IKIGAI

BARBERSHOP

By:Himani Dilipbhai Makwana

Table of Contents

INTRODUCTION:

Introduction of the Company

IKIGAI Barbershop is a customer-centric business that focuses on providing premium grooming services. The company offers a range of services, including haircuts, beard styling, and skincare treatments.

Mission

IKIGAI's mission is to design and implement a robust, centralized database system tailored for the barbershop industry. The goal is to streamline operations, ensuring seamless integration of customer management, appointment scheduling, inventory tracking, and financial operations.

Purpose

The primary purpose of this database system is to enhance the efficiency of IKIGAI Barbershop by centralizing data, reducing redundancy, and ensuring data integrity. This system will allow managers and employees to access critical business information in real-time.

Objective

The key objectives of the database system include:

- **Optimizing appointment scheduling** to prevent overlaps and ensure efficient staff allocation.
- **Enhancing inventory management** by tracking product stock levels and supplier details.
- **Automating revenue and expense tracking** for improved financial management.

Identify Entities

Customers

Personal details

Appointments

Feedback

Employees

Skills

Schedule

Performance

Services

Details

Prices

Duration

Appointments

Date, Time

Customer

Employee

Products

Details

Inventory

Payments

Transactions

Receipts

PRELIMINARY LIST OF TABLES:

- BRANCH
- Payments
- Customers
- Employees
- Customer Promotions
- Appointment Products
- Appointments
- Services
- Products
- Suppliers
- Promotions

FINAL LIST OF TABLES:

- BRANCH
- Payments
- Customers
- Employees
- Appointments
- Services
- Products
- Suppliers
- Promotions

Data Dictionary:

Table: Customers

Field_name	Attribute	Description
customer_id	Int, Auto-Increment, Primary Key	Unique ID for each customer
name	varchar(100)	Customer's name
phone	varchar(12)	Contact number
email	varchar(100)	Email address
dicount_eligibilty	int	Customer eligibility to get offer
preferences	text	Customer's all time preferences

Table: Branches

Field_name	Attribute	Description
branch_id	Int, Auto-Increment, Primary Key	Unique ID for each branch
name	varchar(100)	Branch name
location	varchar(100)	Branch location

Table: Employees

Field_name	Attribute	Description
employee_id	Int, Auto-Increment, Primary Key	Unique ID for each employee
name	varchar(100)	Employee's name
role	varchar(50)	Job role (e.g., Stylist, Receptionist)
skills	text	Skills of the employee
availability	text	Work schedule
hire_date	date	Employee's hiring date
resign_date	date	Employee's resignation date
branch_id	Int, Foreign key	reference to the branch they work in

Table: Suppliers

Field_name	Attribute	Description
supplier_id	Int, Auto-Increment, Primary Key	Unique ID for each supplier
name	varchar(100)	Supplier's name
contact_info	text	Supplier's contact information

Table: Products

Field_name	Attribute	Description
product_id	Int, Auto-Increment, Primary Key	Unique ID for each product
name	varchar(100)	Product name
quantity	int	Quantity in stock
price	decimal(10,2)	Price of the product
supplier_id	int, Foreign key	Reference to the supplier

Table: Services

Field_name	Attribute	Description
service_id	Int, Auto-Increment, Primary Key	Unique ID for each service
name	varchar(100)	Name of the service
price	decimal(10,2)	Price of the service
duration	int	Duration of the service in minutes

Table: Promotions

Field_name	Attribute	Description
promotion_id	Int, Auto-Increment, Primary Key	Unique ID for each promotion
name	varchar(100)	Promotion name
discount_percentage	decimal(5,2)	Discount percentage
valid_from	date	Promotion start date
valid_to	date	Promotion end date

