Case Study: Designing a Relational Database for IKIGAI Barbershop

By: Rey Anthony Rodriguez | 000931509

DATA037- 008

As a data science student with no prior experience in SQL, I set out on a journey to learn how to design and implement a relational database from scratch. I wanted to move beyond theory and develop a real-world database system that could support a business's day-to-day operations. This case study represents an opportunity to apply theoretical knowledge by designing a fully functional relational database for a small business, or in my case, a non-existent business which I named IKIGAI – a Japanese belief meaning "reason for being", a value I would like to uphold and share with others too!

Why a Barbershop?

I chose a barbershop for this case study because it is a small business with structured yet diverse data requirements. A barbershop manages customer information, appointment scheduling, employee records, services, product inventory, financial transactions, and customer feedback—all of which make it an ideal candidate for a well-designed relational database. The project allowed me to explore key database concepts while creating something practical and useful for small business operations.

Mission and Objectives

Before starting any database project, it's crucial to define a mission and set clear objectives that align with it. This helps maintain focus throughout the development process and ensures that the final product meets the intended needs.

Mission Statement:

For me, the mission of this project was to create a centralized and efficient relational database system for a barbershop, designed to streamline operations and enhance decision-making.

The objectives I set were:

- **Optimize Appointment Scheduling** – Simplify the process of managing customer bookings, assigning barbers, and tracking appointments.
- **Enhance Inventory Management** – Ensure that products are properly tracked and suppliers are easily linked to products for efficient inventory control.
- **Automate Revenue and Expense Tracking** – Develop a system for tracking payments, revenue, and expenses in real-time to provide insightful financial data.

By having these clear objectives, I was able to structure the database design and features in a way that directly aligned with the overall mission, making sure the database served its intended purpose without unnecessary complexity.

Identifying the Requirements

To create a well-structured database, I first needed to identify what information the barbershop required. This involved understanding the different components of the business and the types of data each would generate. I identified the following core requirements:

- Customers should be able to book appointments with employees for specific services.
- Employees should be associated with a specific branch and have defined roles.
- Services and products should be managed efficiently, including inventory tracking.
- Payments should be recorded for every appointment.
- Customer feedback should be collected through reviews.

With these requirements in mind, I moved on to identifying the **entities** that would make up the database.

Identifying Preliminary Entities and Finalized Tables

In database design, **entities** represent the core objects that hold data. Once I identified the entities, I determined the **attributes** (pieces of information) each entity should store. Here's the final list of entities and their attributes:

1. **Customers**: Individuals who use the barbershop's services.
2. **Employees**: Barbers or staff.
3. **Appointments**: Scheduled services.
4. **Services**: Different grooming services offered.
5. **Products**: Items sold or used in the barbershop, like shampoos or razors.
6. **Suppliers**: Companies or individuals providing products.
7. **Payments**: Financial transactions for services or products.
8. **Branches**: Different barbershop locations.
9. **Reviews**: Customer feedback on services,

After identifying the entities, I then focused on their attributes, or the specific pieces of information that each entity would store. I considered what data would be most relevant for the barbershop's operations and how each attribute would contribute to managing customer relationships, appointments, inventory, and finances effectively. By selecting the right attributes, I ensured that the database could capture all necessary information without overcomplicating the structure.

Customers • name • phone • email • preferences	**Employees** • name • phone • email • role • skills • salary	**Appointments** • appointment date • customer name • employee name • service name • product name • payment details • status
Services • name • price • duration	**Products** • name • quantity • price • supplier name	**Suppliers** • name • phone • address
Payment Mode • appointment details • payment method	**Branches** • name • location	**Reviews** • appointment details • rating • comments

Now that the attributes for each entity have been identified, I can move forward with finalizing the table structures by defining input conditions. These conditions ensure data integrity, consistency, and accuracy within the database.

Input conditions include:

- **Data Types**: Choosing the appropriate data type for each attribute (e.g., VARCHAR for names, DECIMAL for prices, INT for Integer numbers which are commonly used for IDs).
- **Constraints**: Implementing rules such as NOT NULL (to prevent missing values), UNIQUE (to avoid duplicates), and CHECK (to enforce valid entries).
- **Foreign Keys**: Establishing connections between tables to maintain referential integrity.

