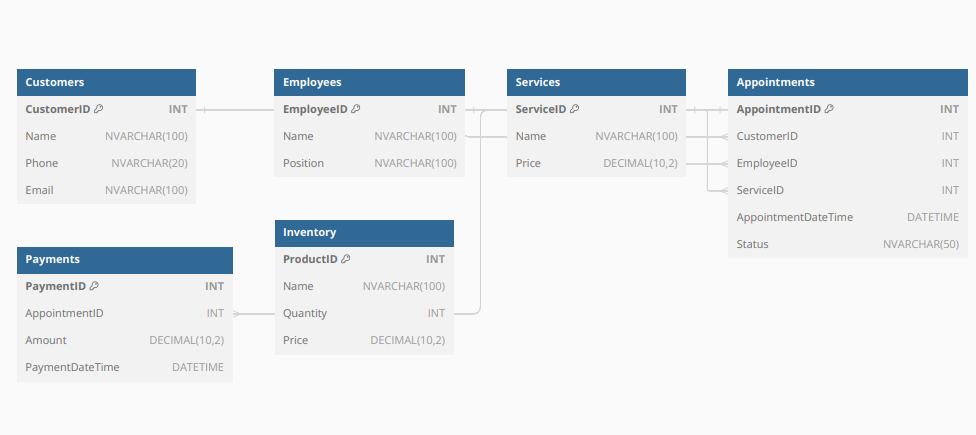
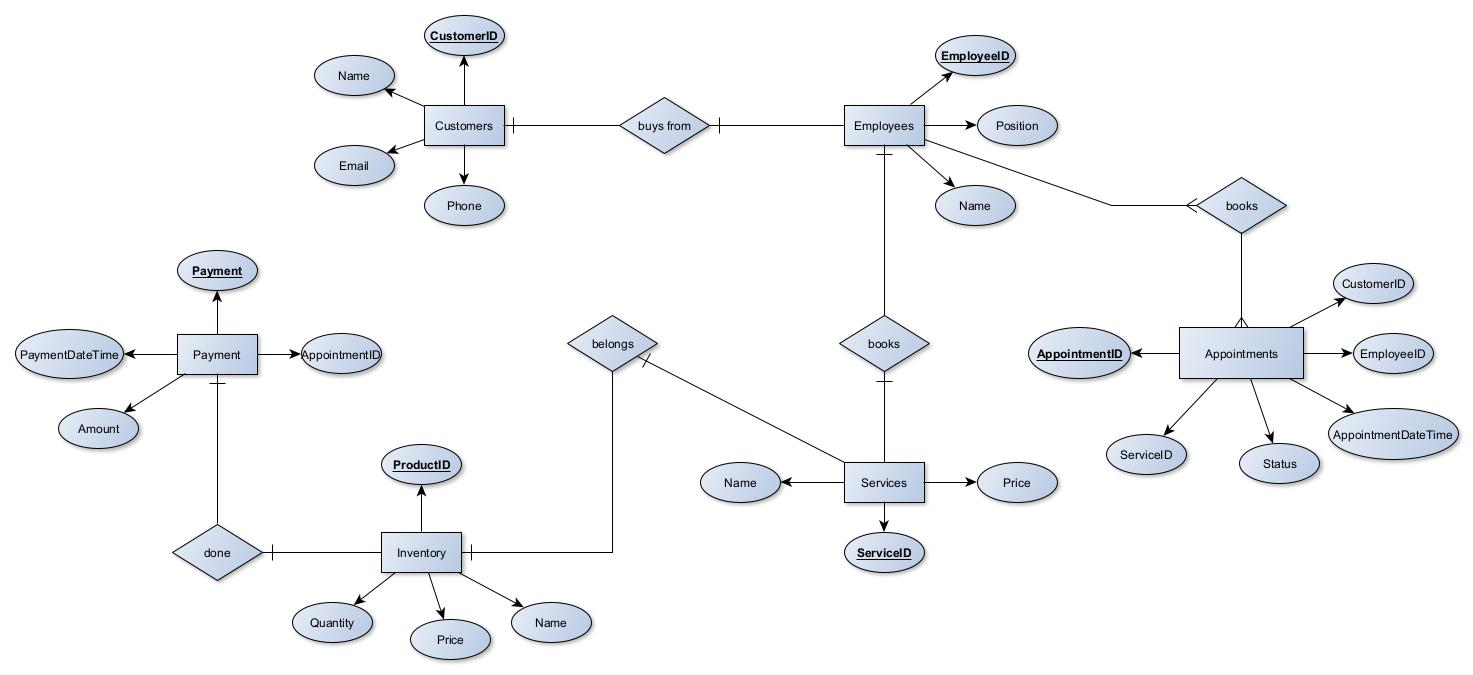
**Salon Booking System Database Design Documentation**

This document outlines the design of a Salon Booking System database, including tables, relationships, queries, functions, triggers, and a stored procedure.