Table: Appointments

Field_name	Attribute	Description
appointment_id	Int, Auto-Increment, Primary Key	Unique ID for each appointment
customer_id	int, Foreign key	Reference to the customer
employee_id	int, Foreign key	Reference to the employee
product_id	int , Primary Key	Reference to the product
service_id	int , Primary Key	Reference to the service
date_time	Datetime	Appointment date and time
status	varchar(50)	Status of the appointment (e.g., Completed, Scheduled)
preferences	Text	Customer's preferences
offer	int, Foreign key	Customer's discount , promotion_id from promotion table
amount	decimal(10,2)	Payment amount

Table: Payments

Field_name	Attribute	Description
payment_id	Int, Auto-Increment, Primary Key	Unique ID for each payment
appointment_id	int, Foreign key	Reference to the appointment
payment_method	varchar(50)	Method of payment (e.g., Cash, Credit Card)
date	datetime	Payment date

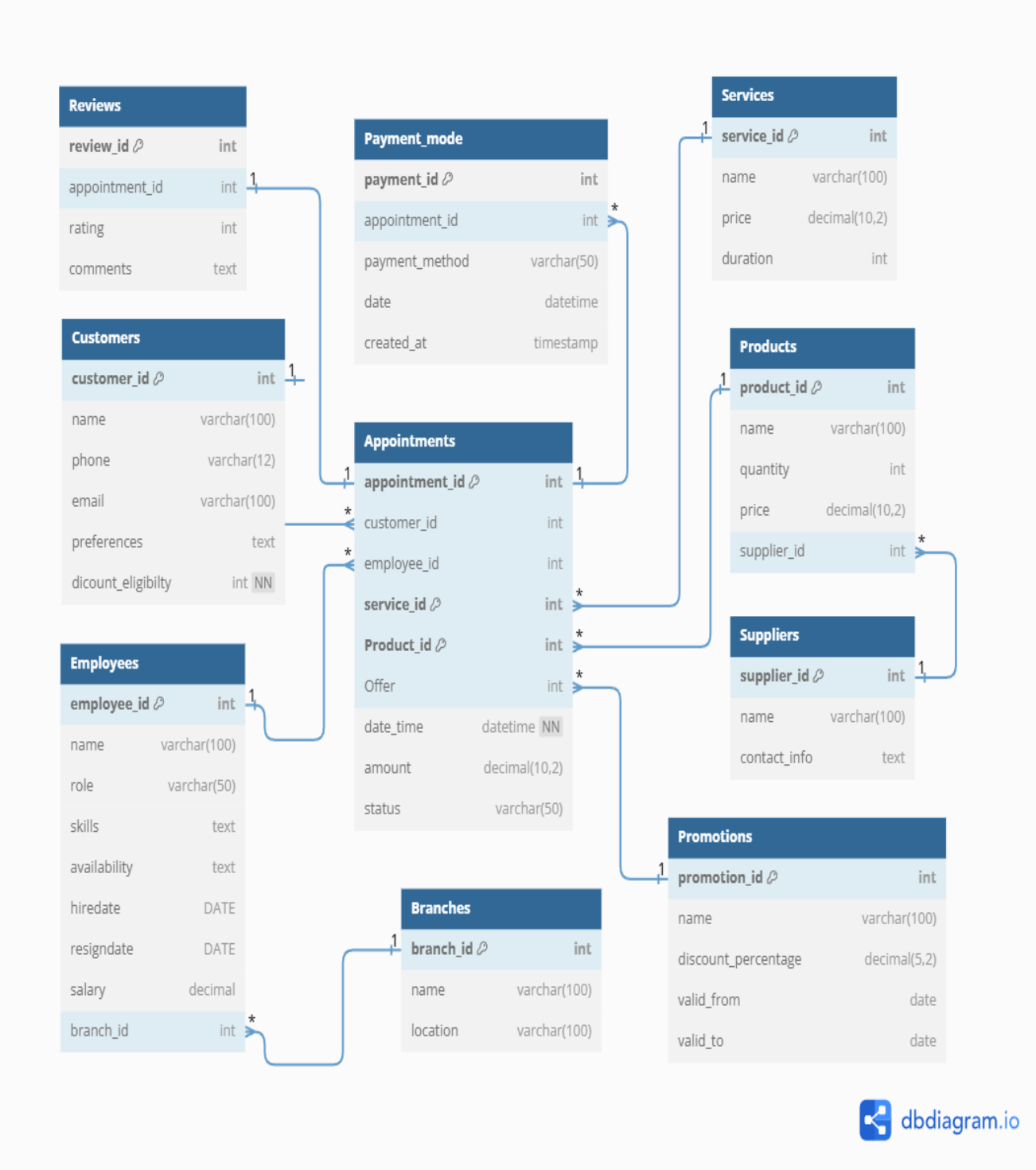
Table: Reviews

Field_name	Attribute	Description
review_id	Int, Auto-Increment, Primary Key	Unique ID for each review
appointment_id	int, Foreign key	Reference to the appointment
rating	int	Customer rating (e.g., 1-5)
comments	text	review comments

Relationship between Table:

Table Name	Related Table	Relationship Type	Foreign Key	Description
Customers	Appointments	One-to-Many	customer_id → Appointments.customer_id	A customer can have multiple appointments.
Employees	Appointments	One-to-Many	employee_id → Appointments.employee_id	An employee can be assigned multiple appointments.
Branches	Employees	One-to-Many	branch_id → Employees.branch_id	A branch can have multiple employees.
Services	Appointments	One-to-Many	service_id → Appointments.service_id	An appointment is associated with one service.
Products	Appointments	Many-to-Many	product_id → Appointments.product_id	A product may be used in multiple appointments.
Promotions	Appointments	One-to-Many	promotion_id → Appointments.offer	A promotion can be applied to multiple appointments.
Products	Suppliers	Many-to-One	supplier_id → Suppliers.supplier_id	A product is supplied by a single supplier.