Finalized Tables with Key Constraints

Customers Table

Field	Data Type
CustomerID [PK]	INT
FirstName	VARCHAR
LastName	VARCHAR
Phone	VARCHAR
Email	VARCHAR
Preferences	TEXT

Products Table

Field	Data Type
ProductID [PK]	INT
Name	VARCHAR
Quantity	INT
Price	DECIMAL
SupplierID	INT

Employees Table

Field	Data Type
EmployeeID [PK]	INT
FirstName	VARCHAR
LastName	VARCHAR
Phone	VARCHAR
Email	VARCHAR
Role	VARCHAR
Skills	TEXT
Salary	DECIMAL
BranchID	INT

Appointments Table

Field	Data Type
AppointmentID [PK]	INT
CustomerID	INT
EmployeeID	INT
ServiceID [CPK]	INT
ProductID [CPK]	INT
PaymentID	INT
datetime	DATETIME [NOT NULL]
status	INT

Branch Table

Field	Data Type
BranchID [PK]	INT
Name	VARCHAR
Phone	VARCHAR
Address	VARCHAR

Suppliers Table

Field	Data Type
SupplierID [PK]	INT
Name	VARCHAR
Phone	VARCHAR
Address	VARCHAR

Services Table

Field	Data Type
ServiceID [PK]	INT
ServiceName	VARCHAR
Price	DECIMAL
Duration	INT

Reviews Table

Field	Data Type
ReviewID [PK]	INT
Rating	TEXT
Comments	TEXT

Payment Mode Table

Field	Data Type
PaymentID [PK]	INT
PaymentMode	VARCHAR

Understanding Relationships Between Tables

Relational databases rely on defining relationships between tables to eliminate redundancy and ensure data consistency. Based on my understanding of the business and what business rules I want to set, I have formed the following table relationships.

1. One-to-One (1:1) Relationships

- Appointments → Reviews: Each appointment can have only one review, and each review corresponds to one appointment.
- Appointments → Payments: Each appointment is linked to exactly one payment method.

2. One-to-Many (1:M) Relationships

- Customers → Appointments: A customer can have multiple appointments, but each appointment belongs to only one customer.
- Employees → Appointments: An employee (barber) can handle multiple appointments, but each appointment is assigned to only one employee.
- Branches → Employees: A branch can have multiple employees, but each employee works at only one branch.
- Suppliers → Products: A supplier can provide multiple products, but each product is sourced from only one supplier.

3. Many-to-Many (M:N) Relationships

- Appointments ↔ Services:
    - A single appointment can include multiple services (e.g., haircut + beard trim).
    - A single service (e.g., haircut) can be booked in multiple appointments.
- Appointments ↔ Products:
    - Customers may purchase multiple products during an appointment.
    - A single product (e.g., hair gel) can be sold in multiple appointments.

The one-to-one and one-to-many relationships in my database seemed straightforward. However, I ran into a challenge when trying to accommodate the many-to-many relationships between Appointments and Services as well as Appointments and Products.

At first, I considered creating a composite primary key in the Appointments table using appointment_id, service_id, and product_id. While this seemed like a simple approach, it led to several problems:

1. It restricts flexibility – An appointment would be forced to have only one service and one product, which is unrealistic.

