



*VERSA CARE*

**GROUP C**

- **SRI LAKSHMI BONTA-Z1978059**
  - **VARUN SURESH – Z2001562**
  - **SAI KALYAN BILLALA – Z1978058**
  - **NEHA JUVVADI-Z1980493**
-

## **PROJECT DESCRIPTION**

Versa Care is a Home Service provider based out of US. It offers a wide range of services to customers across various cities in US. With a database at its core, Versa Care ensures efficient operations and seamless customer experiences. The company operates in several major cities including New York, Boston, San Francisco, Los Angeles, New Jersey with its headquarters situated in Chicago.

At the heart of Versa Care Company's operations lies its diverse range of services, categorized into different service categories such as beauty, home repairs, cleaning, wellness, and more. The company has a dedicated mobile app where each service is meticulously detailed, outlining the specific offerings and procedures. Service providers, who are integral to this company's ecosystem are registered within the system with information about their expertise, availability, and service history is stored.

Customers interact with Versa Care through its user-friendly mobile application or website, where they can create accounts and store their details for seamless booking experiences. Upon logging in, customers can browse through the various services available in their city and make bookings based on their requirements. These bookings are recorded in the system under the booking table, which tracks details such as booking id, service type, service provider assigned, customer id and booking status. Payment for services is handled through secure platform, with customers having the option to choose from various payment methods including credit/debit cards, pay-pal, digital wallets, or cash. Payment details associated with each booking are stored within the payment table, ensuring transparency and accountability.

Once service is completed and recorded by professional, a notification is sent to the customer via email/app/text message where customers are encouraged to provide feedback and ratings. This feedback not only helps Versa Care maintain service quality but also influences the incentives received by service providers. Positive feedback may result in incentives for service providers, motivating them to deliver exceptional service. In addition to service bookings and payments, Versa Care also manages promotional offers and discounts through the promotions for newly registered customers and loyal customers. These offers are tailored to attract more customers and enhance customer loyalty.

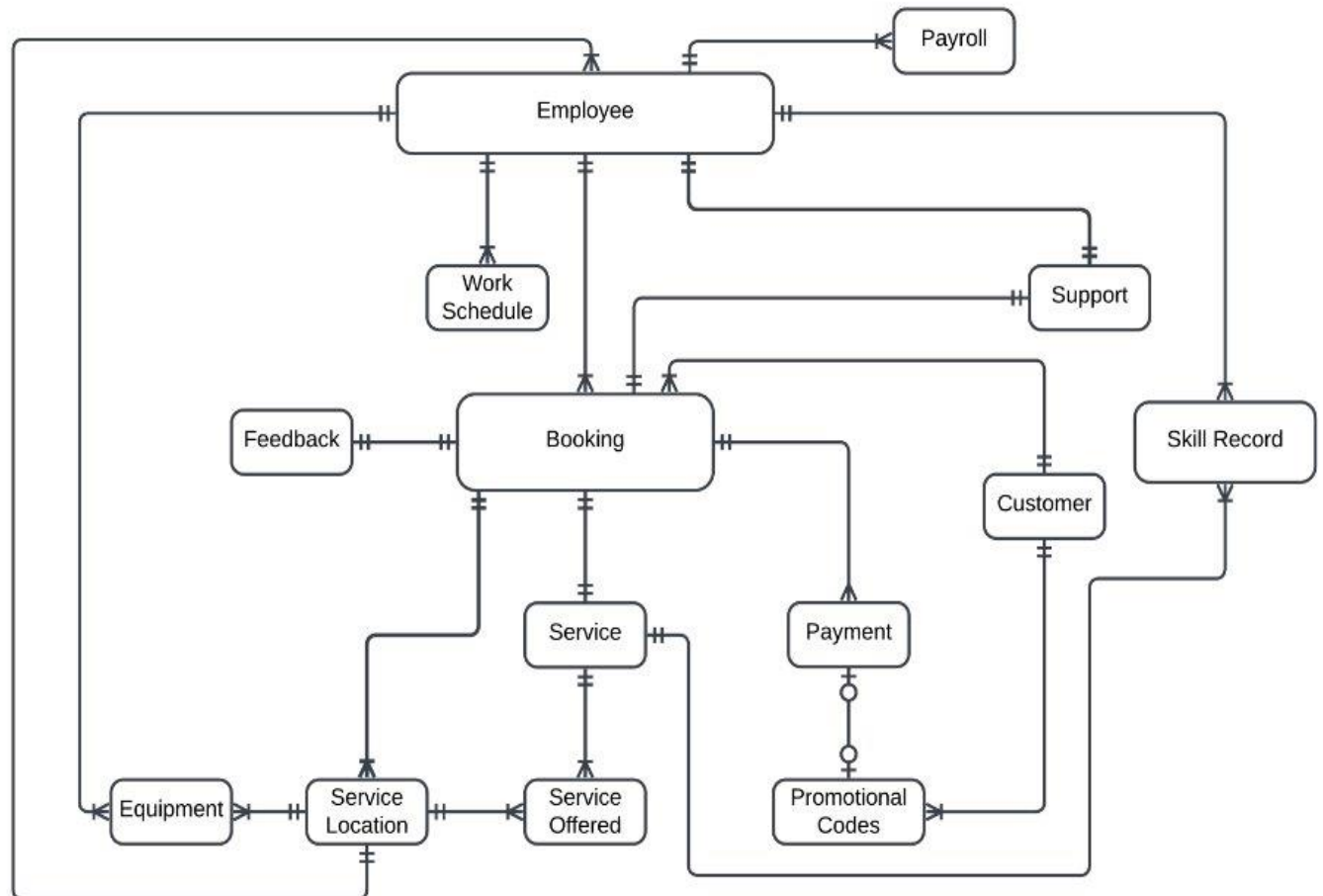
Behind the scenes, Versa Care partners with training centers to upskill their service providers. Details of training programs offered, and equipment provided to service providers are stored within the skill and equipment tables, ensuring that service quality is always maintained. Versa Care also places a strong emphasis on customer support, with a dedicated table to track support tickets and resolutions. This ensures that customer issues or queries are addressed promptly, further enhancing customer satisfaction and the company make sures that the same issue is not repeated to other customers.