Appointments	Payment_mode	One-to-Many	appointment_id → Payment_mode.appointment_id	An appointment has many corresponding payment.
Appointments	Reviews	One-to-One	appointment_id → Reviews.appointment_id	Each appointment may have one review.

Entity Relationship Diagram:



https://github.com/8111-himani/Data_Science_Sait_Course

Database Designing:

```sql
-- Create and use the HairSalon database name Chatters
create DATABASE IKIGAI;
use IKIGAI;

-- Table to store customer details

CREATE TABLE Customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each customer
    name VARCHAR(100),                          -- Customer's name
    phone VARCHAR(12),                          -- Contact number
    email VARCHAR(100),                         -- Email address
    dicount_eligibilty int NOT NULL,         --Customer eligibilty
    preferences TEXT                            -- Customer's preferences
);
-- Table to store branch details
CREATE TABLE Branches (
    branch_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each branch
    name VARCHAR(100),                         -- Branch name
    location VARCHAR(100)                      -- Branch location
);


-- Table to store employee details
CREATE TABLE Employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each employee
    name VARCHAR(100),                          -- Employee's name
    role VARCHAR(50),                           -- Job role (e.g., Stylist, Receptionist)
    skills TEXT,                                -- Skills of the employee
    availability TEXT,                          -- Work schedule
    hire_date date,                              -- Employee's hiring date
    resign_date date,                              -- Employee's resignation date
    branch_id INT  ,                            -- Reference to the branch they work in
     FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);

-- Table to store supplier details
CREATE TABLE Suppliers (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each supplier
    name VARCHAR(100),                          -- Supplier's name
    contact_info TEXT                           -- Supplier's contact information
);