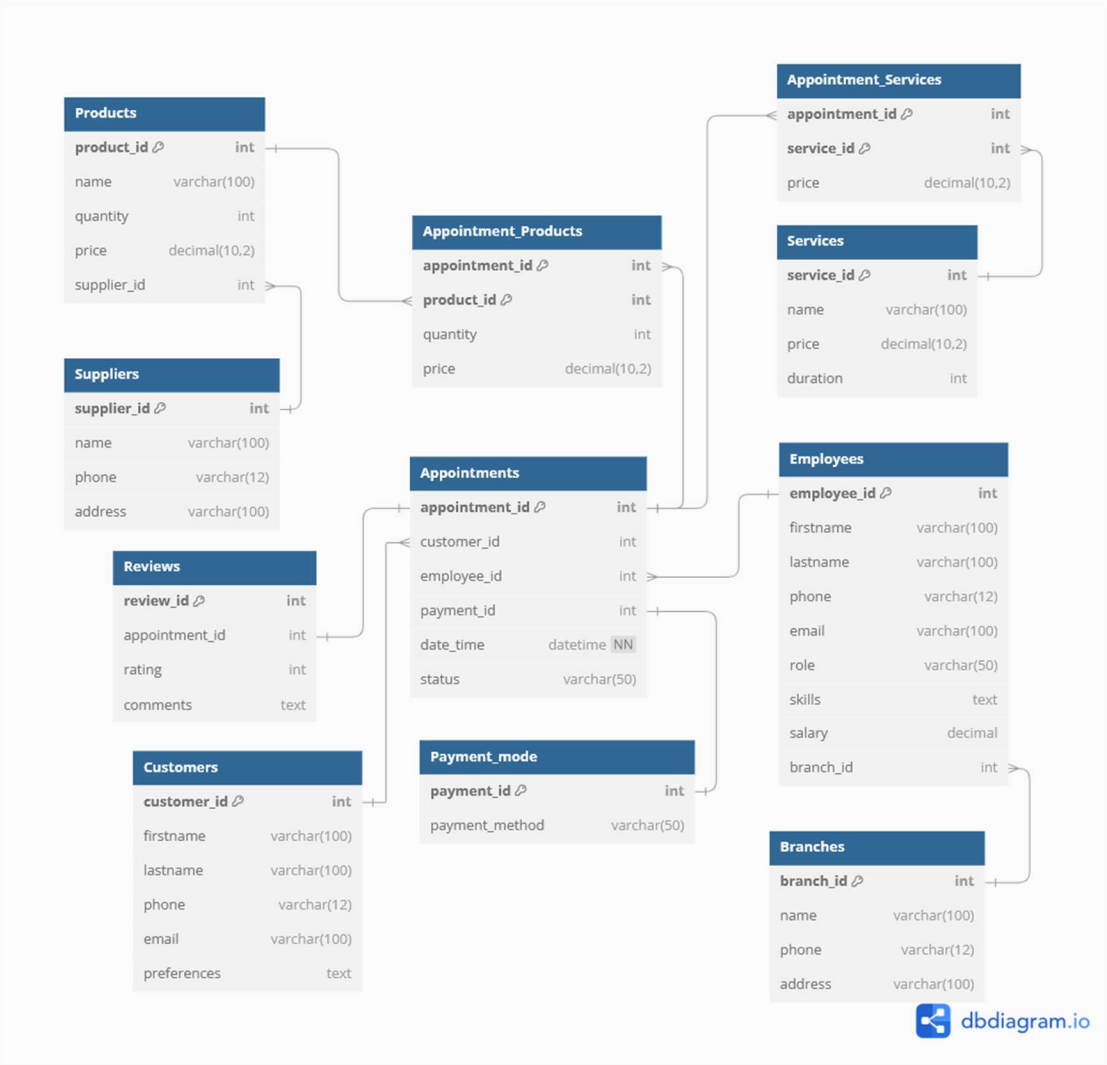
2. It introduces redundancy – If a customer books an appointment with multiple services and multiple products, the database will need to duplicate the entire appointment record, making it inefficient and difficult to manage.

After some research, I discovered the proper solution: using junction tables. By introducing two new tables—Appointment_Services and Appointment_Products—I was able to properly model the many-

to-many relationships while avoiding redundancy. These tables store associations between appointments, services, and products separately, ensuring data consistency, flexibility, and efficiency. Inside each of the junction tables, I later added a “price” attribute to record the price of the product or service at the time of the appointment.

Now that the entities and tables were identified, all that is left to do is visualize it using an Entity Relationship (ER) Diagram. To design my ER diagram, I used dbdiagram.io, a simple yet powerful tool for visualizing database structures. It allowed me to define tables, set primary and foreign keys, and establish relationships using an intuitive text-based schema. As I iterated through different database designs, dbdiagram.io made it easy to update and refine my structure, ensuring that the one-to-one, one-to-many, and many-to-many relationships were correctly implemented. The ability to quickly visualize changes helped me identify issues, such as the need for junction tables, and ensure a well-structured, normalized database before actual implementation.