Lastly, Versa Care maintains comprehensive employee records and payroll information for its internal staff through the employee and payroll tables respectively, ensuring smooth operations and employee satisfaction. Additionally, all financial transactions within the platform are recorded in the company's transaction table, providing a clear audit trail of financial activities. In essence, the company's database-driven approach facilitates seamless interactions between customers, service providers, and internal staff, enabling the company to deliver exceptional services while continuously improving its operations and customer satisfaction levels.

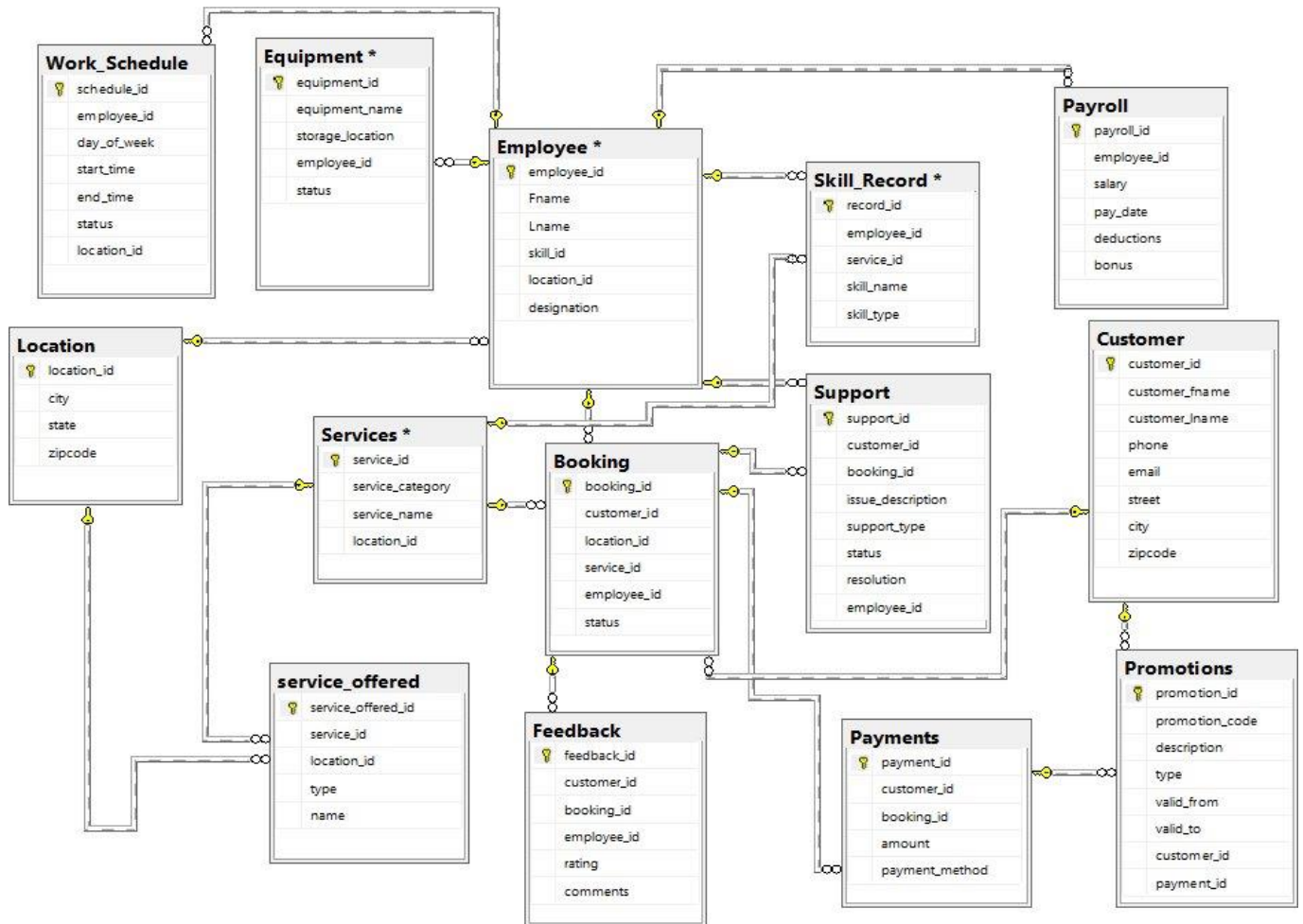
## **ENTITIES AND ATTRIBUTES**

1. **Customer:** customer\_id, customer\_fname, customer\_lname, phone, email, street, city, zipcode
2. **Booking :** booking\_id, customer\_id, location\_id, service\_id, employee\_id, status
3. **Services:** service\_id, service\_category, service\_name, location\_id
4. **Service offered:** service\_offered\_id, service\_id, location\_id, type, name
5. **Location:** location\_id, city, state, zipcode
6. **Equipment:** equipment\_id, equipment\_name, storage\_location, employee\_id, status