```

```sql
-- Table to store product inventory
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each product
    name VARCHAR(100),                         -- Product name
    quantity INT,                              -- Quantity in stock
    price DECIMAL(10, 2),                      -- Price of the product
    supplier_id INT ,                          -- Reference to the supplier
    FOREIGN key (supplier_id) REFERENCES Suppliers(supplier_id)
);



-- Table to store available services
CREATE TABLE Services (
    service_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each service
    name VARCHAR(100),                         -- Name of the service
    price DECIMAL(10, 2),                      -- Price of the service
    duration INT                               -- Duration of the service in minutes
);

-- Table to store promotional offers
CREATE TABLE Promotions (
    promotion_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each promotion
    name VARCHAR(100),                           -- Promotion name
    discount_percentage DECIMAL(5, 2),           -- Discount percentage
    valid_from DATE,                             -- Promotion start date
    valid_to DATE                                -- Promotion end date
);


-- Table to store appointment details
CREATE TABLE Appointments (
    appointment_id INT AUTO_INCREMENT  NOT NULL UNIQUE, -- Unique ID for each appointment
    customer_id INT,                            -- Reference to the customer
    employee_id INT,                            -- Reference to the employee
    product_id INT,                             -- Reference to the product
    service_id INT,                             -- Reference to the service
    date_time DATETIME,                         -- Appointment date and time
    status VARCHAR(50),                         -- Status of the appointment (e.g., Completed, Scheduled)
    preferences TEXT,                         -- Customer's preferences
    offer int,                        --Customer's discount
    amount DECIMAL(10, 2),                    -- Payment amount
    FOREIGN KEY (offer) REFERENCES Promotions(promotion_id),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id),
    FOREIGN KEY (service_id) REFERENCES Services(service_id),
    PRIMARY KEY (appointment_id,product_id,service_id)
);


-- Table to store payment details
CREATE TABLE Payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each payment
    appointment_id INT,                        -- Reference to the appointment

    payment_method VARCHAR(50),                -- Method of payment (e.g., Cash, Credit Card)
    date DATETIME,                             -- Payment date
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id)
);

-- Table to store customer reviews
CREATE TABLE Reviews (
    review_id INT AUTO_INCREMENT PRIMARY KEY, -- Unique ID for each review
    appointment_id INT,                       -- Reference to the appointment
    rating INT,                               -- Customer rating (e.g., 1-5)
    comments TEXT,                            -- Review comments
    FOREIGN KEY (appointment_id) REFERENCES Appointments(appointment_id)
);
```

Inserting sample Data:

```sql
-- Inserting data into the Customers table
INSERT INTO Customers (name, phone, email, dicount_eligibilty, preferences)
VALUES
('John Doe', '1234567890', 'john.doe@example.com', 1, 'Prefers hair coloring'),
('Jane Smith', '9876543210', 'jane.smith@example.com', 2, 'Interested in organic products');

-- Inserting data into the Branches table
INSERT INTO Branches (name, location)
VALUES
('Downtown Salon', '123 Main St, Downtown'),
('Uptown Hair Studio', '456 Uptown Ave, Uptown');

-- Inserting data into the Branches table
INSERT INTO Branches (name, location)
VALUES
('Downtown Salon', '123 Main St, Downtown City'),
('Uptown Beauty', '456 Uptown Ave, Uptown City');

-- Inserting data into the Employees table
INSERT INTO Employees (name, role, skills, availability, hire_date, resign_date, branch_id)
VALUES
('Alice Johnson', 'Stylist', 'Hair Cutting, Coloring, Styling', 'Mon-Fri 9AM-5PM', '2021-05-15', NULL, 1),
('Bob Smith', 'Receptionist', 'Customer Service, Scheduling', 'Mon-Fri 8AM-4PM', '2020-03-20', NULL, 2);

-- Inserting data  into the Suppliers table
INSERT INTO Suppliers (name, contact_info)
VALUES
('Beauty Supplies Co.', '123 Beauty St., Beauty City, BC, 12345. Phone: (555) 123-4567'),
('Haircare Solutions', '456 Hair Ave., Hairtown, HT, 67890. Phone: (555) 234-5678');

-- Inserting  data into the Products table
INSERT INTO Products (name, quantity, price, supplier_id)
VALUES
('Shampoo', 50, 10.99, 1),
('Conditioner', 60, 12.99, 2);
```

```sql
-- Inserting data into the Services table
INSERT INTO Services (name, price, duration)
VALUES
('Haircut', 20.00, 30),
('Shave', 15.00, 20);

-- Inserting data into the Promotions table
INSERT INTO Promotions (name, discount_percentage, valid_from, valid_to)
VALUES
('New Year Special', 20.00, '2025-01-01', '2025-01-31'),
('Winter Sale', 15.00, '2025-01-15', '2025-02-15');

-- Inserting data into the Appointments table
INSERT INTO Appointments (appointment_id, customer_id, employee_id, product_id, service_id, date_time, status, preferences, offer,amount)
VALUES
(1, 1, 1, 1, 1, '2025-01-28 10:00:00', 'Scheduled', 'Quiet environment, natural light', 1,100.00),
(2, 2, 2, 2, 2, '2025-01-28 11:00:00', 'Completed', 'Early morning slots preferred', 2,250.60);

-- Inserting data into the Payments table
INSERT INTO Payments (appointment_id,  payment_method, date)
VALUES
(1,  'Credit Card', '2025-01-10 14:30:00'),
(2,  'Cash', '2025-01-12 15:00:00'),
(3,  'Credit Card', '2025-01-15 09:30:00');

-- Inserting 20 rows into the Reviews table
INSERT INTO Reviews ( appointment_id, rating, comments)
VALUES
(1, 5, 'Great service! I had a wonderful experience.'),
(2, 4, 'Good haircut, but the wait time was a bit long.'),
(3, 5, 'Fantastic styling! Will definitely come back.');
```