Entity Relationship Diagram



Database Creation Process

Using SQL, I translated my ER diagram into a relational database. The key steps were:

- Table Creation:
  - Defined tables with appropriate data types.
  - Established primary keys for unique identification.
  - Implemented foreign keys to define relationships.
- Data Insertion:
  - Populated tables with sample data to test functionality.
- Query Optimization:
  - Created indexes to speed up queries.
  - Used joins to connect data efficiently.

Below is exactly as written in my Azure Data Studios.

```
-- Create and use database name IkigaiBarbershop
DROP DATABASE IkigaiBarbershop;
CREATE DATABASE IkigaiBarbershop;
USE IkigaiBarbershop;

-- Customers Table
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    firstname VARCHAR(100),
    lastname VARCHAR(100),
    phone VARCHAR(12),
    email VARCHAR(100),
    preferences TEXT
);

-- Branches Table
CREATE TABLE Branches (
    branch_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    phone VARCHAR(12),
    address VARCHAR(100)
);

-- Employees Table
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY AUTO_INCREMENT,
    firstname VARCHAR(100),
    lastname VARCHAR(100),
    phone VARCHAR(12),
    email VARCHAR(100),
    role VARCHAR(50),
    skills TEXT,
    salary DECIMAL(10,2),
    branch_id INT,
    FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);
```

```sql
-- Services Table
CREATE TABLE Services (
    service_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    price DECIMAL(10,2),
    duration INT
);

-- Suppliers Table
CREATE TABLE Suppliers (
    supplier_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    phone VARCHAR(12),
    address VARCHAR(100)
);

-- Products Table
CREATE TABLE Products (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    quantity INT,
    price DECIMAL(10,2),
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id)
);

-- Payment Mode Table
CREATE TABLE Payment_mode (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    payment_method VARCHAR(50)
);
-- Appointments Table
CREATE TABLE Appointments (
    appointment_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    employee_id INT,
    payment_id INT,
    date_time DATETIME NOT NULL,
    status VARCHAR(50),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id),
    FOREIGN KEY (payment_id) REFERENCES Payment_mode(payment_id)
);

-- Appointment_Services Junction Table
CREATE TABLE Appointment_Services (
    appointment_id INT,
    service_id INT,
    price DECIMAL(10,2),
    PRIMARY KEY (appointment_id, service_id),
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id),
    FOREIGN KEY (service_id) REFERENCES Services(service_id)
);
```

```sql
-- Appointment_Products Junction Table
CREATE TABLE Appointment_Products (
    appointment_id INT,
    product_id INT,
    quantity INT,
    price DECIMAL(10,2),
    PRIMARY KEY (appointment_id, product_id),
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
-- Reviews Table
CREATE TABLE Reviews (
    review_id INT PRIMARY KEY AUTO_INCREMENT,
    appointment_id INT,
    rating INT,
    comments TEXT,
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id)
);

-- Adding sample data