Schema Diagram.



ER Diagram.



⁜ Tables

Customers: Stores customer information like ID, name, phone number, and email.

Employees: Stores employee information like ID, name, and position.

Services: Stores information about services offered, including ID, name, and price.

Appointments: Stores appointment details like ID, customer ID (foreign key referencing Customers table), employee ID (foreign key referencing Employees table), service ID (foreign key referencing Services table), appointment date and time, and appointment status.

Payments: Stores payment information for appointments, including ID, appointment ID (foreign key referencing Appointments table), amount paid, and payment date and time.

Inventory: Stores product information used for appointments, including ID, name, quantity, and price.

TABLES

**## Customers Table**

| Column       | Data Type     | Description              |

|--------------|---------------|--------------------------|

| CustomerID   | INT           | Primary key              |

| Name         | NVARCHAR(100) | Customer's name          |

| Phone        | NVARCHAR(20)  | Customer's phone number  |

| Email        | NVARCHAR(100) | Customer's email address |

**### Sample Data**

| CustomerID | Name           | Phone         | Email                |

|------------|----------------|---------------|----------------------|

| 1          | John Doe       | 123-456-7890  | john@example.com     |

| 2          | Jane Smith     | 987-654-3210  | jane@example.com     |

| 3          | Alice Johnson  | 555-123-4567  | alice@example.com    |

| 4          | Bob Brown      | 777-888-9999  | bob@example.com      |

| 5          | Emily Davis    | 111-222-3333  | emily@example.com    |

**## Employees Table**

| Column       | Data Type     | Description              |

|--------------|---------------|--------------------------|

| EmployeeID   | INT           | Primary key              |

| Name         | NVARCHAR(100) | Employee's name          |

| Position     | NVARCHAR(100) | Employee's position      |

**### Sample Data**

| EmployeeID | Name           | Position           |

|------------|----------------|--------------------|

| 1          | Michael Scott  | Manager            |

| 2          | Dwight Schrute | Assistant Manager  |

| 3          | Jim Halpert    | Stylist            |

| 4          | Pam Beesly     | Receptionist       |

| 5          | Angela Martin  | Stylist            |

**## Services Table**

| Column       | Data Type     | Description              |

|--------------|---------------|--------------------------|

| ServiceID    | INT           | Primary key              |

| Name         | NVARCHAR(100) | Service name             |

| Price        | DECIMAL(10,2) | Service price            |

**### Sample Data**

| ServiceID | Name          | Price  |

|-----------|---------------|--------|

| 1         | Haircut       | 20.00  |

| 2         | Hair Color    | 50.00  |

| 3         | Manicure      | 30.00  |

| 4         | Pedicure      | 35.00  |

| 5         | Massage       | 60.00  |

**## Appointments Table**

| Column             | Data Type     | Description                          |

|--------------------|---------------|--------------------------------------|

| AppointmentID      | INT           | Primary key, Auto-increment           |

| CustomerID         | INT           | Foreign key referencing Customers    |

| EmployeeID         | INT           | Foreign key referencing Employees    |

| ServiceID          | INT           | Foreign key referencing Services     |

| AppointmentDateTime| DATETIME      | Date and time of appointment         |

| Status             | NVARCHAR(50) | Appointment status (e.g., Pending)   |

**### Sample Data**

| AppointmentID | CustomerID | EmployeeID | ServiceID | AppointmentDateTime      | Status   |

|---------------|------------|------------|-----------|--------------------------|----------|

| 1             | 1          | 3          | 1         | 2024-03-21 10:00:00      | Pending  |