Views:

VIEW 1 - APPOINTMENTS

Scenario

In a business environment that offers multiple services to its customers, understanding how appointments are distributed across services is critical for optimizing resource allocation and improving customer satisfaction. For this purpose, a view called **Appointment Summary** is created to summarize appointment details across services.

Purpose

The purpose of this view is to **analyze and count the number of appointments** scheduled for each service. This information is essential for identifying popular services, tracking resource utilization, and making data-driven decisions.

Key Details

Tables Used

The following tables are utilized to gather the necessary information:

1. **Appointments**: Contains the core appointment details like appointment ID, customer ID, service ID, employee ID, date, and status.

2. **Services**: Lists the services provided by the business, including their name and price.
3. **Customers**: Provides customer-specific details like name, phone, and preferences.
4. **Employees**: Stores information about employees such as name, role, and availability.

View Name

The name of the view is **Appointment Summary**. It consolidates data from multiple tables to provide a detailed overview of appointments.

Fields Used

The following fields are included in the view:

- **Appointment_id**: A unique identifier for each appointment.
- **Customer_name**: The name of the customer who booked the appointment.
- **Employee_name**: The name of the employee assigned to the service.
- **Service_name**: The name of the service provided during the appointment.
- **Appointment_date**: The date and time of the appointment.

- **Preferences**: Specific customer preferences related to the appointment.
- **Appointment_status**: The current status of the appointment (e.g., Scheduled, Completed, Cancelled).

Implementation

Creating the View

The SQL statement below demonstrates how the **Appointment Summary** view is created by joining relevant tables:

```sql
-- View for appointment details including customer, employee, and service
CREATE VIEW AppointmentDetails AS
SELECT
    a.appointment_id,
    c.name AS customer_name,
    e.name AS employee_name,
    s.name AS service_name,
    a.date_time AS appointment_date,
    a.preferences as preferences,
    a.status AS appointment_status
FROM
    Appointments a
LEFT JOIN
    Customers c ON a.customer_id = c.customer_id
LEFT JOIN
    Employees e ON a.employee_id = e.employee_id
LEFT JOIN
    Services s ON a.service_id = s.service_id;
```

Example Query

The following query retrieves information for a specific customer, filtering by appointment date:

```sql
-- Query to show appointment details for a specific customer
SELECT *
FROM AppointmentDetails
WHERE customer_name = 'John Doe' and  DATE(appointment_date) = "2025-01-28";
```

Output:

	appointment_id	customer_name	employee_name	service_name	appointment_date	preferences	appointment_status
1	1	John Doe	Alice Johnson	Haircut	2025-01-28 10:00:00	Quiet environment, natural light	Scheduled

VIEW 2 – PRODUCT INVETORY

Scenario

Efficient inventory management is crucial for businesses to monitor stock levels and ensure uninterrupted operations. To facilitate this, a view called **Product Inventory** is created, combining product and supplier data for better stock and supplier management.

