A Database Design for Salon De Maria Appointment System

A Project Study Presented to the

School of Architecture, Computing and Engineering National University

Philippines

Lipa City

In Partial Fulfillment of the Requirement for the Course: CTINFFMGL:

Information Management

By:

Jei Q. Pastrana

March 11, 2024

Salon De Maria Appointment System

1. Clients
   - Each Client has a unique ClientID and provides essential details including the client's first name, last name, email, and phone number.

2. Appointment
   - Book your Appointment! Appointments are made by clients and represented with a unique AppointmentID. They are associated with their selected service (ServiceID), the staff (StaffID), and, if applicable, a discount (DiscountID), while also monitoring their a ppointment status.
3. Staff
   - Meet our team! Staffs are integral to the system, each identified by a StaffID and providing details like first name, last name, specialty, email, phone, and their availability.

4. Payment
   - Secure transactions! Payments are linked to appointments, detailing PaymentID, associated AppointmentID, the amount paid (PaymentAmount), the date of payment (PaymentDate), and the payment method used (PaymentMehod).
5. Discount
   - Each discount is exclusive offers with a unique DiscountID, capturing the associated ServiceID and StaffID if the given discount was staff-initiated. Attributes such as discount_code, amount, and the validity period (ValidFrom to ValidTo) are also recorded.
6. Service
   - The numerous services offered are represented with a unique ServiceID. It includes attributes such as ServiceName, ServiceDescription, ServicePrice, and its associated service category (SC_ID).

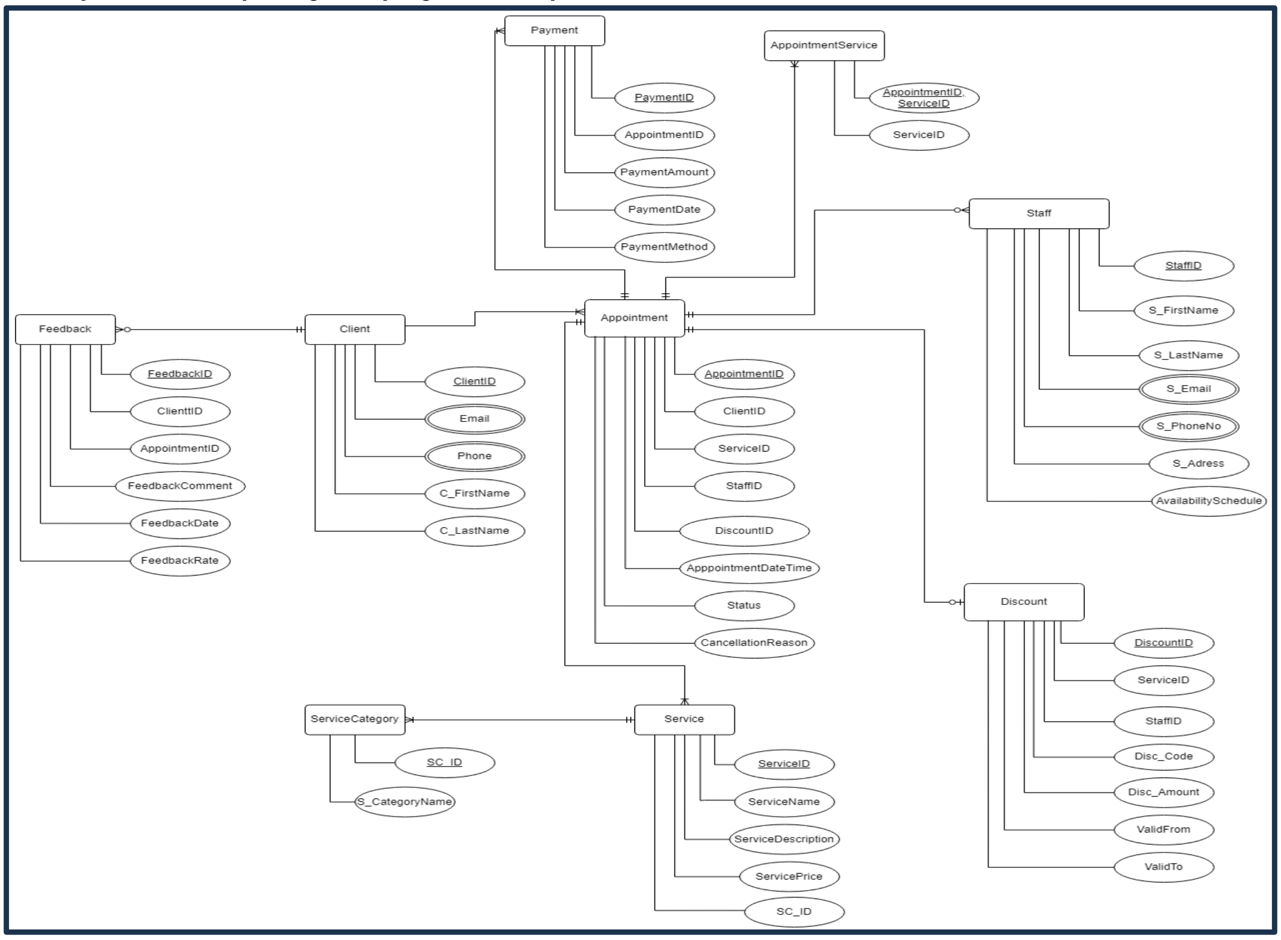
7. Service Category
   - Different services are categorized into various groups, each identified by SC_ID and labeled with a name.

8. Feedback
   - Share your thoughts! Clients can provide feedback, each identified by a FeedbackID, and connect to ClientID and AppointmentID. Including a Rating, Comment, and the date of the feedback.
9. AppointmentService
   - Additional Services? Clients may avail more service during their appointment by connecting AppointmentService to both Appointment (AppointmentID), Service (ServiceID), and tracks the extra services availed within a single appointment.

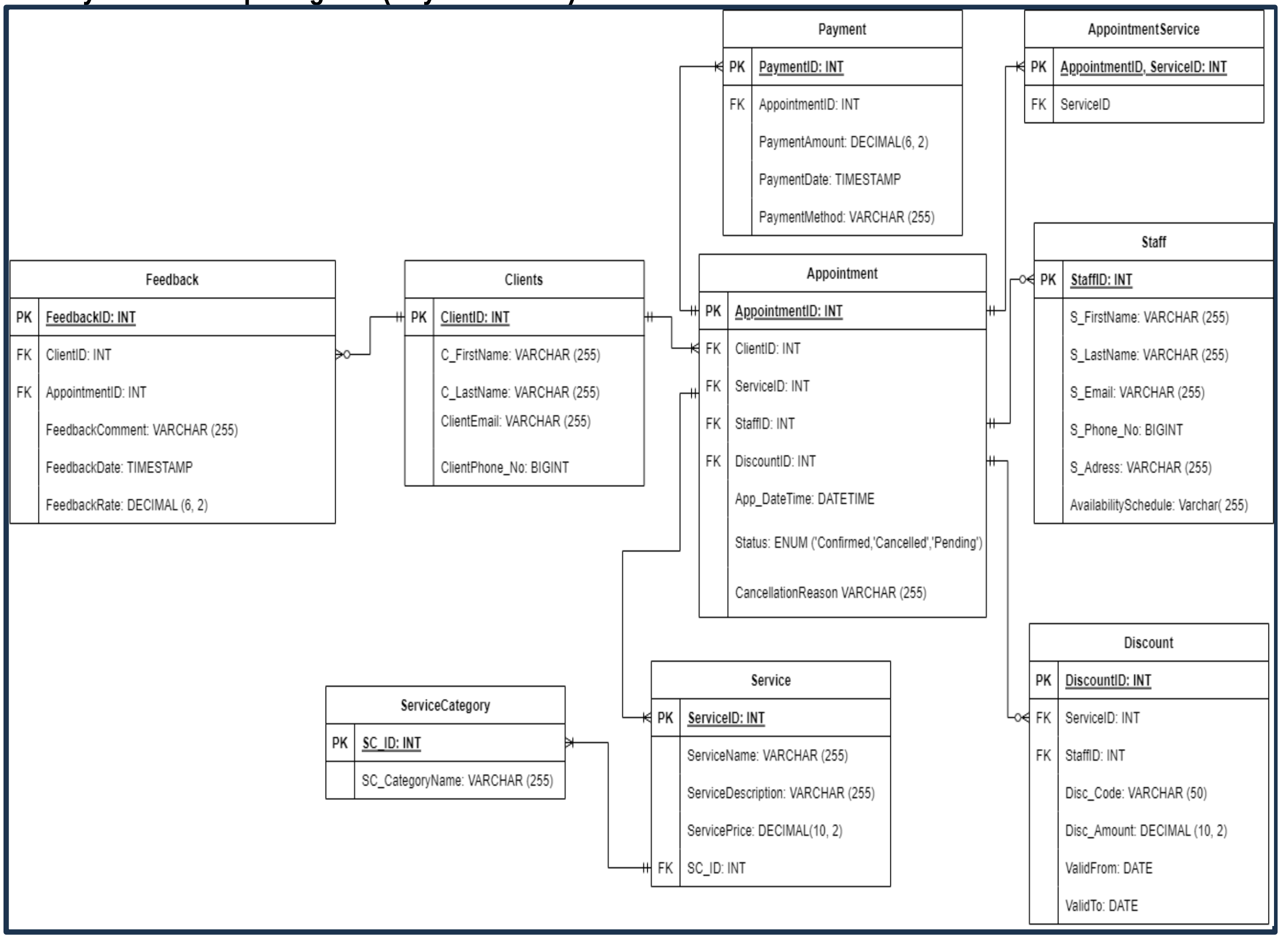
Requirements

1. Clients can make appointments, and each appointment is associated with a specific client.
2. Each appointment shall take account of the specific services, and each service can have different types and rates.
3. Service Category is the organization of services offered and should be linked to Service, and each service is associated with their respective service category. ￼
4. Payments should link to appointments, having the PaymentDate, Amount, and PaymentMethod.
5. Discounts are associated with appointments, capturing the discount amount, validity period, and, if applicable, each discount can be staff-initiated.
6. Staff are associated with handling the appointments, and each appointment is managed by a specific staff.
7. Feedback can be provided by clients after their appointments, receiving ratings and comments, and the feedback date.

Entity-Relationship Diagram (Logical Level)



Entity-Relationship Diagram (Physical Level)



These relationships help define how data in one table is related to data in another table, allowing to retrieve and manipulate information across tables in a meaningful way.

1. **Staff and Appointment**
    - The **Appointment** table has a foreign key **StaffID** that references the primary key **StaffID** in the **Staff** table. This relationship indicates that a single staff member can be linked to zero or many appointments, implying that not all staff members may have appointments. However, when an appointment is made, it is associated with exactly one staff member.
2. **Client and Appointment**
    - The **Appointment** table has a foreign key **ClientID** that references the primary key **ClientID** in the **Clients** table, establishing a one-to-many relationship where a single client can book multiple appointments over time, but each individual appointment is linked to one specific client.
3. **ServiceCategory and Service**
    - The **Service** table has a foreign key **SC_ID** that references the primary key **SC_ID** in the **ServiceCategory**. This establishes a relationship where each service is associated with a single service category, allowing multiple services to belong to the same category.
4. **Client and Feedback**
    - The **Feedback** table has a foreign key **AppointmentID** that references the primary key **AppointmentID** in the **Clients** table. This establishes where a client can provide zero or many feedback entries, but each feedback entry is associated with a single client.
5. **Appointment and Feedback**
    - The **Feedback** table has a foreign key **AppointmentID** that references the primary key **AppointmentID** in the **Appointment** table. This establishes a relationship where an appointment can have zero or many feedback entries associated with it, while each feedback entry is tied to a single appointment.
6. **Discount and Appointment**
    - The **Appointment** table has a foreign key **DiscountID** that references the primary key **DiscountID** in the **Discount** table. This establishes a relationship where an appointment can have a discount associated with it, but not necessarily for every appointment. Each appointment can have zero or one discount linked to it, while each discount is tied to a single appointment.
7. **Staff and Discount**
    - The **Discount** table has a foreign key **StaffID** that references the primary key **StaffID** in the **Staff** table. This establishes a relationship where a staff member can be associated with multiple discounts, while each discount is linked to a single staff member. This relationship facilitates tracking of staff-initiated promotions or discounts, if applicable.
8. **Payment and Appointment**
    - The **Payment** table has a foreign key **AppointmentID** that references the primary key **Appointment ID** in the **Appointment** table, establishing a relationship where each payment is tied to a specific appointment. However, each appointment can have multiple payments at once, allowing a client to cover the cost of all services availed during her appointment in one collective payment if needed.
9. **AppointmentService, Appointment, and Service Tables**
    - The **AppointmentService** , **Appointment,** and **Service** Tables are connected through foreign key relationships.
    - **AppointmentService.AppointmentID** references **Appointment.AppointmentID.**
    - **AppointmentService.ServiceID** references **Service.ServiceID.**
    - This establishes a relationship where an appointment can have multiple services tied to it, and each service can be associated with multiple appointments. This way, we can keep track of the additional services availed in a single appointment if needed.