7. **Payment :** payment\_id, customer\_id, booking\_id, amount, payment\_method
8. **Promotions:** promotion\_id, promotion\_code, description, type, valid\_from, valid\_to, customer\_id, payment\_id
9. **Feedback:** feedback\_id, customer\_id, booking\_id, employee\_id, rating, comments
10. **Support :** support\_id, customer\_id, booking\_id, issue\_description, support\_type, status, resolution, employee\_id
11. **Employee:** employee\_id, Fname, Lname, skill\_id, location\_id, designation
12. **Skill Record:** record\_id, employee\_id, service\_id, skill\_name, skill\_type
13. **Work schedule:** schedule\_id, employee\_id, day\_of\_week, start\_time, end\_time, status, location\_id
14. **Payroll:** payroll\_id, employee\_id, salary, pay\_date, deductions, bonus

## ENTITY RELATIONSHIP DIAGRAM



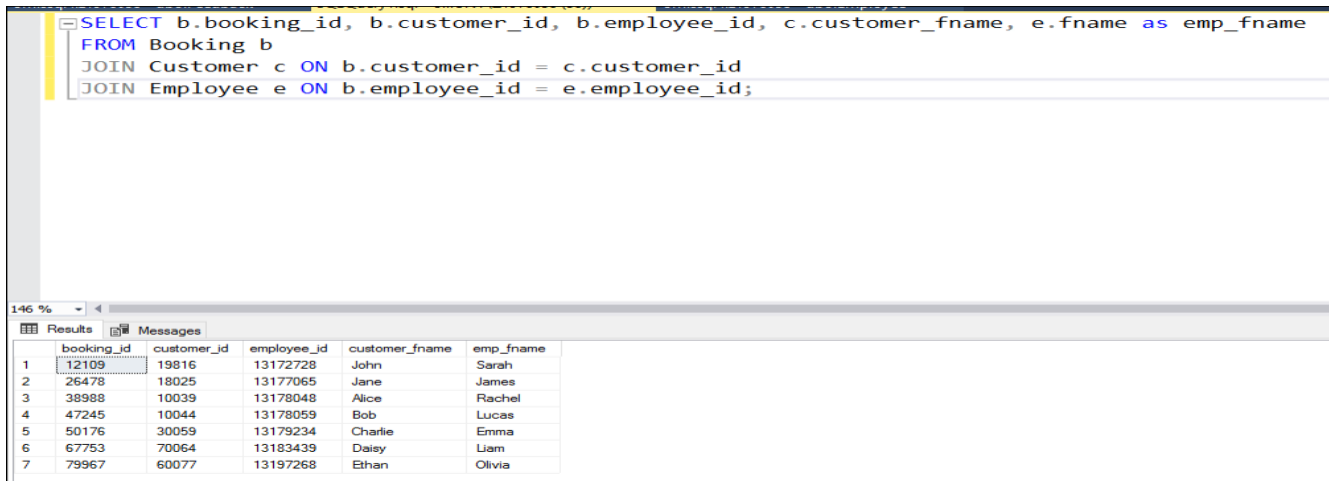
## DATABASE DIAGRAM



## SQL QUERIES

### 1. Find the booking details, customer information and employee details:

```
SELECT b.booking_id, b.customer_id, b.employee_id, c.customer_fname, e.fname as  
emp_fname  
FROM Booking b JOIN Customer c ON b.customer_id = c.customer_id  
JOIN Employee e ON b.employee_id = e.employee_id;
```



The screenshot shows a SQL query editor with the following query:

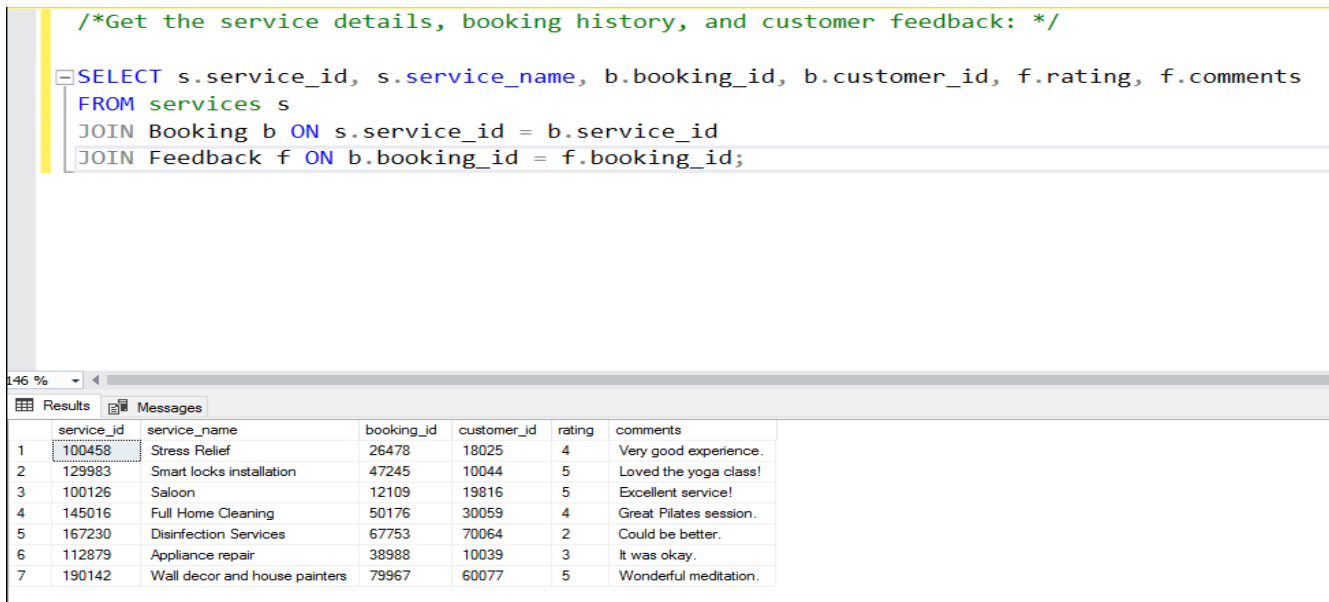
```
SELECT b.booking_id, b.customer_id, b.employee_id, c.customer_fname, e.fname as emp_fname  
FROM Booking b  
JOIN Customer c ON b.customer_id = c.customer_id  
JOIN Employee e ON b.employee_id = e.employee_id;
```

Below the query editor, the 'Results' tab is active, displaying the following data:

	booking_id	customer_id	employee_id	customer_fname	emp_fname
1	12109	19816	13172728	John	Sarah
2	26478	18025	13177065	Jane	James
3	38988	10039	13178048	Alice	Rachel
4	47245	10044	13178059	Bob	Lucas
5	50176	30059	13179234	Charlie	Emma
6	67753	70064	13183439	Daisy	Liam
7	79967	60077	13197268	Ethan	Olivia

### 2. Get the service details, booking history, and customer feedback:

```
SELECT s.service_id, s.service_name, b.booking_id, b.customer_id, f.rating, f.comments  
FROM services s JOIN Booking b ON s.service_id = b.service_id  
JOIN Feedback f ON b.booking_id = f.booking_id;
```



The screenshot shows a SQL query editor with the following query:

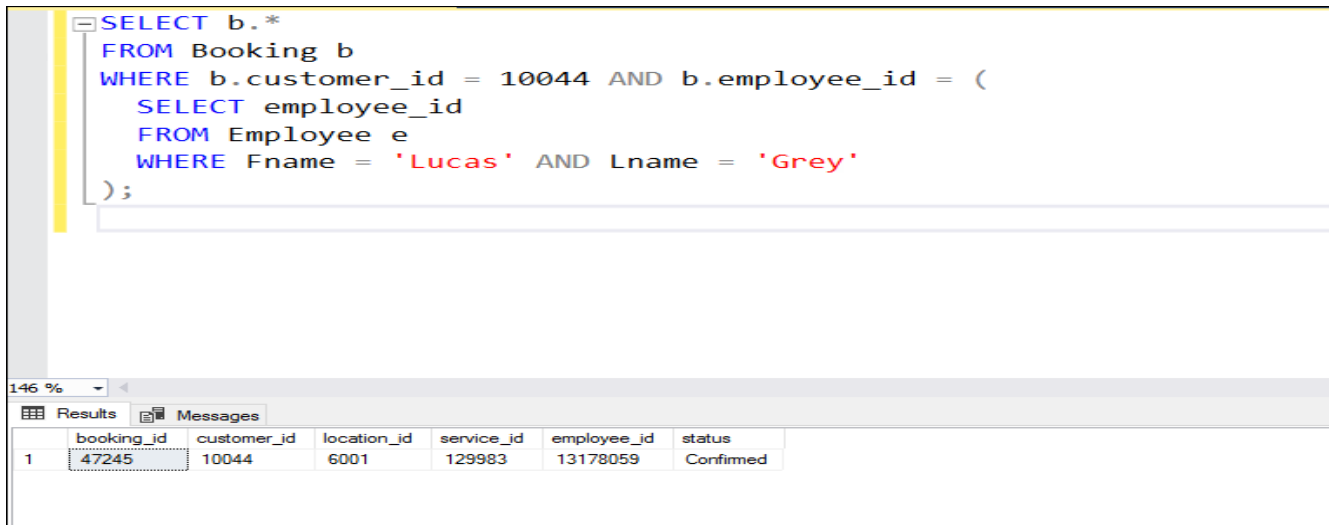
```
/*Get the service details, booking history, and customer feedback: */  
SELECT s.service_id, s.service_name, b.booking_id, b.customer_id, f.rating, f.comments  
FROM services s  
JOIN Booking b ON s.service_id = b.service_id  
JOIN Feedback f ON b.booking_id = f.booking_id;
```

Below the query editor, the 'Results' tab is active, displaying the following data:

	service_id	service_name	booking_id	customer_id	rating	comments
1	100458	Stress Relief	26478	18025	4	Very good experience.
2	129983	Smart locks installation	47245	10044	5	Loved the yoga class!
3	100126	Saloon	12109	19816	5	Excellent service!
4	145016	Full Home Cleaning	50176	30059	4	Great Pilates session.
5	167230	Disinfection Services	67753	70064	2	Could be better.
6	112879	Appliance repair	38988	10039	3	It was okay.
7	190142	Wall decor and house painters	79967	60077	5	Wonderful meditation.