Purpose

The purpose of this view is to gather inventory details along with supplier information. It enables businesses to:

- Track stock levels for better inventory management.
- Monitor supplier performance and maintain supplier relationships.

Key Details

Tables Used

The following tables are used in this view:

1. **Products**: Contains product-related details such as product ID, name, quantity, price, and supplier ID.
2. **Suppliers**: Provides information about suppliers, including their name and contact information.

View Name

The name of the view is **ProductInventory**, which consolidates data from the **Products** and **Suppliers** tables to present a unified view of product inventory.

Fields Used

The fields included in the view are:

- **product_id**: Unique identifier for each product.
- **product_name**: The name of the product.
- **product_quantity**: The quantity of the product in stock.
- **product_price**: The price of the product.
- **supplier_name**: The name of the supplier providing the product.
- **supplier_contact**: Contact information for the supplier.

Implementation

Creating the View

The SQL statement below demonstrates how the **ProductInventory** view is created:

```sql
-- View for product inventory along with supplier information
CREATE VIEW ProductInventory AS
SELECT
    p.product_id,
    p.name AS product_name,
    p.quantity,
    p.price,
    s.name AS supplier_name,
    s.contact_info AS supplier_contact
FROM
    Products p
JOIN
    Suppliers s ON p.supplier_id = s.supplier_id;
```

Finding the Most Popular Product

The following query identifies the most popular product based on usage:

```sql
-- Query to show payment details for a specific customer
SELECT *
FROM PaymentDetails
WHERE customer_name = 'John Doe' and  DATE(appointment_date) = "2025-01-28";
```

Output:

	payment_id	customer_name	amount	payment_method	payment_date	promotion_name	appointment_date	appointment_status
1	1	John Doe	100.00	Credit Card	2025-01-10 14:30:00	New Year Special	2025-01-28 10:00:00	Scheduled

VIEW 3 – PAYMENT DETAILS

Scenario

Want to track payment details along with associated promotions and appointment information to analyze revenue and discounts used.

Purpose

To analyze payments, applied promotions, and appointment details for financial tracking and customer discount usage.

Key Details

Tables Used

The following tables are used in this view:

1. **Payments** : Contains payment-related details such as paymentID, , payment_method, date and appointment_id .
2. **Appointments** : Contains the core appointment details like appointment ID, customer ID, service ID, employee ID, date, and status.
3. **Promotions** : Contains promotion-related details such as promotionID, , name,discount_percentage, valid_from and valid_to .
4. **Customers :** Provides customer-specific details like name, phone, and preferences.

View Name

- The name of the view is **PaymentDetails**, which consolidates data from the **Payments, Appointments, Promotions** and **Customers** tables to present a unified view of payment.

Fields Used

The fields included in the view are:

- **Paymentt_id**: Unique identifier for each payment.
- **Payment_method**: The name of the payment mode .
- **date**: The date of the payment.
- **name**: The name of the Customer.
- **Promotion_name**: name of the offer.
- **Date_time**: appointment date and time.
- **amount**: price of appointment after all deduction of offer.
- **Status**: Appointment Status.