Relation Schema and SQL Implementation of the Design

1. **Relation:** Clients

   Attributes:

   - ClientID (INT, Primary Key)
   - C_FirstName (VarChar 255)
   - C_LastName (VarChar 255)
   - ClientEmail (VarChar 255)
   - ClientPhone_No (VarChar 255)

   Constraints:

   - Primary Key: ClientID
   - Unique: ClientEmail
   - Unique: ClientPhone_No

   SQL Query

```
CREATE TABLE Clients(
    ClientID INT PRIMARY KEY AUTO_INCREMENT,
    C_FirstName VARCHAR(255) NOT NULL,
    C_LastName VARCHAR(255) NOT NULL,
    ClientEmail VARCHAR(255) UNIQUE NOT NULL,
    ClientPhone_No BIGINT UNIQUE NOT NULL
);
```

2. **Relation:** ServiceCategory

   Attributes:

   - SC_ID (INT, Primary Key)
   - SC_CategoryName (VarChar 255)

   Constraints:

   - Primary Key: SC_ID

   SQL Query

```
CREATE TABLE ServiceCategory(
    SC_ID INT PRIMARY KEY AUTO_INCREMENT,
    SC_CategoryName VARCHAR(255)
);
```

3. **Relation:** Service

   Attributes:

   - ServiceID (INT, Primary Key)
   - ServiceName (VarChar 255)
   - ServiceDescription (VarChar 255)
   - ServicePrice (VarChar 255)
   - SC_ID (INT)

   Constraints:

   - Primary Key: ServiceID
   - Foreign Key: SC_ID references ServiceCategory (SC_ID)

SQL Query

```
CREATE TABLE Service(
    ServiceID INT PRIMARY KEY AUTO_INCREMENT,
    ServiceName VARCHAR(255) NOT NULL,
    ServiceDescription VARCHAR(255) NOT NULL,
    ServicePrice DECIMAL(10, 2) NOT NULL,
    SC_ID INT,
    FOREIGN KEY(SC_ID) REFERENCES ServiceCategory(SC_ID) ON UPDATE
RESTRICT ON DELETE RESTRICT
);
```

4. **Relation:** Staff
   Attributes:
   - StaffID (INT, Primary Key)
   - S_FirstName (VarChar 255)
   - S_LastName (VarChar 255)
   - S_Email (VarChar 255)
   - S_Phone (BIGINT)
   - AvailabilitySchedule (VarChar 255)

   Constraints:

   - Primary Key: StaffID

SQL Query

```
CREATE TABLE Staff(
    StaffID INT PRIMARY KEY AUTO_INCREMENT,
    S_FirstName VARCHAR(255) NOT NULL,
    S_LastName VARCHAR(255) NOT NULL,
    S_Email VARCHAR(255) UNIQUE NOT NULL,
    S_Phone BIGINT UNIQUE NOT NULL,
    S_Address VARCHAR(255) NOT NULL,
    AvailabilitySchedule VARCHAR(255) DEFAULT NULL
);
```

5. **Relation:** Discount
   Attributes:
   - DiscountID(INT, Primary Key)
   - ServiceID (INT)
   - StaffID (INT)
   - Disc_Code (VarChar 255)
   - Disc_Amount (VarChar 255)
   - ValidFrom (DATE)
   - ValidTo (DATE)

   Constraints:

   - Primary Key: DiscountID
   - Foreign Key: ServiceID references Service(ServiceID)
   - Foreign Key: StaffID references Staff(StaffID)

SQL Query:

```
CREATE TABLE DISCOUNT(
    DiscountID INT PRIMARY KEY AUTO_INCREMENT,
    ServiceID INT NOT NULL,
    StaffID INT DEFAULT NULL,
    Disc_Code VARCHAR(255) DEFAULT NULL,
    Disc_Amount DECIMAL(10, 2) DEFAULT NULL,
    ValidFrom DATE,
    Validto DATE,
    FOREIGN KEY(ServiceID) REFERENCES Service(ServiceID) ON UPDATE
RESTRICT ON DELETE RESTRICT
);
```

6. **Relation:** Appointment
   Attributes:
   - AppointmentID (INT, Primary Key)
   - ClientID (INT)
   - ServiceID (INT)
   - StaffID (INT)
   - DiscountID (INT)
   - App_DateTime (DATETIME)
   - Status (ENUM)
   - CancellationReason (Varchar 255)

   Constraints:

   - Primary Key: ApppointmentID
   - Foreign Key: ClientID references Clients(ClientID)
   - Foreign Key: ServiceID references Service(ServiceID)
   - Foreign Key: StaffID references Staff(StaffID)
   - Foreign Key:DiscountID references Discount(DiscountID)

   SQL Query

```
CREATE TABLE Appointment(
    AppointmentID INT PRIMARY KEY AUTO_INCREMENT,
    ClientID INT NOT NULL,
    ServiceID INT NOT NULL,
    StaffID INT NOT NULL,
    DiscountID INT,
    App_DATETIME DATETIME,
    STATUS ENUM
        ('Confirmed', 'Cancelled', 'Pending'),
        CancellationReason VARCHAR(255) DEFAULT NULL,
        FOREIGN KEY(ClientID) REFERENCES Clients(ClientID) ON UPDATE
RESTRICT ON DELETE RESTRICT,
        FOREIGN KEY(StaffID) REFERENCES Staff(StaffID) ON UPDATE
RESTRICT ON DELETE RESTRICT,
        FOREIGN KEY(ServiceID) REFERENCES Service(ServiceID) ON
UPDATE RESTRICT ON DELETE RESTRICT
);
```

7. **Relation:** Payment
   Attributes:
   - Payment_ID (INT, Primary Key)
   - AppointmentID (INT)
   - Payment_Amount (Decimal 6,2)
   - Payment_Date (Timestamp)
   - Payment_Method (Varchar 255)

Constraints:

- Primary Key: PaymentID
- Foreign Key: AppointmentID references Appointment(AppointmentID)

SQL Query

```
CREATE TABLE Payment(
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    AppointmentID INT NOT NULL,
    PaymentAmount DECIMAL(6, 2) NOT NULL,
    PaymentDate TIMESTAMP,
    PaymentMethod VARCHAR(255) NOT NULL,
    FOREIGN KEY(AppointmentID) REFERENCES Appointment(AppointmentID)
ON UPDATE RESTRICT ON DELETE RESTRICT
);
```

8. **Relation:** Feedback
   Attributes:
   - FeedbackID (INT, Primary Key)
   - ClientID (INT)
   - AppointmentID (INT)
   - FeedbackComment (Varchar 255)
   - FeedbackDate TIMESTAMP
   - FeedbackRate DECIMAL(6, 2)

Constraints:

- Primary Key: FeedBackID
- Foreign Key: ClientID references Clients(ClientID)
- Foreign Key: AppointmentID references Appointment(AppointmentID)

SQL Query

```
CREATE TABLE Feedback(
    FeedbackID INT PRIMARY KEY AUTO_INCREMENT,
    ClientID INT NOT NULL,
    AppointmentID INT NOT NULL,
    FeedbackComment VARCHAR(255) NOT NULL,
    FeedbackDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FeedbackRate DECIMAL(6, 2) NOT NULL,
    FOREIGN KEY(ClientID) REFERENCES Clients(ClientID) ON UPDATE
RESTRICT ON DELETE RESTRICT,
    FOREIGN KEY(AppointmentID) REFERENCES Appointment(AppointmentID)
ON UPDATE RESTRICT ON DELETE RESTRICT
);
```

9. **Relation:** AppointmentService
   Attributes:
   - AppointmentID (INT, Primary Key)
   - ServiceID (INT,Primary Key)

Constraints:

- Primary Key: AppointmentID and ServiceID
- Foreign Key: AppointmentID references Appointment(AppointmentID)
- Foreign Key: ServiceID references Service(ServiceID)

SQL Query:

```
CREATE TABLE AppointmentService(
    AppointmentID INT,
    ServiceID INT,
    PRIMARY KEY(AppointmentID, ServiceID),
    FOREIGN KEY(AppointmentID) REFERENCES Appointment(AppointmentID) ON UPDATE
RESTRICT ON DELETE RESTRICT,
    FOREIGN KEY(ServiceID) REFERENCES Service(ServiceID) ON UPDATE RESTRICT ON DELETE
RESTRICT
);
```

