receipt_number VARCHAR(50)
);

-- Add some indexes for performance
CREATE INDEX idx_reservations_date ON Reservations(reservation_date);
CREATE INDEX idx_orders_date ON Orders(order_date);
CREATE INDEX idx_payments_date ON Payments(payment_date);
CREATE INDEX idx_order_items_order_id ON Order_Items(order_id);
CREATE INDEX idx_menu_category ON Menu(category_id);
CREATE INDEX idx_ingredients_supplier ON Ingredients(supplier_id);
CREATE INDEX idx_shifts_date ON Shifts(shift_date);


-------------------------------------------------------------
-- SAMPLE DATA INSERTS AND VIEWS FOR ADDITIONAL REPORTS
-------------------------------------------------------------

-- 4. Create a View to Show Inventory Status (ingredients in stock)
CREATE OR REPLACE VIEW inventory_status AS
SELECT
    ingredient_id,
    name,
    unit,
    quantity_on_hand,
    reorder_level,
    cost_per_unit,
    storage_location,
    CASE
        WHEN quantity_on_hand < reorder_level THEN TRUE
        ELSE FALSE
    END AS needs_reorder
FROM Ingredients
ORDER BY name;

-- Optional: Create a View for Staff Productivity Aggregation
CREATE OR REPLACE VIEW staff_productivity_extended AS
SELECT
    s.employee_id,
    s.first_name,
    s.last_name,
    COUNT(o.order_id) AS orders_served,
    COALESCE(SUM(o.final_amount), 0) AS total_sales,
    COALESCE(AVG(p.tip_amount), 0) AS average_tip
FROM Staff s
LEFT JOIN Orders o ON s.employee_id = o.server_id
LEFT JOIN Payments p ON o.order_id = p.order_id
GROUP BY s.employee_id, s.first_name, s.last_name
ORDER BY total_sales DESC;
```