Implementation

Creating the View

The SQL statement below demonstrates how the **ProductInventory** view is created:

```sql
-- View for payment details with associated
-- promotions and appointment information
CREATE VIEW PaymentDetails AS
SELECT
    p.payment_id,
    c.name as customer_name,
    p.amount,
    p.payment_method,
    p.date AS payment_date,
    pr.name AS promotion_name,
    a.date_time AS appointment_date,
    a.status AS appointment_status
FROM
    Payments p
LEFT JOIN
    Appointments a ON p.appointment_id = a.appointment_id
LEFT JOIN
    Promotions pr ON a.offer = pr.promotion_id
RIGHT  JOIN Customers c ON a.customer_id = c.customer_id;
```

Printing receipt of specific customer

The following query show payment detail of specific customer:

```sql
-- Query to show payment details for a specific customer
SELECT *
FROM PaymentDetails
WHERE customer_name = 'John Doe' and  DATE(appointment_date) = "2025-01-28";
```

Output:

	payment_id	customer_name	amount	payment_method	payment_date	promotion_name	appointment_date	appointment_status
1	1	John Doe	100.00	Credit Card	2025-01-10 14:30:00	New Year Special	2025-01-28 10:00:00	Scheduled

Conclusion

In today’s dynamic and data-driven world, effectively managing information across interconnected systems is crucial for success. The use of well-designed database views, such as the ones highlighted in this article, showcases how organizations can derive actionable insights from their data while maintaining clarity and simplicity. Whether it's tracking product inventories, analyzing payment details, or monitoring key performance indicators, these views provide a structured and streamlined approach to handling complex datasets.

The **Product Inventory View** enables businesses to maintain optimal stock levels while fostering strong supplier relationships, ensuring operational efficiency. By combining product and supplier data, this view allows for proactive decision-making, such as identifying inventory shortages or selecting reliable suppliers. The capability to query the most popular products further empowers stakeholders to focus on high-demand items, boosting profitability and customer satisfaction.

Similarly, the **Payment Details View** demonstrates the importance of integrating payments, promotions, and customer interactions to evaluate financial performance and customer engagement. This view provides insights into revenue trends, promotional effectiveness, and customer preferences, helping businesses tailor their strategies for maximum impact. The ability to filter data by customer and appointment dates makes it a valuable tool for personalized marketing and targeted customer service initiatives.

In a broader sense, these examples underline the power of relational databases and SQL views in unlocking the potential of stored data. By

creating logical representations of data that simplify its complexity, SQL views help organizations focus on insights rather than the intricacies of raw data. This abstraction not only improves decision-making but also ensures data security by restricting access to sensitive information through carefully defined views.

From a technical perspective, the SQL scripts showcased in this article highlight the simplicity and elegance of creating views using CREATE VIEW statements and leveraging JOIN clauses. These examples are scalable and can be customized to suit various business needs, demonstrating the flexibility of SQL as a tool for data management. The inclusion of filtering options, such as WHERE clauses, ensures that data retrieval is precise and efficient.

However, it's important to note that the implementation of views is not without challenges. Businesses must carefully assess the underlying database schema and ensure that it is optimized for performance. Poorly designed views or excessive use of joins can lead to inefficiencies, slowing down data retrieval processes. Hence, a balanced approach that prioritizes both performance and functionality is essential.

Looking ahead, the role of data analytics in driving innovation and competitiveness will continue to grow. Organizations that embrace database management best practices, such as the use of views, will be better equipped to adapt to evolving market demands and leverage their data as a strategic asset. Additionally, the integration of modern technologies like artificial intelligence and machine learning into database systems presents exciting opportunities for predictive analytics and automated decision-making.

In conclusion, SQL views are a testament to the transformative power of structured data management. They empower businesses to derive

meaningful insights from complex datasets, enhance operational efficiency, and make informed decisions. The examples of the Product Inventory and Payment Details views serve as a starting point for exploring the vast possibilities of SQL in solving real-world problems. By investing in robust database practices and fostering a culture of data-driven decision-making, organizations can position themselves for long-term success in an increasingly competitive landscape.

As technology continues to evolve, the importance of data as a cornerstone of modern business strategy cannot be overstated. Organizations must remain committed to refining their data management practices, ensuring that they are well-equipped to harness the full potential of their most valuable asset—information.

By mastering the art of data management through tools like SQL views, businesses can not only meet the challenges of today but also unlock new opportunities for growth and innovation in the future. Let this serve as a reminder that the journey toward effective data utilization begins with a single query—one that transforms data into actionable insights and possibilities.