Populating Database with Dummy Data

```
----Populate Appointment Table
INSERT INTO appointment (ClientID, ServiceID, StaffID, DiscountID, App_DATETIME,
STATUS, CancellationReason) VALUES
(1, 1, 1, 1, '2024-02-01 11:00:00', 'Confirmed', NULL),
(2, 2, 7, 2, '2024-02-03 10:00:00', 'Confirmed', NULL),
(3, 3, 3, 3, '2024-02-07 10:00:00', 'Confirmed', NULL),
(4, 4, 4, 4, '2024-02-01 11:00:00', 'Confirmed', NULL),
(5, 5, 5, 5, '2024-02-02 12:00:00', 'Confirmed', NULL),
(6, 6, 6, NULL, '2024-02-06 09:30:35', 'Confirmed', NULL),
(7, 7, 7, NULL, '2024-02-03 12:06:35', 'Confirmed', NULL),
(8, 8, 7, NULL, '2024-02-03 15:00:35', 'Confirmed', NULL),
(9, 9, 7, NULL, '2024-02-04 13:06:35', 'Confirmed', NULL),
(10, 1, 1, 1, '2024-02-01 12:00:00', 'Confirmed', NULL),
(11, 2, 2, 2, '2024-02-06 08:00:00', 'Confirmed', NULL),
(12, 3, 2, 3, '2024-02-06 10:00:00', 'Confirmed', NULL),
(13, 3, 2, 4, '2024-02-06 12:00:00', 'Confirmed', NULL),
(14, 5, 5, 5, '2024-02-02 14:00:00', 'Confirmed', NULL),
(15, 6, 4, NULL, '2024-02-01 13:00:00', 'Confirmed', NULL),
(16, 7, 3, NULL, '2024-02-07 14:00:00', 'Confirmed', NULL),
(17, 8, 7, NULL, '2024-02-04 16:00:00', 'Confirmed', NULL),
(18, 9, 7, NULL, '2024-02-04 17:00:00', 'Confirmed', NULL),
(19, 1, 1, 1, '2024-02-05 13:00:00', 'Confirmed', NULL),
(20, 2, 2, 2, '2024-03-06 08:00:00', 'Pending', NULL),
(21, 3, 3, NULL, '2024-03-06 12:00:00', 'Pending', NULL),
(22, 4, 4, NULL, '2024-03-07 12:00:00', 'Pending', NULL),
(23, 5, 5, NULL, '2024-03-08 12:00:00', 'Pending', NULL),
(24, 6, 6, NULL, '2024-03-06 12:00:00', 'Pending', NULL),
(25, 7, 7, NULL, '2024-03-03 12:00:00', 'Cancelled', 'Day Off'),
(26, 8, 4, NULL, '2024-02-01 11:00:00', 'Cancelled', 'Conflicting Schedule'),
(27, 9, 3, NULL, '2024-03-05 12:00:00', 'Cancelled', 'Day Off'),
(28, 1, 1, NULL, '2024-03-08 12:00:00', 'Cancelled', 'Other Commitments'),
(29, 2, 2, NULL, '2024-03-08 12:00:00', 'Cancelled', 'Health reasons'),
(30, 3, 3, NULL, '2024-03-05 12:00:00', 'Cancelled', 'Day Off');
```

```

--Populate  AppointmentService Table
INSERT INTO AppointmentService (AppointmentID, ServiceID) VALUES
(2, 8),
(2, 9),
(3, 1),
(4, 3),
(4, 8),
(8, 1),
(10, 3),
(11, 8),
(12, 9),
(14, 8);
```

```

--Populate Clients table
INSERT INTO clients (C_FirstName,C_LastName,ClientEmail,ClientPhone_No)VALUES
('Maria', 'Clara', 'MariaClara@gmail.com', '09123456789'),
('Anna', 'Gonzales', 'AnnaGonzales@gmail.com', '09187654321'),
('Karen', 'Santos', 'KarenSantos@gmail.com', '09234567890'),
('Liza', 'Reyes', 'LizaReyes@gmail.com', '09456789012'),
('Catherine', 'Garcia', 'CatherineGarcia@gmail.com', '09321098765'),
('Sarah', 'Lopez', 'SarahLopez@gmail.com', '09876543210'),
('Grace', 'Martinez', 'GraceMartinez@gmail.com', '09765432109'),
('Jennifer', 'Torres', 'JenniferTorres@gmail.com', '09987654321'),
('Christine', 'Rodriguez', 'ChristineRodriguez@gmail.com', '09543210987'),
('Michelle', 'Dela Cruz', 'MichelleDelaCruz@gmail.com', '09109876543'),
('Isabella', 'Santos', 'IsabellaSantos@gmail.com', '09220225678'),
('Sophia', 'Gonzales', 'SophiaGonzales@gmail.com', '09157654321'),
('Angelica', 'Reyes', 'AngelicaReyes@gmail.com', '09264567891'),
('Julia', 'Torres', 'JuliaTorres@gmail.com', '09546789012'),
('Marian', 'Garcia', 'MarianGarcia@gmail.com', '09491098765'),
('Chloe', 'Lopez', 'ChloeLopez@gmail.com', '09321543210'),
('Nicole', 'Martinez', 'NicoleMartinez@gmail.com', '09505432109'),
('Daniella', 'Torres', 'DaniellaTorres@gmail.com', '09907654321'),
('Zoe', 'Rodriguez', 'ZoeRodriguez@gmail.com', '09653210987'),
('Jasmine', 'Dela Cruz', 'JasmineDelaCruz@gmail.com', '09119876543'),
('Ella', 'Gonzalez', 'EllaGonzalez@gmail.com', '09143456789'),
('Sophie', 'Santos', 'SophieSantos@gmail.com', '09157754321'),
('Angel', 'Reyes', 'AngelReyes@gmail.com', '09264567890'),
('Juliana', 'Torres', 'JulianaTorres@gmail.com', '09476789012'),
('Mariel', 'Garcia', 'MarielGarcia@gmail.com', '09381098765'),
('Celine', 'Lopez', 'CelineLopez@gmail.com', '09806543210'),
('Nina', 'Martinez', 'NinaMartinez@gmail.com', '09725432109'),
('Diana', 'Torres', 'DianaTorres@gmail.com', '09917654321'),
('Sofia', 'Rodriguez', 'SofiaRodriguez@gmail.com', '09563210987'),
('Angelica', 'Dela Cruz', 'AngelicaDelaCruz@gmail.com', '09639876543');
```

```
--Populate Discount Table
INSERT INTO Discount (ServiceID, StaffID, Disc_Code, Disc_Amount, ValidFrom, ValidTo)
VALUES
    (1, 1, 'DISCOUNT1', 10.00, '2024-02-01', '2024-03-15'),
    (2, 2, 'DISCOUNT2', 125.00, '2024-02-01', '2024-03-15'),
    (3, 3, 'DISCOUNT3', 14.00, '2024-02-01', '2024-03-15'),
    (4, 4, 'DISCOUNT4', 50.00, '2024-02-01', '2024-03-15'),
    (5, NULL, 'PROMO1', 100.00, '2024-02-01', '2024-03-15');
```

```
--Populate Feedback Table
INSERT INTO feedback (ClientID, AppointmentID, FeedbackComment, FeedbackDate,
FeedbackRate) VALUES
(1, 1, 'I had a great haircut experience at Salon de Maria. The stylist was attentive
to my preferences and gave me the perfect haircut.', '2024-02-01 12:00:00', 4.5),
(2, 2, 'I absolutely loved the hair coloring! It turned out exactly how I wanted
it.', '2024-02-04 12:00:00', 4.5),
(2, 2, 'Manicure was great! Nails look pretty.', '2024-02-04 12:00:00', 4.5),
(2, 2, 'Pedicure was relaxing. Results are fantastic.', '2024-02-04 12:00:00', 4.6),
(3, 3, 'The hair styling was amazing! The stylist really understood what I wanted.',
'2024-02-08 14:00:00', 4.8),
(3, 3, 'Haircut was perfect! Excellent job by the stylist.', '2024-02-08 14:00:00',
4.8),
(4, 4, 'The hair treatments were relaxing and rejuvenating. My hair feels so healthy
now.', '2024-02-02 11:00:00', 4.3),
(4, 4, 'Hair styling was amazing! Stylist really understood.', '2024-02-02 11:00:00',
4.3),
(4, 4, 'Manicure was great! Nails look pretty.', '2024-02-02 11:00:00', 4.3),
(5, 5, 'The Brazilian blowout was a game-changer! My hair has never looked better.',
'2024-02-03 12:00:00', 4.9),
(6, 6, 'The semi rebonding treatment didn''t turn out as expected. My hair still
looks frizzy.', '2024-02-07 09:30:35', 3.2),
```

```sql
(7, 7, 'The full rebonding treatment was disappointing. My hair feels dry and damaged.', '2024-02-08 12:06:35', 2.5),
(8, 8, 'The manicure was great! My nails look so pretty and well-groomed.', '2024-02-04 15:00:35', 4.7),
(9, 9, 'The pedicure was relaxing and the results are fantastic. I feel pampered.', '2024-02-05 13:06:35', 4.6),
(10, 10, 'The haircut was just what I needed! The stylist did an excellent job.', '2024-02-02 12:00:00', 4.8),
(10, 10, 'Hair styling was amazing! Stylist really understood.', '2024-02-02 12:00:00', 4.8),
(11, 11, 'The hair coloring was uneven and patchy. Im disappointed with the results.', '2024-02-07 08:00:00', 2.7),
(11, 11, 'Manicure was great! Nails look pretty.', '2024-02-07 08:00:00', 4.7),
(12, 12, 'The hair styling was okay, but it didn''t last long. My hair fell flat after a few hours.', '2024-02-07 10:00:00', 3.5),
(12, 12, 'Pedicure was relaxing. Results are fantastic.', '2024-02-07 10:00:00', 4.2),
(13, 13, 'The hair treatments left my hair feeling greasy and weighed down.', '2024-02-07 12:00:00', 2.0),
(14, 14, 'The Brazilian blowout didn''t live up to the hype. My hair is still frizzy and unmanageable.', '2024-02-03 14:00:00', 2.8),
(14, 14, 'Manicure was great! Nails look pretty.', '2024-02-03 14:00:00', 4.8),
(15, 15, 'The semi rebonding treatment was a disaster! My hair feels dry and damaged.', '2024-02-02 13:00:00', 1.5);
```

```sql
--Populate AppointmentService Table
INSERT INTO appointmentservice(AppointmentID, ServiceID)
VALUES('2', '8'),('2', '9'),('3', '1'),('8', '1'),('4', '3'),('4', '8'),('10', '3'),('11', '8'),('12', '9'),('14', '8');
```

```sql
--Populate Payment Table
INSERT INTO payment (AppointmentID, PaymentAmount, PaymentDate, PaymentMethod) VALUES
(1, 90.00, '2024-02-01 11:30:00', 'CASH'),
(2, 1450.00, '2024-03-03 11:00:00', 'MOBILE WALLET:GCASH'),
(3, 240.00, '2024-02-07 12:00:00', 'MOBILE WALLET:GCASH'),
(4, 740.00, '2024-02-01 13:00:00', 'MOBILE WALLET:GCASH'),
(5, 1000.00, '2024-02-02 15:00:00', 'MOBILE WALLET:GCASH'),
(6, 800.00, '2024-02-06 11:30:00', 'CASH'),
(7, 1500.00, '2024-02-03 14:00:00', 'CASH'),
(8, 200.00, '2024-02-03 14:30:00', 'CASH'),
(9, 100.00, '2024-02-03 14:30:00', 'CASH'),
(10, 240.00, '2024-02-01 13:00:00', 'CASH'),
(11, 1350.00, '2024-02-06 10:00:00', 'CASH'),
(12, 240.00, '2024-02-06 11:00:00', 'CASH'),
(13, 140.00, '2024-02-06 14:00:00', 'CASH'),
(14, 1100.00, '2024-02-02 15:00:00', 'CASH'),
(15, 800, '2024-02-01 15:00:00', 'CASH'),
(16, 1500, '2024-02-07 16:00:00', ' Credit Card '),
(17, 100, '2024-02-04 16:30:00', ' Credit Card'),
(18, 100, '2024-02-04 17:30:00', ' Credit Card'),
(19, 100, '2024-02-05 13:30:00', 'Credit Card');
```

```sql
--Populate Service Table
INSERT INTO service (ServiceName, ServiceDescription, ServicePrice, SC_ID) VALUES
('Haircut', 'Professional hair cutting service', 100.00, 1),
('Hair coloring', 'Various hair coloring techniques including highlights, balayage, and full color', 1250.00, 1),
('Hair styling', 'Professional hair styling services including blowouts and updos', 140.00, 1),
('Hair treatments', 'Specialized hair treatments for conditioning and nourishing
hair', 500.00, 1),
('Brazilian Blowout', 'Hair treatment for smoothing and conditioning', 1000.00, 1),
('Semi Rebond', 'Semi-permanent hair straightening treatment', 800.00, 1),
('Full Rebond', 'Permanent hair straightening treatment', 1500.00, 1),
('Manicure', 'Nail care treatment for hands', 100.00, 2),
('Pedicure', 'Nail care treatment for feet', 100.00, 2);
```

```
--Populate ServiceCategory Table
INSERT INTO servicecategory (SC_CategoryName) VALUES
('Hair Services'),
('Nail Services');
```

```
--Populate Staff Table
INSERT INTO staff
(S_FirstName,S_LastName,S_Email,S_PhoneNo,S_Address,AvailabilitySchedule) VALUES
('Choi','Dela Cruz','ChoiDelacruz@gmail.com','09789012345','Tanauan City,
Batangas','Monday-Friday, 9:00 AM - 5:00 PM'),
('Maria','Bondos','MariaBondos@yahoo.com','09123456789','Tanauan City,
Batangas','Tuesday-Saturday, 8:00 AM - 8:00 PM'),
('Angel','Parra','AngelParra@gmail.com','09234567890','Tanauan City,
Batangas','Wednesday-Sunday, 10:00 AM - 6:00 PM'),
('Alohamia ',' Arante','Alohamiaarante@gmail.com','09345678901','Tanauan City,
Batangas','Thursday-Monday, 11:00 AM - 7:00 PM'),
('Joy','De Guzman','Joy.deguzman@gmail.com','09456789012','Tanauan City,
Batangas','Friday-Tuesday, 12:00 PM - 8:00 PM'),
('Rowena San Lorenzo','Bazar','RowenaBazar18@gmail.com','09567890123','Tanauan City,
Batangas','Tuesday-Sunday, 9:30 AM - 5:45 PM'),
('Mika','Santos','MikaSantos@gmail.com','09678901234','Tanauan City,
Batangas','Saturday-Wednesday, 11:00 AM - 8:00 PM');
```

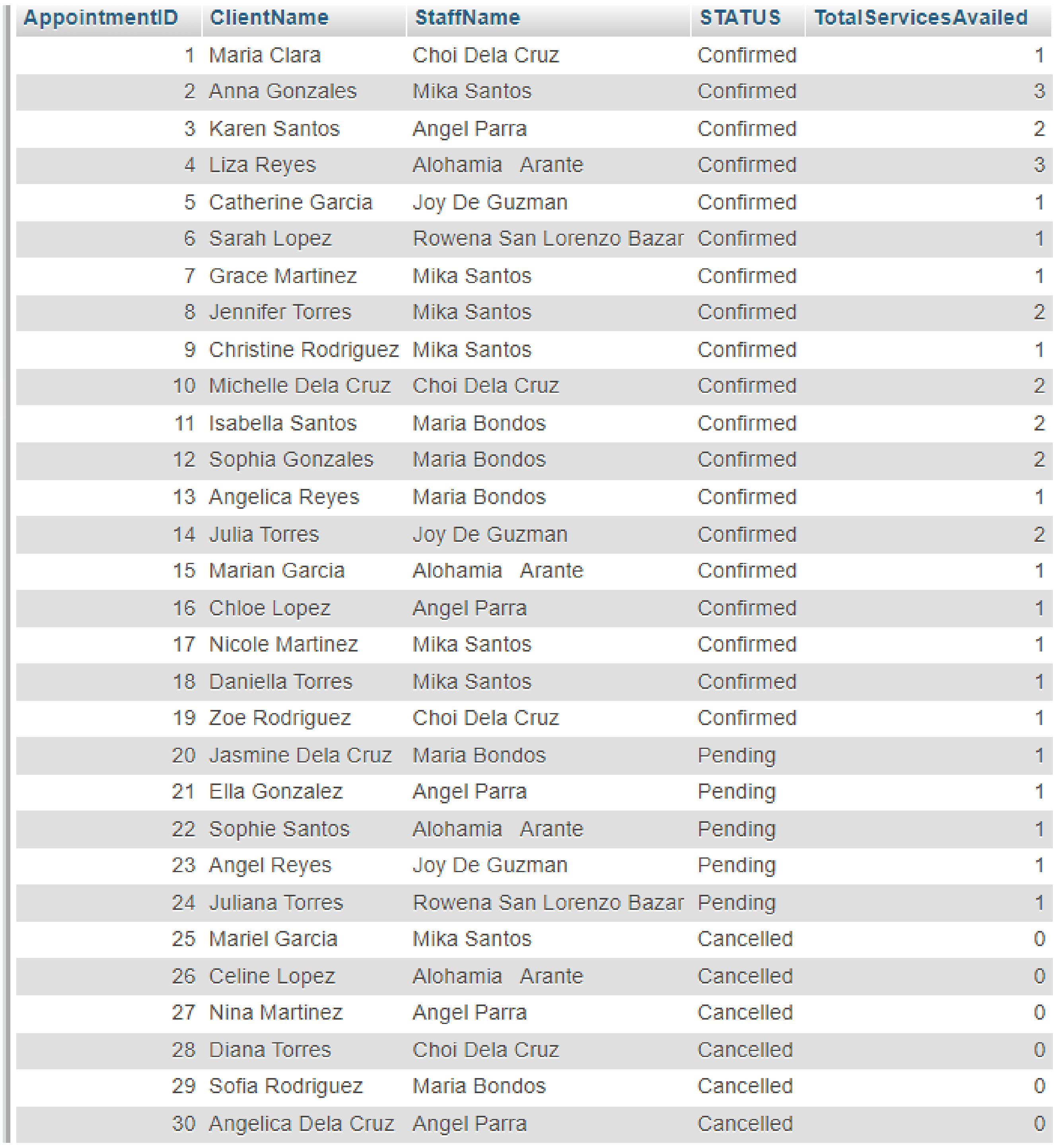
SQL Queries to Retrieve Information in the Database