INSERT INTO Customers (customer_id, firstname, lastname, phone, email, preferences) VALUES
(1, 'Alice', 'Johnson', '1234567890', 'alice@example.com', 'Short Haircut'),
(2, 'Bob', 'Smith', '2345678901', 'bob@example.com', 'Beard Trim'),
(3, 'Charlie', 'Brown', '3456789012', 'charlie@example.com', 'Long Haircut'),
(4, 'David', 'Miller', '4567890123', 'david@example.com', 'Shave'),
(5, 'Eve', 'Davis', '5678901234', 'eve@example.com', 'Hair Coloring'),
(6, 'Frank', 'White', '6789012345', 'frank@example.com', 'Haircut and Beard Trim'),
(7, 'Grace', 'Hall', '7890123456', 'grace@example.com', 'Short Haircut'),
(8, 'Hank', 'Moore', '8901234567', 'hank@example.com', 'Buzz Cut'),
(9, 'Ivy', 'Taylor', '9012345678', 'ivy@example.com', 'Hair Straightening'),
(10, 'Jack', 'Wilson', '0123456789', 'jack@example.com', 'Haircut and Beard Trim'),
(11, 'Kara', 'Anderson', '1231231234', 'kara@example.com', 'Hair Coloring'),
(12, 'Leo', 'Thomas', '2342342345', 'leo@example.com', 'Shave'),
(13, 'Mia', 'Harris', '3453453456', 'mia@example.com', 'Layered Haircut'),
(14, 'Noah', 'Martin', '4564564567', 'noah@example.com', 'Beard Trim'),
(15, 'Olivia', 'Clark', '5675675678', 'olivia@example.com', 'Haircut'),
(16, 'Paul', 'Rodriguez', '6786786789', 'paul@example.com', 'Straight Haircut');
INSERT INTO Branches (name, phone, address) VALUES
('Downtown Barbershop', '5559998888', '789 Elm St'),
('Uptown Salon', '5557776666', '456 Oak St'),
('Midtown Cuts', '5556665555', '123 Pine St'),
('Elite Barber Lounge', '5553332222', '890 Maple St'),
('Luxury Hair Studio', '5551112223', '321 Birch St'),
('The Classic Barber', '5554445556', '741 Spruce St'),
('Fade Masters', '5559990001', '159 Redwood St'),
('The Trendy Stylist', '5558887774', '258 Cherry St');
INSERT INTO Employees (employee_id, firstname, lastname, phone, email, role, skills, salary,
branch_id) VALUES
(1, 'John', 'Doe', '1112223333', 'john@example.com', 'Barber', 'Fades, Beard Trim', 2500.00,
1),
(2, 'Jane', 'Smith', '4445556666', 'jane@example.com', 'Stylist', 'Hair Coloring, Highlights',
2700.00, 2),
(3, 'Mike', 'Brown', '5556667777', 'mike@example.com', 'Barber', 'Haircuts, Shaving', 2600.00,
1),
```

```
(4, 'Anna', 'Davis', '6667778888', 'anna@example.com', 'Hairdresser', 'Layered Haircut, Hair
Coloring', 2800.00, 2),
(5, 'Rick', 'Moore', '7778889999', 'rick@example.com', 'Stylist', 'Hair Straightening',
2500.00, 1),
(6, 'Susan', 'Lee', '8889990000', 'susan@example.com', 'Barber', 'Buzz Cut, Shave', 2400.00,
3),
(7, 'Tom', 'Anderson', '9990001111', 'tom@example.com', 'Barber', 'Haircuts, Fades', 2550.00,
2),
(8, 'Emily', 'Harris', '0001112222', 'emily@example.com', 'Hairdresser', 'Highlights, Hair
Coloring', 2750.00, 3),
(9, 'Jake', 'Miller', '1113334444', 'jake@example.com', 'Barber', 'Haircuts, Beard Trim',
2600.00, 1),
(10, 'Laura', 'Wilson', '2224445555', 'laura@example.com', 'Stylist', 'Hair Styling,
Braiding', 2700.00, 3),
(11, 'Sam', 'Clark', '3335556666', 'sam@example.com', 'Barber', 'Fades, Haircuts', 2500.00,
2),
(12, 'Diana', 'Rodriguez', '4446667777', 'diana@example.com', 'Hairdresser', 'Hair Coloring,
Blowdry', 2650.00, 3),
(13, 'Kevin', 'White', '5557778888', 'kevin@example.com', 'Barber', 'Shaving, Beard Styling',
2550.00, 1),
(14, 'Sophia', 'Martin', '6668889999', 'sophia@example.com', 'Stylist', 'Hair Highlights,