**## Payments Table**

| Column             | Data Type     | Description                          |

|--------------------|---------------|--------------------------------------|

| PaymentID          | INT           | Primary key                          |

| AppointmentID      | INT           | Foreign key referencing Appointments|

| Amount             | DECIMAL(10,2) | Payment amount                       |

| PaymentDateTime    | DATETIME      | Date and time of payment             |

**### Sample Data**

| PaymentID | AppointmentID | Amount  | PaymentDateTime      |

|-----------|---------------|---------|----------------------|

| 1         | 1             | 0.00    | 2024-03-21 10:05:00  |

**## Inventory Table**

| Column       | Data Type     | Description                          |

|--------------|---------------|--------------------------------------|

| ProductID    | INT           | Primary key                          |

| Name         | NVARCHAR(100) | Product name                         |

| Quantity     | INT           | Quantity available in inventory       |

| Price        | DECIMAL(10,2) | Price per unit                       |

**### Sample Data**

| ProductID | Name           | Quantity | Price  |

|-----------|----------------|----------|--------|

| 1         | Shampoo        | 50       | 10.00  |

| 2         | Conditioner    | 40       | 8.00   |

| 3         | Hair Gel       | 30       | 12.00  |

| 4         | Nail Polish    | 100      | 5.00   |

| 5         | Massage Oil    | 20       | 15.00  |

stomer's name          |

| Phone        | NVARCHAR(20)  | Customer's phone number  |

| Email        | NVARCHAR(100) | Customer's email address |

**### Sample Data**

| CustomerID | Name           | Phone         | Email                |

|------------|----------------|---------------|----------------------|

| 1          | John Doe       | 123-456-7890  | john@example.com     |

| 2          | Jane Smith     | 987-654-3210  | jane@example.com     |

| 3          | Alice Johnson  | 555-123-4567  | alice@example.com    |

| 4          | Bob Brown      | 777-888-9999  | bob@example.com      |

| 5          | Emily Davis    | 111-222-3333  | emily@example.com    |

**## Employees Table**

| Column       | Data Type     | Description              |

|--------------|---------------|--------------------------|

| EmployeeID   | INT           | Primary key              |

| Name         | NVARCHAR(100) | Employee's name          |

| Position     | NVARCHAR(100) | Employee's position      |

**### Sample Data**

| EmployeeID | Name           | Position           |

|------------|----------------|--------------------|

| 1          | Michael Scott  | Manager            |

| 2          | Dwight Schrute | Assistant Manager  |

| 3          | Jim Halpert    | Stylist            |

| 4          | Pam Beesly     | Receptionist       |

| 5          | Angela Martin  | Stylist            |

**## Services Table**

| Column       | Data Type     | Description              |

|--------------|---------------|--------------------------|

| ServiceID    | INT           | Primary key              |

| Name         | NVARCHAR(100) | Service name             |

| Price        | DECIMAL(10,2) | Service price            |

**### Sample Data**

| ServiceID | Name          | Price  |

|-----------|---------------|--------|

| 1         | Haircut       | 20.00  |

| 2         | Hair Color    | 50.00  |

| 3         | Manicure      | 30.00  |

| 4         | Pedicure      | 35.00  |

| 5         | Massage       | 60.00  |

**## Appointments Table**

| Column             | Data Type     | Description                          |

|--------------------|---------------|--------------------------------------|

| AppointmentID      | INT           | Primary key, Auto-increment           |

| CustomerID         | INT           | Foreign key referencing Customers    |

| EmployeeID         | INT           | Foreign key referencing Employees    |

| ServiceID          | INT           | Foreign key referencing Services     |

| AppointmentDateTime| DATETIME      | Date and time of appointment         |

| Status             | NVARCHAR(50) | Appointment status (e.g., Pending)   |

**### Sample Data**

| AppointmentID | CustomerID | EmployeeID | ServiceID | AppointmentDateTime      | Status   |

|---------------|------------|------------|-----------|--------------------------|----------|

| 1             | 1          | 3          | 1         | 2024-03-21 10:00:00      | Pending  |

**## Payments Table**

| Column             | Data Type     | Description                          |

|--------------------|---------------|--------------------------------------|

| PaymentID          | INT           | Primary key                          |

| AppointmentID      | INT           | Foreign key referencing Appointments|

| Amount             | DECIMAL(10,2) | Payment amount                       |

| PaymentDateTime    | DATETIME      | Date and time of payment             |

**### Sample Data**

| PaymentID | AppointmentID | Amount  | PaymentDateTime      |

|-----------|---------------|---------|----------------------|

| 1         | 1             | 0.00    | 2024-03-21 10:05:00  |

**## Inventory Table**

| Column       | Data Type     | Description                          |

|--------------|---------------|--------------------------------------|

| ProductID    | INT           | Primary key                          |

| Name         | NVARCHAR(100) | Product name                         |

| Quantity     | INT           | Quantity available in inventory       |

| Price        | DECIMAL(10,2) | Price per unit                       |

**### Sample Data**

| ProductID | Name           | Quantity | Price  |

|-----------|----------------|----------|--------|

| 1         | Shampoo        | 50       | 10.00  |

| 2         | Conditioner    | 40       | 8.00   |

| 3         | Hair Gel       | 30       | 12.00  |

| 4         | Nail Polish    | 100      | 5.00   |

| 5         | Massage Oil    | 20       | 15.00  |

⁜ Rational:

Separate tables for customers, employees, and services improve data organization and reduce redundancy.

Foreign keys ensure data integrity between tables.

The Appointments table captures core appointment details and references relevant personnel and services.

The Payments table tracks financial transactions for appointments.

The Inventory table manages product stock used during appointments.

⁜ Queries

Sample Queries: Several queries are included to retrieve data from different tables, demonstrating data access capabilities.