1. **Retrieve all appointments from the appointment table, including the client's first and last name, the associated staff, the status, and the total services availed.**

   The query begins by selecting data from the appointment table, which stores information about appointments. It also retrieves data from the clients table to associate clients with their respective appointments. The staff table is included to link staff members with their assigned appointments. Using the CONCAT function, the client's first and last name are merged into the ClientName, and similarly for the staff's first and last name, creating the StaffName. By using COUNT(DISTINCT service.ServiceID) + COUNT(DISTINCT appointmentservice.ServiceID), the query calculates th e total number of unique services availed during each appointment. It utilizes an inner join and only a left join in the appointments ervice table to fetch all appointments in the result, whether they have associated services or not. A Case Expression is employed to verify whether the status is 'cancelled'. If it is, the TotalServicesAvailed column will be assigned a value of 0. This SQL query retrieves essential information about ap pointments, including the appointment ID, client name, staff name, appointment status, and the total number of services availed during each appointment .

```
SELECT
    appointment.AppointmentID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName,
    CONCAT(
        staff.S_FirstName,
        ' ',
        staff.S_LastName
    ) AS StaffName,
    appointment.STATUS,
    CASE WHEN appointment.STATUS = 'cancelled' THEN 0 ELSE COUNT(DISTINCT
service.ServiceID) + COUNT(
        DISTINCT appointmentservice.ServiceID
    )
END AS TotalServicesAvailed
FROM
    appointment
INNER JOIN clients ON appointment.ClientID = clients.ClientID
INNER JOIN staff ON staff.StaffID = appointment.StaffID
LEFT JOIN appointmentservice ON appointmentservice.AppointmentID =
appointment.AppointmentID
LEFT JOIN service ON appointment.ServiceID = service.ServiceID
GROUP BY
    appointment.AppointmentID;
```

Output:

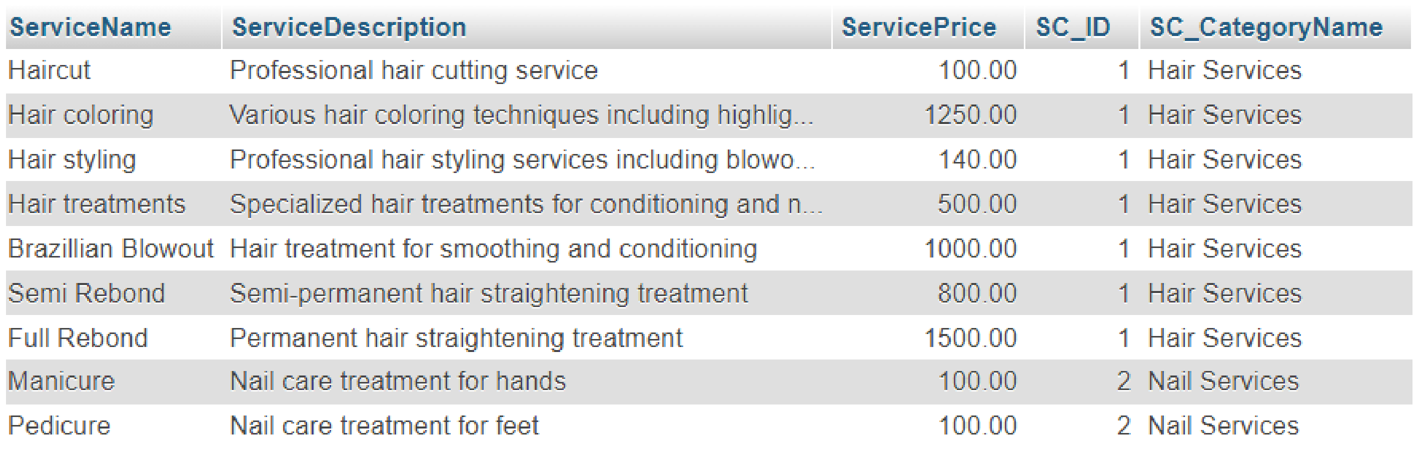


AppointmentID	ClientName	StaffName	STATUS	TotalServicesAvailed
1	Maria Clara	Choi Dela Cruz	Confirmed	1
2	Anna Gonzales	Mika Santos	Confirmed	3
3	Karen Santos	Angel Parra	Confirmed	2
4	Liza Reyes	Alohamia Arante	Confirmed	3
5	Catherine Garcia	Joy De Guzman	Confirmed	1
6	Sarah Lopez	Rowena San Lorenzo Bazar	Confirmed	1
7	Grace Martinez	Mika Santos	Confirmed	1
8	Jennifer Torres	Mika Santos	Confirmed	2
9	Christine Rodriguez	Mika Santos	Confirmed	1
10	Michelle Dela Cruz	Choi Dela Cruz	Confirmed	2
11	Isabella Santos	Maria Bondos	Confirmed	2
12	Sophia Gonzales	Maria Bondos	Confirmed	2
13	Angelica Reyes	Maria Bondos	Confirmed	1
14	Julia Torres	Joy De Guzman	Confirmed	2
15	Marian Garcia	Alohamia Arante	Confirmed	1
16	Chloe Lopez	Angel Parra	Confirmed	1
17	Nicole Martinez	Mika Santos	Confirmed	1
18	Daniella Torres	Mika Santos	Confirmed	1
19	Zoe Rodriguez	Choi Dela Cruz	Confirmed	1
20	Jasmine Dela Cruz	Maria Bondos	Pending	1
21	Ella Gonzalez	Angel Parra	Pending	1
22	Sophie Santos	Alohamia Arante	Pending	1
23	Angel Reyes	Joy De Guzman	Pending	1
24	Juliana Torres	Rowena San Lorenzo Bazar	Pending	1
25	Mariel Garcia	Mika Santos	Cancelled	0
26	Celine Lopez	Alohamia Arante	Cancelled	0
27	Nina Martinez	Angel Parra	Cancelled	0
28	Diana Torres	Choi Dela Cruz	Cancelled	0
29	Sofia Rodriguez	Maria Bondos	Cancelled	0
30	Angelica Dela Cruz	Angel Parra	Cancelled	0

2. **Retrieve all services along with their corresponding service categories.**
The SQL query retrieves information about services and their corresponding service categories from the database. It begins by selecting specific fields such as ServiceName, ServiceDescription, ServicePrice, and the SC_ID from the service table. Additionally, it includes the SC_CategoryName field from the servicecategory table. The query then utilizes an INNER JOIN operation to link the service table with the servicecategory table based on the SC_ID attribute. This association allows the query to retrieve the service category name corresponding to each service. Finally, the results are grouped by the unique ServiceID and SC_ID pairs to ensure accurate data representation. It enables businesses to organize and analyze service data efficiently, facilitating informed decision-making processes related to service management, pricing, and categorization.

```
SELECT
    service.ServiceName,
    service.ServiceDescription,
    service.ServicePrice,
    service.SC_ID,
    servicecategory.SC_CategoryName
FROM
    service
INNER JOIN
    servicecategory ON service.SC_ID = servicecategory.SC_ID
GROUP BY
    service.ServiceID, servicecategory.SC_ID;
```

Output:



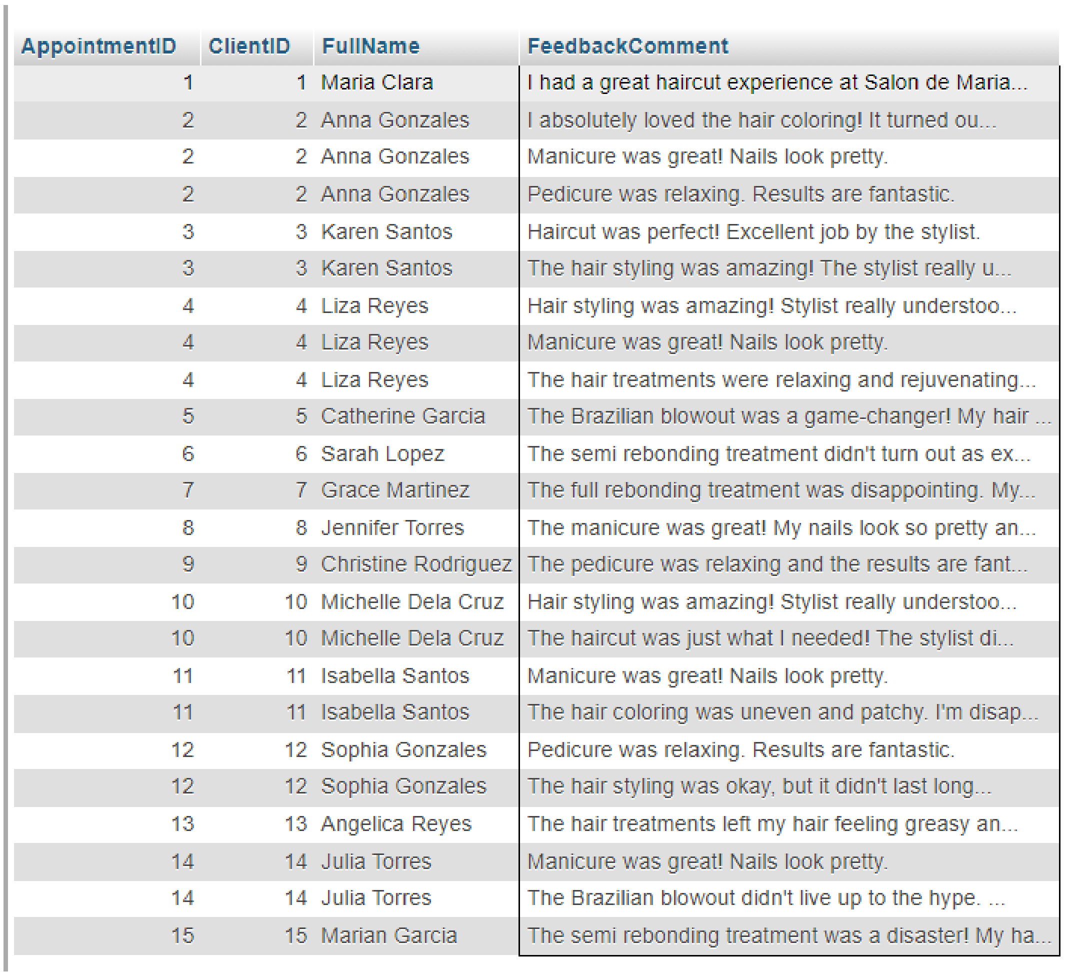
ServiceName	ServiceDescription	ServicePrice	SC_ID	SC_CategoryName
Haircut	Professional hair cutting service	100.00	1	Hair Services
Hair coloring	Various hair coloring techniques including highlig...	1250.00	1	Hair Services
Hair styling	Professional hair styling services including blowo...	140.00	1	Hair Services
Hair treatments	Specialized hair treatments for conditioning and n...	500.00	1	Hair Services
Brazillian Blowout	Hair treatment for smoothing and conditioning	1000.00	1	Hair Services
Semi Rebond	Semi-permanent hair straightening treatment	800.00	1	Hair Services
Full Rebond	Permanent hair straightening treatment	1500.00	1	Hair Services
Manicure	Nail care treatment for hands	100.00	2	Nail Services
Pedicure	Nail care treatment for feet	100.00	2	Nail Services