Perm', 2750.00, 2),
(15, 'Oscar', 'Taylor', '7779990000', 'oscar@example.com', 'Hairdresser', 'Straight Haircuts',
2600.00, 1),
(16, 'Rachel', 'Thomas', '8880001111', 'rachel@example.com', 'Stylist', 'Braiding, Haircuts',
2700.00, 3);
INSERT INTO Services (service_id, name, price, duration) VALUES
(1, 'Haircut', 20.00, 30),
(2, 'Beard Trim', 15.00, 15),
(3, 'Hair Coloring', 50.00, 60),
(4, 'Shave', 10.00, 10),
(5, 'Hair Straightening', 60.00, 90),
(6, 'Buzz Cut', 15.00, 20),
(7, 'Layered Haircut', 30.00, 45),
(8, 'Blow Dry', 25.00, 30),
(9, 'Perm', 70.00, 120),
(10, 'Fades', 25.00, 30),
(11, 'Braiding', 40.00, 60),
(12, 'Hair Highlights', 65.00, 90),
(13, 'Scalp Treatment', 35.00, 45),
(14, 'Keratin Treatment', 80.00, 120),
(15, 'Hair Mask Therapy', 45.00, 60),
(16, 'Hair Rebonding', 90.00, 150);
INSERT INTO Suppliers (name, phone, address) VALUES
('ABC Barber Supplies', '5551234567', '123 Market St'),
('XYZ Grooming Essentials', '5557654321', '456 Main St'),
('Best Hair Products Inc.', '5553456789', '789 Salon Ave'),
('Luxury Hair Goods', '5556789012', '159 Barber Blvd'),
('Premier Styling Tools', '5557890123', '357 Cutters Rd'),
('Top Notch Clippers', '5558901234', '951 Trim Ln'),
('Elite Haircare', '5559012345', '852 Groomer St'),
('High-End Scissors Co.', '5550123456', '741 Stylists Ct');
INSERT INTO Products (product_id, name, quantity, price, supplier_id) VALUES
(1, 'Shampoo', 50, 10.00, 1),
```

```
(2, 'Beard Oil', 30, 15.00, 2),
(3, 'Hair Gel', 40, 12.00, 1),
(4, 'Razor Blades', 100, 5.00, 3),
(5, 'Conditioner', 60, 10.00, 1),
(6, 'Hair Spray', 25, 18.00, 2),
(7, 'Hair Wax', 35, 14.00, 1),
(8, 'Hair Mousse', 20, 22.00, 3),
(9, 'Scalp Oil', 15, 30.00, 2),
(10, 'Hair Serum', 50, 25.00, 3),
(11, 'Hair Color Kit', 40, 35.00, 1),
(12, 'Bleaching Powder', 30, 20.00, 2),
(13, 'Hair Growth Tonic', 20, 40.00, 3),
(14, 'Hair Vitamin Capsules', 25, 50.00, 1),
(15, 'Heat Protectant Spray', 45, 28.00, 2),
(16, 'Leave-in Conditioner', 30, 32.00, 3);
INSERT INTO Payment_mode (payment_id, payment_method) VALUES
(1, 'Credit Card'),
(2, 'Debit Card'),
(3, 'Cash'),
(4, 'Credit Card'),
(5, 'Debit Card'),
(6, 'Cash'),
(7, 'Credit Card'),
(8, 'Debit Card'),
(9, 'Cash'),
(10, 'Credit Card'),
(11, 'Debit Card'),
(12, 'Cash'),
(13, 'Credit Card'),
(14, 'Debit Card'),
(15, 'Cash'),
(16, 'Credit Card');
INSERT INTO Appointments (appointment_id, customer_id, employee_id, payment_id, date_time,
status) VALUES
(1, 1, 1, 1, '2025-02-10 09:00:00', 'Completed'),
(2, 2, 2, 2, '2025-02-10 09:30:00', 'Booked'),
(3, 3, 3, 3, '2025-02-10 10:00:00', 'Cancelled'),
(4, 4, 4, 4, '2025-02-10 10:30:00', 'Completed'),
(5, 5, 5, 5, '2025-02-10 11:00:00', 'Booked'),
(6, 6, 6, 6, '2025-02-10 11:30:00', 'Cancelled'),
(7, 7, 7, 7, '2025-02-10 12:00:00', 'Completed'),
(8, 8, 8, 8, '2025-02-10 12:30:00', 'Booked'),
(9, 9, 9, 9, '2025-02-10 13:00:00', 'Completed'),
(10, 10, 10, 10, '2025-02-10 13:30:00', 'Cancelled'),
(11, 11, 11, 11, '2025-02-10 14:00:00', 'Booked'),
(12, 12, 12, 12, '2025-02-10 14:30:00', 'Completed'),
(13, 13, 13, 13, '2025-02-10 15:00:00', 'Booked'),
(14, 14, 14, 14, '2025-02-10 15:30:00', 'Cancelled'),
(15, 15, 15, 15, '2025-02-10 16:00:00', 'Completed'),
(16, 16, 16, 16, '2025-02-10 16:30:00', 'Booked');
```