Employee Availability: A query retrieves employees available for a specific service during a time slot.

Total Appointment Cost: A query calculates the total cost for an appointment by summing the prices of included services.

Customer Appointments: A query retrieves customer names, appointment times, and service names for all appointments.

⁜ Functions

CalculateTotalAppointmentCost: This function calculates the total cost of an appointment by summing the prices of associated services.

Rationale:

Functions encapsulate reusable calculations, improving code maintainability.

This function simplifies cost retrieval for appointments.

⁜ Triggers

UpdateInventoryOnAppointment: This trigger automatically reduces product quantity in the Inventory table when a new appointment is created, ensuring inventory reflects service usage.

Rationale:

Triggers automate actions based on database events, streamlining inventory management.

This trigger maintains accurate inventory levels.

⁜ Stored Procedure

BookAppointment: This stored procedure simplifies booking appointments by handling data insertion into the Appointments table and potentially updating inventory, all within a transaction.

```markdown

**## Stored Procedures, Triggers, and Functions**

**### BookAppointment Stored Procedure**

The `BookAppointment` stored procedure is used to book an appointment in the salon booking system. It takes input parameters such as CustomerID, EmployeeID, ServiceID, and AppointmentDateTime, and inserts a new record into the 'Appointments' table with the provided details.

CREATE PROCEDURE BookAppointment

    @CustomerID INT,

    @EmployeeID INT,

    @ServiceID INT,

    @AppointmentDateTime DATETIME

AS

BEGIN

    INSERT INTO Appointments (CustomerID, EmployeeID, ServiceID, AppointmentDateTime, Status)

    VALUES (@CustomerID, @EmployeeID, @ServiceID, @AppointmentDateTime, 'Pending');

END;

```

**### UpdateInventoryOnAppointment Trigger**

The `UpdateInventoryOnAppointment` trigger is fired after an appointment is inserted into the 'Appointments' table. It updates the inventory based on the service booked by decrementing the quantity of the corresponding product.

```sql

CREATE TRIGGER UpdateInventoryOnAppointment

ON Appointments

AFTER INSERT

AS

BEGIN

    UPDATE Inventory

    SET Quantity = Quantity - 1

    WHERE ProductID IN (

        SELECT ProductID FROM Services

        WHERE ServiceID IN (

            SELECT ServiceID FROM inserted

        )

    );

END;

```

**### CalculateTotalAppointmentCost Function**

The `CalculateTotalAppointmentCost` function calculates the total cost of an appointment based on the appointment ID by summing up the prices of the services associated with the appointment.

```sql

CREATE FUNCTION CalculateTotalAppointmentCost (@AppointmentID INT)

RETURNS DECIMAL(10, 2)

AS

BEGIN

    DECLARE @TotalCost DECIMAL(10, 2);

    SELECT @TotalCost = SUM(s.Price)

    FROM Appointments a

    INNER JOIN Services s ON a.ServiceID = s.ServiceID

    WHERE a.AppointmentID = @AppointmentID;

    RETURN @TotalCost;

END;

```

Rationale:

Stored procedures promote code reusability and data integrity by encapsulating complex operations.

This procedure streamlines appointment booking and inventory updates.

⁜ Transactions

A sample transaction demonstrates booking an appointment, updating inventory, and handling potential errors.

```sql

BEGIN TRANSACTION;

DECLARE @AppointmentID INT;

-- Book the appointment

INSERT INTO Appointments (CustomerID, EmployeeID, ServiceID, AppointmentDateTime, Status)

VALUES (1, 3, 1, '2024-03-21 10:00:00', 'Pending');

-- Get the AppointmentID of the newly inserted appointment

SET @AppointmentID = SCOPE\_IDENTITY();

-- Update inventory based on the service booked

UPDATE Inventory

SET Quantity = Quantity - 1

WHERE ProductID IN (

    SELECT ProductID FROM Services

    WHERE ServiceID = 1 -- Assuming ServiceID for "Haircut"

);

-- Check if any error occurred

IF @@ERROR <> 0

BEGIN

    ROLLBACK TRANSACTION;

    PRINT 'An error occurred. Rolling back transaction.';

END

ELSE

BEGIN

    COMMIT TRANSACTION;

    PRINT 'Appointment booked successfully.';

END;

```

Rationale:

Transactions ensure data consistency by treating a series of database operations as a single unit.

This example showcases error handling and transaction management.

⁜ Additional Notes

This design uses reasonable data types for each table column.

Comments can be added to the SQL code for better readability and maintainability.

Security measures should be implemented to restrict unauthorized data access.

This documentation provides a foundation for understanding the Salon Booking System database design and its functionalities. You can customize and extend this design based on specific business requirements.