3. Retrieving All The Clients That Gave A Feedback

The SQL query retrieves information regarding appointments along with the feedback provided by clients, filtering out appointments with no feedback comments. It selects specific fields such as AppointmentID, ClientID, and a concatenated FullName derived from the client's first and last names. Additionally, it includes the FeedbackComment field from the feedback table. The query employs an INNER JOIN operation to associate appointments with their respective clients based on the ClientID attribute. It also utilizes a LEFT JOIN to link appointments with feedback based on the AppointmentID attribute. The WHERE clause filters out appointments with null feedback comments, ensuring that only appointments with client feedback are included in the results. The results are grouped by ClientID, AppointmentID, FullName, and FeedbackComment to ensure unique combinations of appointment and feedback details are retrieved.

```
SELECT
    appointment.AppointmentID,
    clients.ClientID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS FullName,
    feedback.FeedbackComment
FROM
    appointment
INNER JOIN clients ON appointment.ClientID = clients.ClientID
LEFT JOIN feedback ON appointment.AppointmentID = feedback.AppointmentID
WHERE
    feedback.FeedbackComment IS NOT NULL
GROUP BY
    clients.ClientID,
    appointment.AppointmentID,
    FullName,
    feedback.FeedbackComment
```

Output:



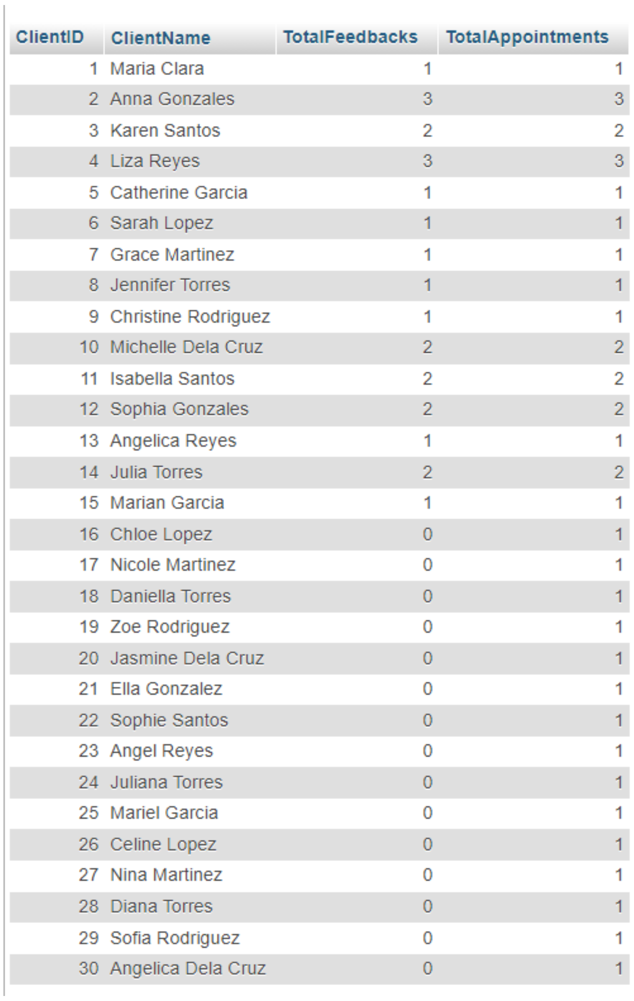
AppointmentID	ClientID	FullName	FeedbackComment
1	1	Maria Clara	I had a great haircut experience at Salon de Maria...
2	2	Anna Gonzales	I absolutely loved the hair coloring! It turned ou...
2	2	Anna Gonzales	Manicure was great! Nails look pretty.
2	2	Anna Gonzales	Pedicure was relaxing. Results are fantastic.
3	3	Karen Santos	Haircut was perfect! Excellent job by the stylist.
3	3	Karen Santos	The hair styling was amazing! The stylist really u...
4	4	Liza Reyes	Hair styling was amazing! Stylist really understoo...
4	4	Liza Reyes	Manicure was great! Nails look pretty.
4	4	Liza Reyes	The hair treatments were relaxing and rejuvenating...
5	5	Catherine Garcia	The Brazilian blowout was a game-changer! My hair ...
6	6	Sarah Lopez	The semi rebonding treatment didn't turn out as ex...
7	7	Grace Martinez	The full rebonding treatment was disappointing. My...
8	8	Jennifer Torres	The manicure was great! My nails look so pretty an...
9	9	Christine Rodriguez	The pedicure was relaxing and the results are fant...
10	10	Michelle Dela Cruz	Hair styling was amazing! Stylist really understoo...
10	10	Michelle Dela Cruz	The haircut was just what I needed! The stylist di...
11	11	Isabella Santos	Manicure was great! Nails look pretty.
11	11	Isabella Santos	The hair coloring was uneven and patchy. I'm disap...
12	12	Sophia Gonzales	Pedicure was relaxing. Results are fantastic.
12	12	Sophia Gonzales	The hair styling was okay, but it didn't last long...
13	13	Angelica Reyes	The hair treatments left my hair feeling greasy an...
14	14	Julia Torres	Manicure was great! Nails look pretty.
14	14	Julia Torres	The Brazilian blowout didn't live up to the hype. ...
15	15	Marian Garcia	The semi rebonding treatment was a disaster! My ha...

4. Retrieving The Total Appointment and Total Feedback of the Client

The SQL query retrieves data related to clients, including their total number of appointments and feedback submissions. It selects the ClientID and concatenates the first and last names of clients to form the ClientName. Two COUNT functions are used to calculate the total number of feedback and appointments associated with each client. The LEFT JOIN operation is employed to ensure that all clients are included in the results, regardless of whether they have appointments or feedback entries. The feedback and appointment tables are linked based on the AppointmentID attribute. The results are then grouped by ClientID, ClientName, feedback.AppointmentID, and appointment.AppointmentID to ensure distinct combinations of client, appointment, and feedback details.

```
SELECT
    clients.ClientID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName,
    COUNT(feedback.AppointmentID) AS TotalFeedbacks,
    COUNT(appointment.AppointmentID) AS TotalAppointments
FROM
    clients
LEFT JOIN appointment ON clients.ClientID = appointment.ClientID
LEFT JOIN feedback ON appointment.AppointmentID = feedback.AppointmentID
GROUP BY
    clients.ClientID,
    ClientName,
    feedback.AppointmentID,
    appointment.AppointmentID
```

Output:



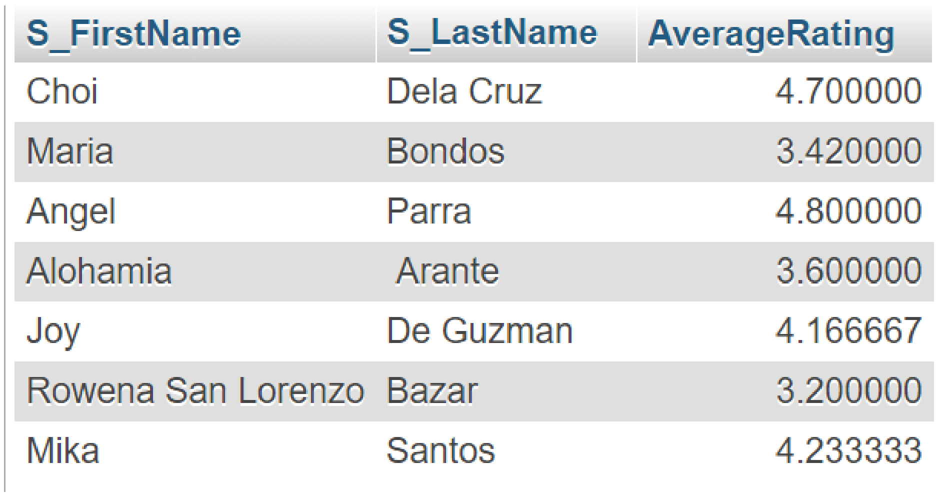
ClientID	ClientName	TotalFeedbacks	TotalAppointments
1	Maria Clara	1	1
2	Anna Gonzales	3	3
3	Karen Santos	2	2
4	Liza Reyes	3	3
5	Catherine Garcia	1	1
6	Sarah Lopez	1	1
7	Grace Martinez	1	1
8	Jennifer Torres	1	1
9	Christine Rodriguez	1	1
10	Michelle Dela Cruz	2	2
11	Isabella Santos	2	2
12	Sophia Gonzales	2	2
13	Angelica Reyes	1	1
14	Julia Torres	2	2
15	Marian Garcia	1	1
16	Chloe Lopez	0	1
17	Nicole Martinez	0	1
18	Daniella Torres	0	1
19	Zoe Rodriguez	0	1
20	Jasmine Dela Cruz	0	1
21	Ella Gonzalez	0	1
22	Sophie Santos	0	1
23	Angel Reyes	0	1
24	Juliana Torres	0	1
25	Mariel Garcia	0	1
26	Celine Lopez	0	1
27	Nina Martinez	0	1
28	Diana Torres	0	1
29	Sofia Rodriguez	0	1
30	Angelica Dela Cruz	0	1

5. Retrieve the Average Feedback Rating for Each Staff Member

The SQL query retrieves the average feedback rating for each staff member. It begins by selecting the first name and last name of staff members along with the average feedback rating calculated from the FeedbackRate. The query performs inner joins between the Staff, Appointment, and Feedback tables based on their corresponding IDs to link staff members with their appointments and feedback. Then, it calculates the average feedback rating for each staff member using the AVG() function. Finally, the results are grouped by the StaffID. This query is crucial for evaluating the performance of staff members based on customer feedback, which can help in improving service quality and customer satisfaction.

```
SELECT
    staff.S_FirstName,
    staff.S_LastName,
    AVG(FeedbackRate) AS AverageRating
FROM
    Staff
INNER JOIN Appointment ON Staff.StaffID = Appointment.StaffID
INNER JOIN Feedback ON Appointment.AppointmentID = Feedback.AppointmentID
GROUP BY
    Staff.StaffID;
```

Output:



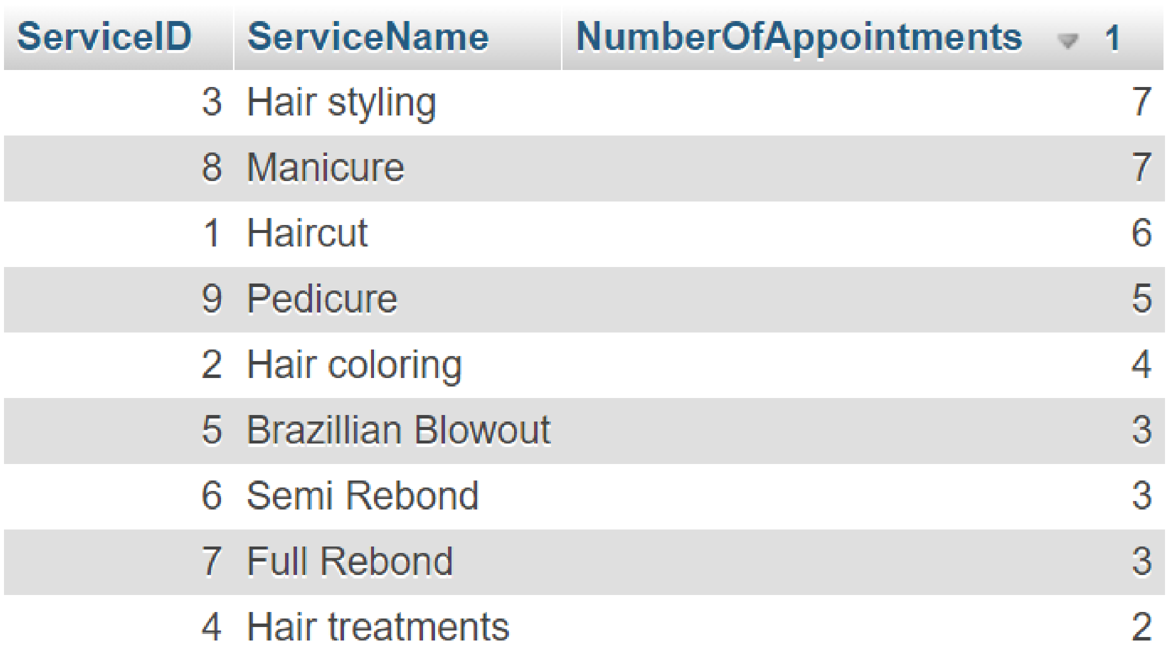
S_FirstName	S_LastName	AverageRating
Choi	Dela Cruz	4.700000
Maria	Bondos	3.420000
Angel	Parra	4.800000
Alohamia	Arante	3.600000
Joy	De Guzman	4.166667
Rowena San Lorenzo	Bazar	3.200000
Mika	Santos	4.233333