Using Views to Optimize Data Access

1. View: Appointment Details View

Scenario: The Appointment Details View is used to provide a comprehensive summary of appointment details, including the customer, employee, service, product, payment, and appointment status. This view is helpful for tracking appointments, managing schedules, and accessing the financial and customer details related to each booking.

Tables Used:

- Appointments | Customers | Employees | Services | Products | Payment_mode

View Name:

- appointment_details_view

Fields Used:

- appointment_id | customer_id | customer_firstname | customer_lastname | employee_id | employee_firstname | employee_lastname | service_name | service_price | product_name | product_price | payment_method | date_time | status

Purpose: The purpose of this view is to consolidate appointment information, linking customer, employee, services, products, and payment details into a single view. This is useful for reports, analysis, or customer service queries regarding individual appointments.

Use Case: Helps employees quickly view upcoming appointments.

```
-- Creating Appointment_Details_View
CREATE VIEW Appointment_Details_View AS
SELECT
    A.appointment_id,
    C.firstname AS customer_firstname,
    C.lastname AS customer_lastname,
    E.firstname AS employee_firstname,
    E.lastname AS employee_lastname,
    A.date_time,
    A.status
FROM Appointments A
JOIN Customers C ON A.customer_id = C.customer_id
JOIN Employees E ON A.employee_id = E.employee_id;

-- Querying Appointment Details

SELECT * FROM Appointment_Details_View
WHERE status = 'Booked';
```

	appointment_id	customer_firstname	customer_lastname	employee_firstname	employee_lastname	date_time	status
1	2	Bob	Smith	Jane	Smith	2025-02-10 09:30:00	Booked
2	5	Eve	Davis	Rick	Moore	2025-02-10 11:00:00	Booked
3	8	Hank	Moore	Emily	Harris	2025-02-10 12:30:00	Booked
4	11	Kara	Anderson	Sam	Clark	2025-02-10 14:00:00	Booked
5	13	Mia	Harris	Kevin	White	2025-02-10 15:00:00	Booked
6	16	Paul	Rodriguez	Rachel	Thomas	2025-02-10 16:30:00	Booked

2. View: Top 5 Lowest Product Stock

Scenario: The Top 5 Lowest Product Stock View identifies products that are running low in stock. This view is essential for inventory management and ensuring that stock levels are consistently monitored, which helps prevent running out of key products needed for appointments.

Tables Used:

- Products

View Name:

- lowest_product_stock_view

Fields Used:

- product_name | quantity

Purpose: The purpose of this view is to display the top 5 products with the lowest stock levels. This enables barbershop management to identify which products need to be restocked soon and helps streamline the inventory management process.

Use Case: Alerts the inventory team to restock essential products before they run out.

```
-- Creating Low_Stock_Products_View

CREATE VIEW Low_Stock_Products_View AS
SELECT
    product_id,
    name AS product_name,
    quantity
FROM Products
ORDER BY quantity ASC
LIMIT 5;


-- Querying Low_Stock_Products View

SELECT * FROM Low_Stock_Products_View;
```

	product_id	product_name	quantity
1	9	Scalp Oil	15
2	13	Hair Growth Tonic	20
3	8	Hair Mousse	20
4	14	Hair Vitamin Capsules	25
5	6	Hair Spray	25

Conclusion

By analyzing the business processes of a barbershop, I was able to identify essential entities and relationships to design an efficient database. My key takeaways include:

- **Understanding Business Needs First** – Before creating tables, identifying real-world requirements is crucial.
- **Primary Keys Ensure Uniqueness** – Every table needs a PK for proper data retrieval.
- **Foreign Keys Maintain Relationships** – FKs help link tables to avoid redundancy.
- **Normalization Helps Prevent Data Duplication** – Structuring data properly ensures efficiency.
- **Choosing the Right Relationship Type** – One-to-Many (1:M) is the most common, while Many-to-Many (M:N) needs junction tables for proper implementation.
- **Using Views for Better Performance** – Creating views can simplify queries and improve accessibility for users.

This project reinforced my database design, SQL implementation, and data integrity skills. In the future, I plan to expand it by integrating business intelligence dashboards to analyze appointment trends, customer behavior, and sales performance.

Final Thoughts

A well-designed database transforms a **small business operation** into a **data-driven enterprise** by improving efficiency, decision-making, and customer satisfaction.