6. Determine the Most Popular Service Based on Appointment Count

The SQL query retrieves the number of appointments for each service provided. It selects the ServiceID and ServiceName from the Service table, alongside the total count of appointments associated with each service. The query utilizes left joins with the AppointmentService and Appointment tables to ensure all services are included, even if they have no appointments. The COUNT() function is used to calculate the total number of appointments for each service, considering both the AppointmentService and Appointment tables. The results are grouped by ServiceID and ServiceName and ordered in descending order based on the number of appointments. This query helps in identifying the popularity of different services and allows for better resource allocation and service planning within the organization.

```
SELECT
    Service.ServiceID,
    Service.ServiceName,
    COUNT(DISTINCT AppointmentService.AppointmentID) + COUNT(DISTINCT
Appointment.AppointmentID) AS NumberOfAppointments
FROM
    Service
LEFT JOIN
    AppointmentService ON Service.ServiceID = AppointmentService.ServiceID
LEFT JOIN
    Appointment ON Appointment.ServiceID = Service.ServiceID
GROUP BY
    Service.ServiceID,
    Service.ServiceName
ORDER BY
    NumberOfAppointments DESC;
```

Output:



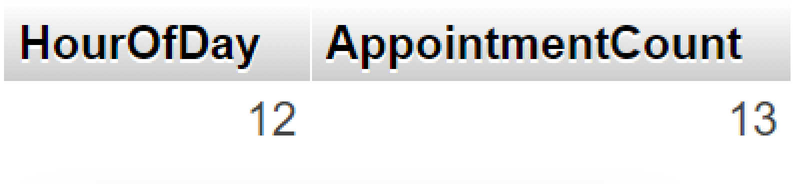
ServiceID	ServiceName	NumberOfAppointments ▼ 1
3	Hair styling	7
8	Manicure	7
1	Haircut	6
9	Pedicure	5
2	Hair coloring	4
5	Brazillian Blowout	3
6	Semi Rebond	3
7	Full Rebond	3
4	Hair treatments	2

7. Identify the Business Peak Time for Appointments

The SQL query retrieves the count of appointments scheduled for each hour of the day. It begins by selecting the hour of the appointment date and time using the HOUR() function applied to the App_DATETIME column of the Appointment table. The COUNT() function is then used to calculate the number of appointments for each hour. The appointments are grouped by the hour of the day using the GROUP BY clause. Finally, the result is ordered in descending order based on the appointment count per hour. This query is valuable for analyzing the distribution of appointment bookings throughout the day, which can help businesses identify peak hours of operation, optimize staffing schedules, and improve overall operational efficiency.

```sql
SELECT
    HOUR(appointment.App_DATETIME) AS HourOfDay,
    COUNT(*) AS AppointmentCount
FROM
    Appointment
GROUP BY
    HOUR(appointment.App_DATETIME)
ORDER BY
    AppointmentCount
DESC
LIMIT 1;
```

Output:



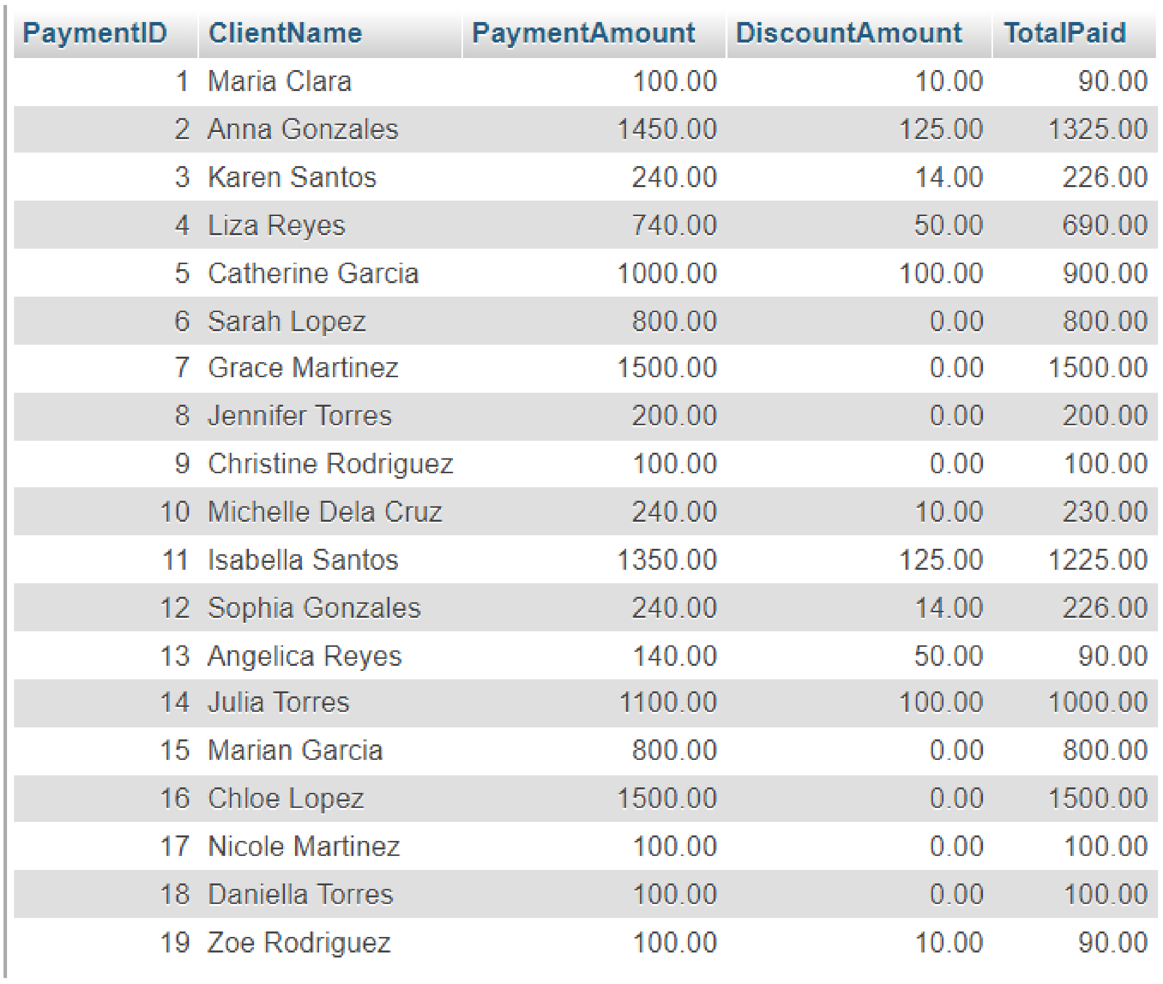
HourOfDay	AppointmentCount
12	13

8. Retrieve the Client's Payment History

This SQL query provides comprehensive payment information associated with appointments, encompassing the payment ID, client name, payment amount, discount amount, and the total amount paid after considering any applicable discounts. It initiates data retrieval from the Payment table and establishes connections with the Appointment, Discount, and Clients tables through appropriate join operations, ensuring a holistic representation of payment details for each appointment. The left join with the Discount table accommodates scenarios where appointments may or may not have associated discounts. To handle potential NULL values in the DiscountAmount column, the COALESCE() function is employed. If a discount amount exists in the discount.Disc_Amount column, it is displayed; otherwise, it defaults to 0. This approach enhances data consistency in the DiscountAmount column. The expression (payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)) calculates the TotalPaid amount for each payment. By subtracting the discount amount (if available) from the total payment amount, it reveals the actual amount paid by the client after factoring in any discounts. This query essentially tracks the payments that have discount amount which is critical for revenue analysis.

```sql
SELECT
    payment.PaymentID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName,
    payment.PaymentAmount,
    COALESCE(discount.Disc_Amount, 0) AS DiscountAmount,
    (
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) AS TotalPaid
FROM
    Payment
INNER JOIN Appointment ON payment.AppointmentID = appointment.AppointmentID
LEFT JOIN Discount ON appointment.DiscountID = discount.DiscountID
INNER JOIN clients ON appointment.ClientID = clients.ClientID;
```

Output:



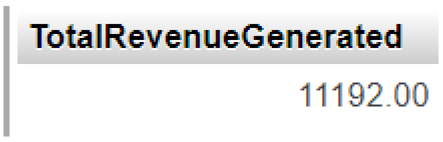
PaymentID	ClientName	PaymentAmount	DiscountAmount	TotalPaid
1	Maria Clara	100.00	10.00	90.00
2	Anna Gonzales	1450.00	125.00	1325.00
3	Karen Santos	240.00	14.00	226.00
4	Liza Reyes	740.00	50.00	690.00
5	Catherine Garcia	1000.00	100.00	900.00
6	Sarah Lopez	800.00	0.00	800.00
7	Grace Martinez	1500.00	0.00	1500.00
8	Jennifer Torres	200.00	0.00	200.00
9	Christine Rodriguez	100.00	0.00	100.00
10	Michelle Dela Cruz	240.00	10.00	230.00
11	Isabella Santos	1350.00	125.00	1225.00
12	Sophia Gonzales	240.00	14.00	226.00
13	Angelica Reyes	140.00	50.00	90.00
14	Julia Torres	1100.00	100.00	1000.00
15	Marian Garcia	800.00	0.00	800.00
16	Chloe Lopez	1500.00	0.00	1500.00
17	Nicole Martinez	100.00	0.00	100.00
18	Daniella Torres	100.00	0.00	100.00
19	Zoe Rodriguez	100.00	10.00	90.00

9. Retrieve The Total Revenue Generated

This SQL query calculates the total revenue generated from payments for appointments, accounting for any applied discounts. It begins by selecting the sum of the difference between the payment amount and the discount amount (if any) from the Payment table. The INNER JOIN operation links the Payment table with the Appointment table based on the appointment ID, ensuring that only valid payments associated with appointments are considered. Additionally, a LEFT JOIN operation with the Discount table allows for the inclusion of any discounts applied to appointments. The SUM() function aggregates the calculated revenue across all payments, providing a comprehensive overview of the total revenue generated from appointments after factoring in discounts. This query is valuable for analyzing the total revenue generated of the salon.

```sql
SELECT
    SUM(
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) AS TotalRevenueGenerated
FROM
    Payment
INNER JOIN Appointment ON payment.AppointmentID = appointment.AppointmentID
LEFT JOIN Discount ON appointment.DiscountID = discount.DiscountID;
```

Output:



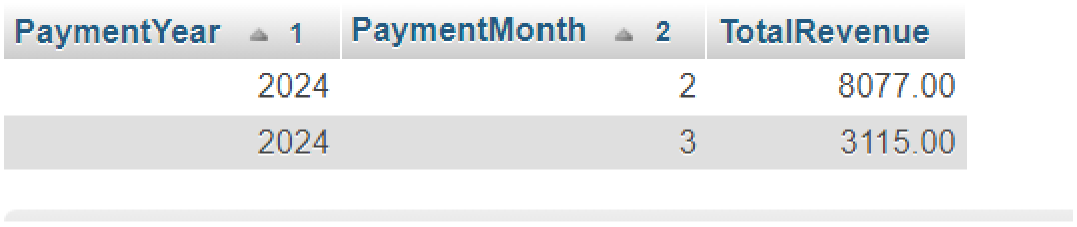
```
TotalRevenueGenerated
             11192.00
```

10. Calculate The Total Revenue Every Month and Year

This SQL query retrieves the total revenue generated from payments for appointments, grouped by year and month of the payment date. It extracts the year and month components from the PaymentDate column using the YEAR() and MONTH() functions, respectively. The SUM() function calculates the total revenue by subtracting any applied discounts from the payment amount. The INNER JOIN operation links the Payment table with the Appointment table based on the appointment ID, ensuring only valid payments associated with appointments are considered. Additionally, a LEFT JOIN operation with the Discount table allows for the inclusion of any discounts applied to appointments. The GROUP BY clause organizes the data by payment year and month, facilitating the aggregation of revenue for each period. This query is valuable for analyzing revenue trends over time, identifying peak revenue periods, and evaluating the impact of discounts on overall revenue generation.

```
SELECT
    YEAR(payment.PaymentDate) AS PaymentYear,
    MONTH(payment.PaymentDate) AS PaymentMonth,
    SUM(
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) AS TotalRevenue
FROM
    Payment
INNER JOIN Appointment ON payment.AppointmentID = appointment.AppointmentID
LEFT JOIN Discount ON appointment.DiscountID = discount.DiscountID
GROUP BY
    YEAR(payment.PaymentDate),
    MONTH(payment.PaymentDate)
ORDER BY
    PaymentYear ASC,
    PaymentMonth ASC;
```

Output:



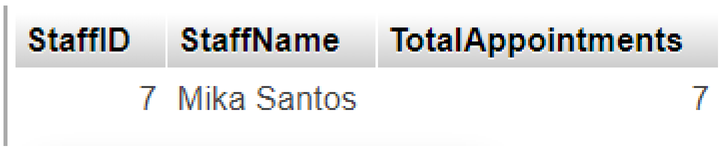
PaymentYear ▲ 1	PaymentMonth ▲ 2	TotalRevenue
2024	2	8077.00
2024	3	3115.00

11. Find Staff with Highest Number of Appointments

This SQL query retrieves the staff member with the highest number of appointments. It begins by selecting the staff ID and concatenating the first and last names to form the staff name. The COUNT() function is then used to calculate the total number of distinct appointments associated with each staff member. The query performs a LEFT JOIN operation between the Staff table and the Appointment table based on the staff ID to ensure that all staff members are included in the result set, regardless of whether they have appointments or not. Additionally, a LEFT JOIN operation with the AppointmentService table allows for the inclusion of appointments associated with services. The results are grouped by staff ID, first name, and last name, and ordered in descending order of total appointments. Finally, the LIMIT 1 clause ensures that only the staff member with the highest number of appointments is returned. This query is useful for identifying staff members who are the busiest in terms of appointment bookings, which can be valuable for optimizing staff schedules and allocating resources efficiently.

```
SELECT
    staff.StaffID,
    CONCAT(
        staff.S_FirstName,
        ' ',
        staff.S_LastName
    ) AS StaffName,
    COUNT(
        DISTINCT appointment.AppointmentID
    ) AS TotalAppointments
FROM
    staff
LEFT JOIN appointment ON staff.StaffID = appointment.StaffID
LEFT JOIN appointmentservice ON appointment.AppointmentID =
appointmentservice.AppointmentID
GROUP BY
    staff.StaffID,
    staff.S_FirstName,
    staff.S_LastName
ORDER BY
    TotalAppointments
DESC
LIMIT 1;
```

Output:



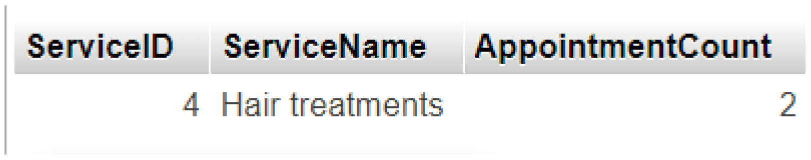
StaffID	StaffName	TotalAppointments
7	Mika Santos	7

12. Identify the Service with Lowest Appointment Rate

This SQL query is designed to identify the service with the lowest number of appointments. It begins by selecting the service ID and service name from the Service table. The query utilizes LEFT JOIN operations to link the Service table with both the AppointmentService and Appointment tables. This ensures that all services are included in the result set, even if they have no associated appointments. The COUNT() function is used to calculate the total number of distinct appointments for each service. The count includes appointments from both the AppointmentService and Appointment tables, as indicated by the JOIN conditions. The results are then grouped by service ID and service name. The ORDER BY clause sorts the services based on the appointment count in ascending order. Finally, the LIMIT 1 clause ensures that only the service with the lowest appointment count is returned. This query is useful for identifying services that may be less popular or in less demand compared to others.

```
SELECT
    service.ServiceID,
    service.ServiceName,
    COUNT(
        DISTINCT AppointmentService.AppointmentID
    ) + COUNT(
        DISTINCT Appointment.AppointmentID
    ) AS AppointmentCount
FROM
    service
LEFT JOIN appointmentservice ON service.ServiceID = appointmentservice.ServiceID
LEFT JOIN Appointment ON Appointment.ServiceID = Service.ServiceID
GROUP BY
    service.ServiceID,
    service.ServiceName
ORDER BY
    AppointmentCount ASC
LIMIT 1;
```

Output:



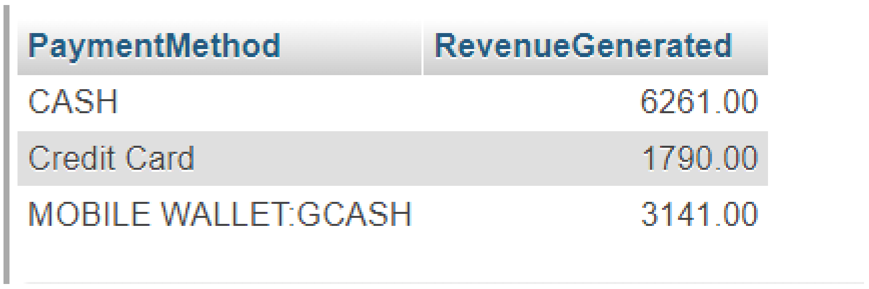
ServiceID	ServiceName	AppointmentCount
4	Hair treatments	2

13. Retrieve the revenue generated from each payment method

This SQL query calculates the revenue generated from different payment methods for services provided by Salon de Maria. It selects the PaymentMethod column from the Payment table and computes the total revenue generated for each payment method by subtracting any applicable discounts from the PaymentAmount using the COALESCE() function to handle potential NULL values in the Discount amount. The query utilizes several joins to fetch relevant data: an INNER JOIN with the Appointment table to link payments with appointments, a LEFT JOIN with the Discount table to incorporate any discounts applied to appointments, and another INNER JOIN with the Service table to connect appointments with the services provided. By grouping the results by PaymentMethod, the query aggregates the revenue data, providing insights into which payment methods contribute the most revenue to Salon de Maria. This analysis helps the salon understand customer payment preferences and enables them to tailor their payment processing strategies accordingly.

```
SELECT
    PaymentMethod,
    SUM(
        PaymentAmount - COALESCE(Disc_Amount, 0)
    ) AS RevenueGenerated
FROM
    Payment
INNER JOIN Appointment ON Payment.AppointmentID = Appointment.AppointmentID
LEFT JOIN Discount ON Appointment.DiscountID = Discount.DiscountID
INNER JOIN Service ON Appointment.ServiceID = Service.ServiceID
GROUP BY
    PaymentMethod;
```

Output:



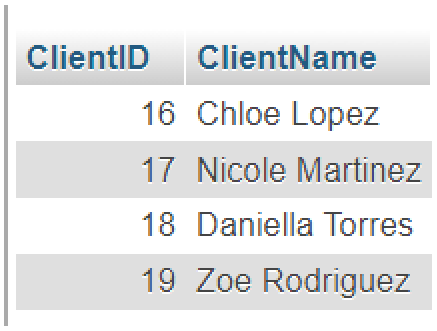
PaymentMethod	RevenueGenerated
CASH	6261.00
Credit Card	1790.00
MOBILE WALLET:GCASH	3141.00

14. Identify Clients Who Have Not Provided Feedback

This SQL query is designed to identify clients who have not provided feedback for their confirmed appointments. It selects the client ID and concatenates the client's first and last name to form the client name from the clients table. The query utilizes LEFT JOIN operations to link the clients table with the appointment table based on the client ID, and then with the feedback table based on the appointmentID. To filter out clients who have not provided feedback, the WHERE clause is used to specify conditions where the feedback.AppointmentID is NULL, indicating no feedback exists for the appointment, and where the appointment status is 'Confirmed'. The significance of this query lies in its ability to identify clients who may have missed providing feedback for their confirmed appointments. This query is significant can to follow up with clients, gather feedback to improve services, and ensure customer satisfaction and engagement.

```
SELECT
    clients.ClientID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName
FROM
    clients
LEFT JOIN appointment ON clients.ClientID = appointment.ClientID
LEFT JOIN feedback ON appointment.AppointmentID = feedback.AppointmentID
WHERE
    feedback.AppointmentID IS NULL AND appointment.STATUS = 'Confirmed';
```

Output:



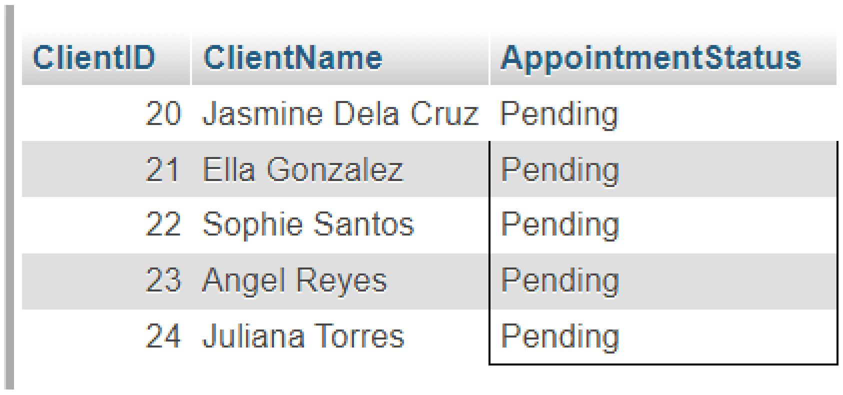
ClientID	ClientName
16	Chloe Lopez
17	Nicole Martinez
18	Daniella Torres
19	Zoe Rodriguez

15. Identify clients with pending appointments

This SQL query retrieves information about clients who have pending appointments. It selects the client ID and concatenates the client's first and last name to form the client name from the clients table. Additionally, it includes a status value 'Pending' as the AppointmentStatus for each record. The query employs an INNER JOIN operation to link the clients table with the appointment table based on the client ID. It then filters the results using a WHERE clause to include only appointments with a status of 'Pending'. With this query, we can retrieve the information to manage appointment schedules, follow up with clients to confirm appointments, and ensure timely service appointment and customer satisfaction.

```
SELECT
    clients.ClientID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName,
    'Pending' AS AppointmentStatus
FROM
    clients
INNER JOIN appointment ON clients.ClientID = appointment.ClientID
WHERE
    appointment.STATUS = 'Pending';
```

Output:



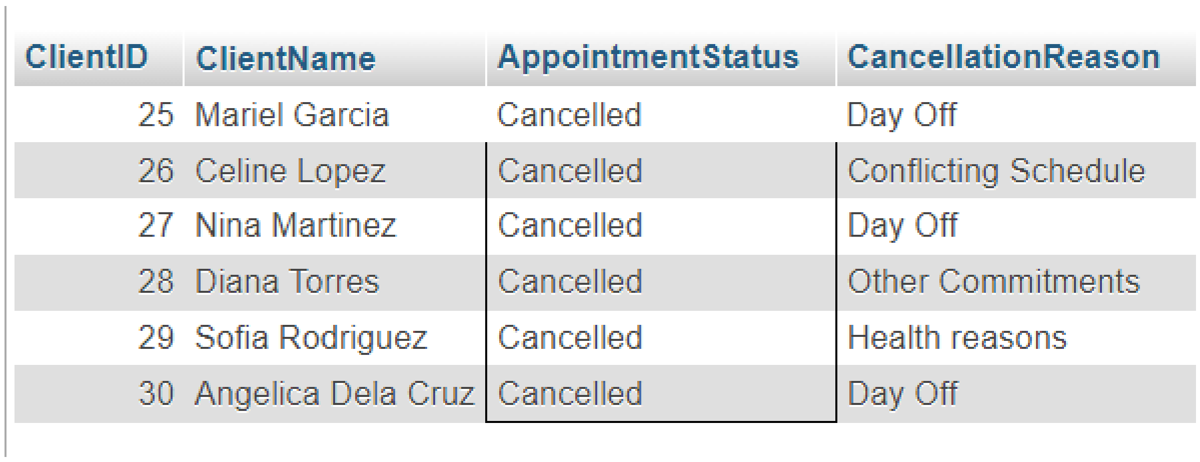
ClientID	ClientName	AppointmentStatus
20	Jasmine Dela Cruz	Pending
21	Ella Gonzalez	Pending
22	Sophie Santos	Pending
23	Angel Reyes	Pending
24	Juliana Torres	Pending

16. Identifying clients whose appointments got cancelled

This SQL query retrieves information about clients whose appointments have been cancelled. It selects the client ID and concatenates the client's first and last name to form the client name from the clients table. Additionally, it includes a static value 'Cancelled' as the AppointmentStatus for each record, along with the cancellation reason from the appointment table. The query employs an INNER JOIN operation to link the clients table with the appointment table based on the client ID. It then filters the results using a WHERE clause to include only appointments with a status of 'Cancelled'. The significance of this query lies in its ability to provide insights into cancelled appointments and the reasons behind them, and take proactive measures to improve customer satisfaction and appointment management processes.

```sql
SELECT
    clients.ClientID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName,
    'Cancelled' AS AppointmentStatus,
    appointment.CancellationReason
FROM
    clients
INNER JOIN appointment ON clients.ClientID = appointment.ClientID
WHERE
    appointment.STATUS = 'Cancelled';
```

Output:



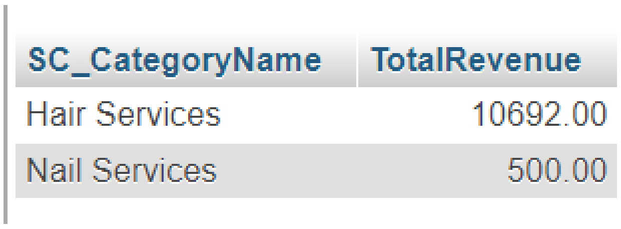
ClientID	ClientName	AppointmentStatus	CancellationReason
25	Mariel Garcia	Cancelled	Day Off
26	Celine Lopez	Cancelled	Conflicting Schedule
27	Nina Martinez	Cancelled	Day Off
28	Diana Torres	Cancelled	Other Commitments
29	Sofia Rodriguez	Cancelled	Health reasons
30	Angelica Dela Cruz	Cancelled	Day Off

17. Identify The Generated Sales From Each Category

This SQL query retrieves total revenue generated from each service category by subtracting any discounts applied from the payment amount in the Payment table. It joins the Payment, Appointment, Discount, Service, and ServiceCategory tables to establish relationships between payments, appointments, services, and service categories. The Payment table is joined with the Appointment table based on the appointment ID, while the Discount table is left-joined with the Appointment table to account for any discounts. Additionally, the Service table is joined with the Appointment table to associate services with appointments, and the ServiceCategory table is joined to categorize services by their respective categories. By using the GROUP BY clause on the service category name, the query aggregates total revenue for each service category. This query provides insights into revenue generated from different service categories, aiding in the analysis of performance or productivity.

```
SELECT
    servicecategory.SC_CategoryName,
    SUM(
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) AS TotalRevenue
FROM
    Payment
INNER JOIN Appointment ON Payment.AppointmentID = Appointment.AppointmentID
LEFT JOIN Discount ON Appointment.DiscountID = Discount.DiscountID
INNER JOIN Service ON Appointment.ServiceID = Service.ServiceID
INNER JOIN ServiceCategory ON Service.SC_ID = servicecategory.SC_ID
GROUP BY
    servicecategory.SC_CategoryName;
```

Output:



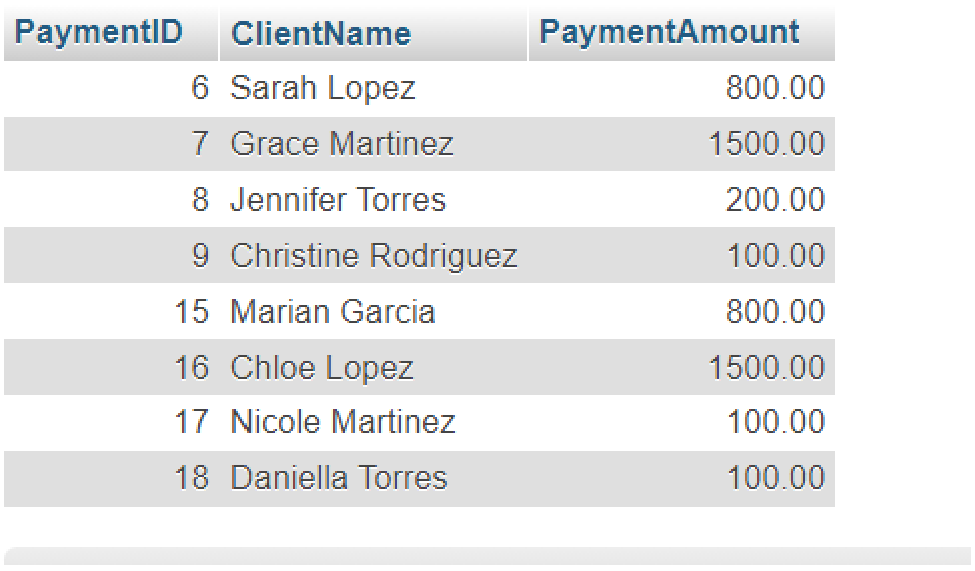
SC_CategoryName	TotalRevenue
Hair Services	10692.00
Nail Services	500.00

18. Identify the payments made without discount or promo

This SQL query retrieves payment information for confirmed appointments where no discount has been applied. It selects the payment ID, clien t name (concatenated from first and last name columns), and payment amount from the Payment table. The query performs an inner join with the Appointment table based on the appointment ID to link payment details with appointments. Another inner join is conducted with the Clients table to associate clients with their respective appointments. Additionally, a left join with the Discount table is employed to exclud e appointments where discounts have been applied. The WHERE clause filters the results to only include confirmed appointments and not the null ones. This query is valuable for analyzing payment details associated with confirmed appointments without discounts, enabling Salon De Maria to track revenue and understand client payment behavior effectively.

```
SELECT
    payment.PaymentID,
    CONCAT(
        clients.C_FirstName,
        ' ',
        clients.C_LastName
    ) AS ClientName,
    payment.PaymentAmount
FROM
    Payment
INNER JOIN Appointment ON payment.AppointmentID = appointment.AppointmentID
INNER JOIN clients ON appointment.ClientID = clients.ClientID
LEFT JOIN Discount ON appointment.DiscountID = discount.DiscountID
WHERE
    discount.DiscountID IS NULL AND appointment.STATUS = 'Confirmed';
```

Output:



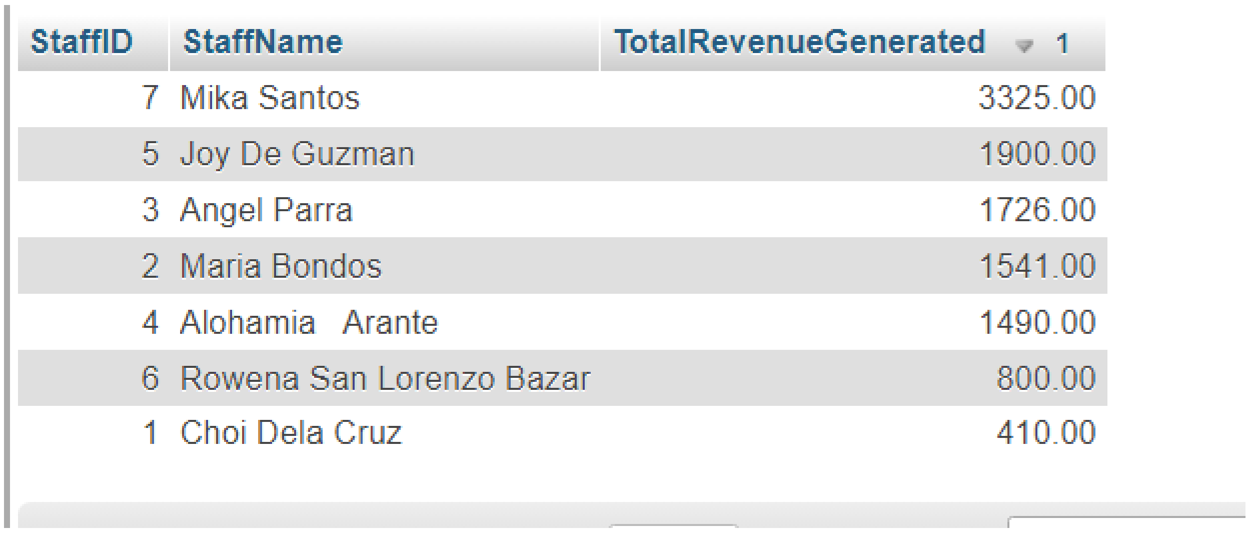
PaymentID	ClientName	PaymentAmount
6	Sarah Lopez	800.00
7	Grace Martinez	1500.00
8	Jennifer Torres	200.00
9	Christine Rodriguez	100.00
15	Marian Garcia	800.00
16	Chloe Lopez	1500.00
17	Nicole Martinez	100.00
18	Daniella Torres	100.00

19. Identify Staff Members with the Highest Total Revenue Generated

This SQL query calculates the total revenue generated by each staff member through appointments. It retrieves the staff ID, concatenated staff name, and the sum of payment amounts minus any discounts applied. The query involves joining the Payment table with the Appointment table based on the appointment ID and then linking the Appointment table with the Staff table using the staff ID. A left join with the Discount table is performed to include any discounts associated with appointments. The results are grouped by staff ID and ordered by total revenue generated in descending order. This query is instrumental in assessing the revenue contribution of individual staff members, aiding in performance evaluation and incentive management within the business.

```
SELECT
    staff.StaffID,
    CONCAT(
        staff.S_FirstName,
        ' ',
        staff.S_LastName
    ) AS StaffName,
    SUM(
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) AS TotalRevenueGenerated
FROM
    payment
INNER JOIN appointment ON payment.AppointmentID = appointment.AppointmentID
INNER JOIN staff ON appointment.StaffID = staff.StaffID
LEFT JOIN discount ON appointment.DiscountID = discount.DiscountID
GROUP BY
    staff.StaffID
ORDER BY
    TotalRevenueGenerated
DESC
    ;
```

Output:



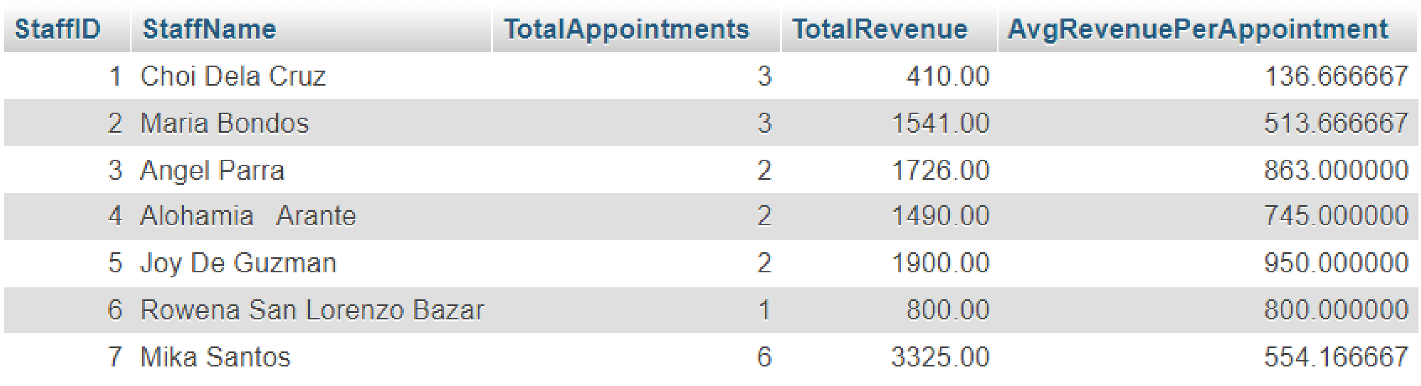
StaffID	StaffName	TotalRevenueGenerated ▼ 1
7	Mika Santos	3325.00
5	Joy De Guzman	1900.00
3	Angel Parra	1726.00
2	Maria Bondos	1541.00
4	Alohamia Arante	1490.00
6	Rowena San Lorenzo Bazar	800.00
1	Choi Dela Cruz	410.00

20. Calculate the Average Revenue per Appointment for Each Staff Member

This SQL query provides comprehensive insights into the performance of staff members by calculating various metrics related to appointments and revenue. It retrieves the staff ID, concatenated staff name, total number of appointments, total revenue generated, and average revenue per appointment. The query involves joining the Staff table with the Appointment and Payment tables based on the staff ID and appointment ID, respectively. A left join with the Discount table is performed to include any discounts associated with appointments. The results are filtered to only include appointments with a 'confirmed' status. The data is then grouped by staff ID and staff name. This query is valuable for evaluating staff productivity, revenue contribution, and efficiency in revenue generation per appointment, assisting in data-driven decision-making for staff management and business optimization.

```sql
SELECT
    staff.StaffID,
    CONCAT(
        staff.S_FirstName,
        ' ',
        staff.S_LastName
    ) AS StaffName,
    COUNT(appointment.AppointmentID) AS TotalAppointments,
    SUM(
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) AS TotalRevenue,
    SUM(
        payment.PaymentAmount - COALESCE(discount.Disc_Amount, 0)
    ) / COUNT(appointment.AppointmentID) AS AvgRevenuePerAppointment
FROM
    staff
INNER JOIN appointment ON staff.StaffID = appointment.StaffID
INNER JOIN payment ON appointment.AppointmentID = payment.AppointmentID
LEFT JOIN discount ON appointment.DiscountID = discount.DiscountID
WHERE
    appointment.Status = 'confirmed'
GROUP BY
    staff.StaffID,
    staff.S_FirstName,
    staff.S_LastName;
```

Output:



StaffID	StaffName	TotalAppointments	TotalRevenue	AvgRevenuePerAppointment
1	Choi Dela Cruz	3	410.00	136.666667
2	Maria Bondos	3	1541.00	513.666667
3	Angel Parra	2	1726.00	863.000000
4	Alohamia Arante	2	1490.00	745.000000
5	Joy De Guzman	2	1900.00	950.000000
6	Rowena San Lorenzo Bazar	1	800.00	800.000000
7	Mika Santos	6	3325.00	554.166667