### 3. Get the booking details for a specific customer and their preferred employee:

```
SELECT *
FROM Booking b
WHERE b.customer_id = 10044 AND b.employee_id IN
      (SELECT employee_id
       FROM Employee e
       WHERE Fname = 'Lucas' AND Lname = 'Grey');
```



The screenshot shows a SQL query editor with the following query:

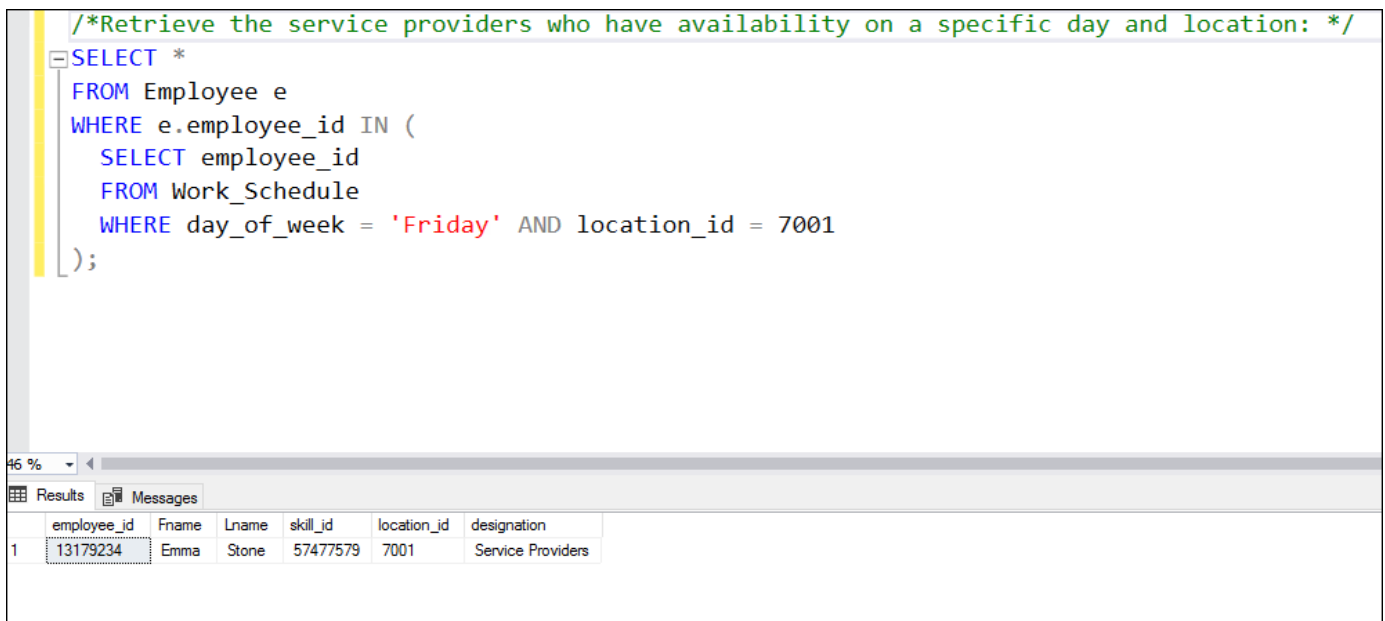
```
SELECT b.*
FROM Booking b
WHERE b.customer_id = 10044 AND b.employee_id = (
  SELECT employee_id
  FROM Employee e
  WHERE Fname = 'Lucas' AND Lname = 'Grey'
);
```

Below the query editor, the 'Results' tab is active, displaying a single row of data:

	booking_id	customer_id	location_id	service_id	employee_id	status
1	47245	10044	6001	129983	13178059	Confirmed

### 4. Retrieve the employees who have availability on a specific day and location:

```
SELECT *
FROM Employee e
WHERE e.employee_id IN
      (SELECT employee_id
       FROM Work_Schedule
       WHERE day_of_week = 'Friday' AND location_id = 7001 );
```



The screenshot shows a SQL query editor with the following query:

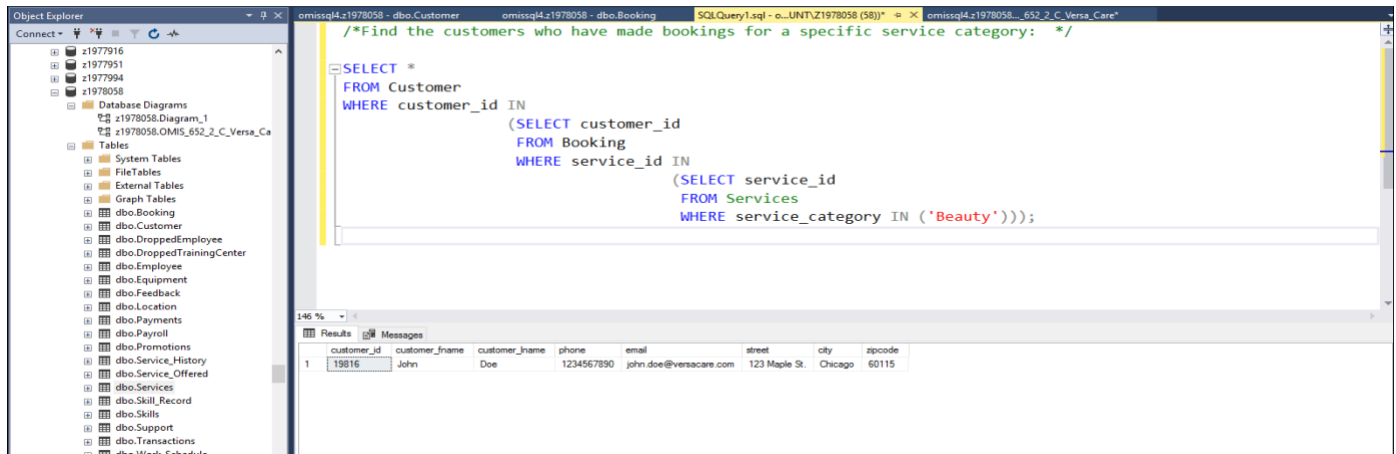
```
/*Retrieve the service providers who have availability on a specific day and location: */
SELECT *
FROM Employee e
WHERE e.employee_id IN (
  SELECT employee_id
  FROM Work_Schedule
  WHERE day_of_week = 'Friday' AND location_id = 7001
);
```

Below the query editor, the 'Results' tab is active, displaying a single row of data:

	employee_id	Fname	Lname	skill_id	location_id	designation
1	13179234	Emma	Stone	57477579	7001	Service Providers

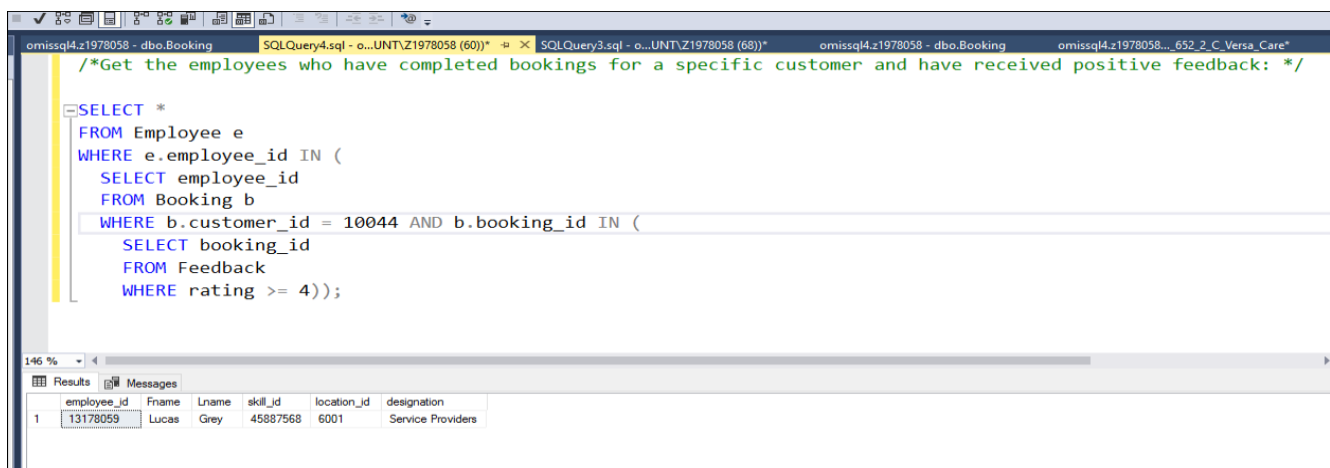
**5. Find the customers who have made bookings for a specific service category:**

```
SELECT *
FROM Customer
WHERE customer_id IN
    (SELECT customer_id
     FROM Booking
     WHERE service_id IN
        (SELECT service_id
         FROM Services
         WHERE service_category IN ('Beauty')));
```



**6. Get the employees who have completed bookings for a specific customer and have received positive feedback:**

```
SELECT sp.*
FROM Employee e
WHERE e.employee_id IN
    (SELECT employee_id
     FROM Booking b
     WHERE b.customer_id = 789 AND b.booking_id IN
        (SELECT booking_id
         FROM Feedback
         WHERE rating >= 4 ));
```





## 7. Retrieve the customers who have used the promotion code during payment:

```
SELECT DISTINCT c.customer_id, c.customer_fname, p.payment_id
FROM customer c
JOIN payments p ON c.customer_id = p.customer_id
JOIN promotions pr ON p.customer_id = c.customer_id;
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database structure, including tables like 'customer', 'payments', and 'promotions'. The right pane shows a SQL query window with the following query:

```
/* Find the average rating for each employee, i
including employees without any feedback: */

SELECT DISTINCT c.customer_id, c.customer_fname, p.payment_id
FROM customer c
JOIN payments p ON c.customer_id = p.customer_id
JOIN promotions pr ON p.customer_id = c.customer_id;
```

Below the query, the 'Results' tab shows the output of the query, which is a table with 7 rows and 3 columns: 'customer\_id', 'customer\_fname', and 'payment\_id'.

customer_id	customer_fname	payment_id
70064	Daisy	13960
10044	Bob	14043
19816	John	15017
10039	Alice	16939
18025	Jane	17824
60077	Ethan	18655
30059	Charlie	19861

## 8. Find all services and their booking history, including services that have not been booked:

```
SELECT s.service_name, b.booking_id, b.status
FROM Services s LEFT JOIN Booking b
ON s.service_id = b.service_id;
```

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database structure, including tables like 'Services' and 'Booking'. The right pane shows a SQL query window with the following query:

```
/*Find all services and their booking history, including services that have not been booked: */

SELECT s.service_name, b.booking_id, b.status
FROM Services s
LEFT JOIN Booking b ON s.service_id = b.service_id;
```

Below the query, the 'Results' tab shows the output of the query, which is a table with 7 rows and 3 columns: 'service\_name', 'booking\_id', and 'status'.

service_name	booking_id	status
Salon	12109	Confirmed
Stress Relief	26478	Pending
Appliance repair	38988	Cancelled
Smart locks installation	47245	Confirmed
Full Home Cleaning	50176	Pending
Disinfection Services	67753	Confirmed
Wall decor and house painters	79967	Cancelled

## 9. Find all equipment and their storage locations, with a specified location:

```
SELECT eq.equipment_id, eq.equipment_name, l.city, e.storage_location
FROM equipment e LEFT JOIN employee emp
ON e.employee_id = emp.employee_id
LEFT JOIN location l
ON emp.location_id = l.location_id
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure for 'z1978058', including tables like 'dbo.Equipment', 'dbo.Location', and 'dbo.Employee'. The right pane shows a SQL query window with the following query:

```
/*Find all equipment and their storage locations, including equipment without a specified location: */
SELECT
    e.equipment_id,
    e.equipment_name,
    e.storage_location,
    l.city
FROM equipment e
LEFT JOIN employee emp
ON e.employee_id = emp.employee_id
LEFT JOIN location l
ON emp.location_id = l.location_id
```

Below the query, the 'Results' tab shows the output of the query:

equipment_id	equipment_name	storage_location	city
15787	High-grade cleaning agents	Room 101	Chicago
24565	Drilling machine	Room 202	New York City
37756	Hair Dryer	Room 303	Boston
47567	Vacuum cleaners	Room 101	Jersey City
57549	Hammers & screwdrivers	Room 202	Los Angeles
65776	Manicure Set	Room 303	San Francisco
74656	painter's tape	Room 101	Austin

## 10. Retrieve the number of services offered at each location, including locations without any services:

```
SELECT l.location_id, l.city, COUNT(s.service_id) AS service_count
FROM Location l LEFT JOIN Services s
ON l.location_id = s.location_id
GROUP BY l.location_id, l.city;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure for 'z1978058', including tables like 'dbo.Location' and 'dbo.Services'. The right pane shows a SQL query window with the following query:

```
/*Retrieve the number of services offered at each location, including locations without any services(if any):*/
SELECT l.location_id,
    l.city,
    COUNT(s.service_id) AS service_count
FROM Location l
LEFT JOIN Services s ON l.location_id = s.location_id
GROUP BY l.location_id, l.city;
```

Below the query, the 'Results' tab shows the output of the query:

location_id	city	service_count
4001	Chicago	1
5001	New York City	1
5003	Boston	1
6001	Jersey City	1
7001	Los Angeles	1
8001	San Francisco	1
8005	Austin	1

## 11. Find the total number of bookings for each employee, including providers without any bookings:

```
SELECT e.employee_id, e.Fname, e.Lname, COUNT(b.booking_id) AS booking_count
FROM Employee e LEFT JOIN Booking b
ON e.employee_id = b.employee_id
GROUP BY e.employee_id, e.Fname, e.Lname;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database schema for 'z1978058', including tables like Employee, Booking, and various support tables. The right pane shows a SQL query window with the following query:

```
/*Find the total number of bookings for each employee,
including providers without any bookings: */
SELECT e.employee_id,
       e.Fname, e.Lname,
       COUNT(b.booking_id) AS booking_count
FROM Employee e
LEFT JOIN Booking b ON e.employee_id = b.employee_id
GROUP BY e.employee_id, e.Fname, e.Lname;
```

Below the query window, the 'Results' tab displays the output of the query:

employee_id	Fname	Lname	booking_count
13172228	Sarah	Connor	1
13177065	James	Smith	1
13178048	Rachel	Adams	1
13178059	Lucas	Grey	2
13179234	Emma	Stone	1
13183439	Liam	Neeson	1
13197268	Olivia	Wilde	1

## 12. Get the total amount of payments made by each customer, including customers who have not made any payments:

```
SELECT c.customer_id, c.customer_Fname, c.customer_Lname, SUM(p.amount) AS
total_payment
FROM Customers c LEFT JOIN Payments p
ON c.customer_id = p.customer_id
GROUP BY c.customer_id, c.customer_Fname, c.customer_Lname;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database schema for 'z1978058'. The right pane shows a SQL query window with the following query:

```
/* Get the total amount of payments made by each customer,,
including customers who have not made any payments: */
SELECT c.customer_id, c.customer_Fname,
       c.customer_Lname,
       SUM(p.amount) AS total_payment
FROM Customer c
LEFT JOIN Payments p ON c.customer_id = p.customer_id
GROUP BY c.customer_id, c.customer_Fname, c.customer_Lname;
```

Below the query window, the 'Results' tab displays the output of the query:

customer_id	customer_Fname	customer_Lname	total_payment
10039	Alice	Johnson	150
10044	Bob	Brown	250
18025	Jane	Smith	200
19816	John	Doe	100
30059	Charlie	Davis	300
60077	Ethan	Ford	400
70064	Daisy	Evans	350

### 13. Find the average rating for each employee, including employees without any feedback:

```
SELECT e.employee_id, e.Fname, e.Lname, AVG(f.rating) AS average_rating
FROM Employee e LEFT JOIN Feedback f
ON e.employee_id = f.employee_id
GROUP BY e.employee_id, e.Fname, e.Lname;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'dbo' schema expanded, showing various tables. The right pane shows a SQL query window with the following query:

```
/* Find the average rating for each employee, i
including employees without any feedback: */

SELECT e.employee_id,
       e.Fname, e.Lname,
       AVG(f.rating) AS average_rating
FROM Employee e
LEFT JOIN Feedback f ON e.employee_id = f.employee_id
GROUP BY e.employee_id, e.Fname, e.Lname;
```

The 'Results' tab shows the output of the query:

	employee_id	Fname	Lname	average_rating
1	13172728	Sarah	Connor	5
2	13177065	James	Smith	4
3	13178048	Rachel	Adams	3
4	13178059	Lucas	Grey	5
5	13179234	Emma	Stone	4
6	13183439	Liam	Neeson	2
7	13197268	Olivia	Wilde	5

### 14. Customers who have made payments but have not provided feedback:

```
SELECT c.customer_id, c.customer_Fname, c.customer_lname
FROM Customers c
WHERE c.customer_id IN
      (SELECT customer_id
       FROM Payments EXCEPT
       SELECT customer_id
       FROM Feedback
       WHERE feedback_id IS NOT NULL);
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'dbo' schema expanded, showing various tables. The right pane shows a SQL query window with the following query:

```
/* Customers who have made payments but
have not provided feedback: */

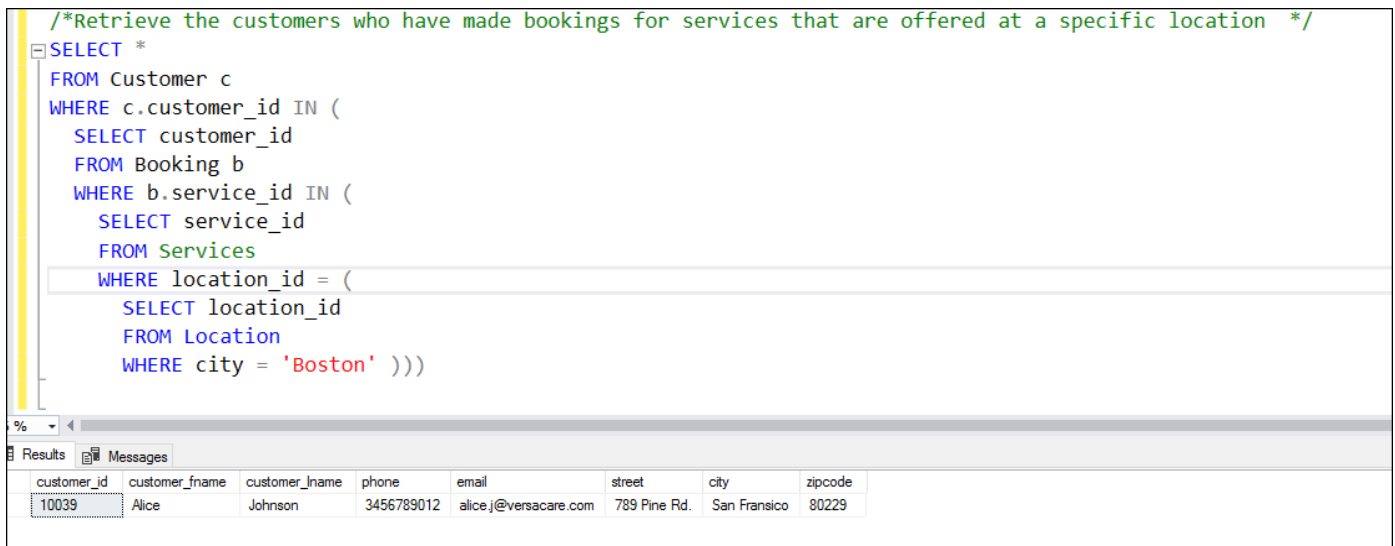
SELECT c.customer_id, c.customer_Fname, c.customer_lname
FROM Customer c
WHERE c.customer_id IN
      ( SELECT customer_id
        FROM Payments EXCEPT
        SELECT customer_id
        FROM Feedback
        WHERE feedback_id IS NOT NULL );
```

The 'Results' tab shows the output of the query:

	customer_id	customer_Fname	customer_lname
1	60077	Ethan	Ford

**15. Retrieve the customers who have made bookings for services that are offered at a specific location:**

```
SELECT *
FROM Customer c
WHERE c.customer_id IN
      (SELECT customer_id
       FROM Booking b
       WHERE b.service_id IN
            (SELECT service_id
             FROM Services
             WHERE location_id =
                  (SELECT location_id
                   FROM Location
                   WHERE city = 'Boston' )))
```



The screenshot shows a SQL IDE with a query editor and a results pane. The query in the editor is the same as the one above. The results pane shows a table with 8 columns: customer\_id, customer\_fname, customer\_lname, phone, email, street, city, and zipcode. The first row of data is highlighted.

customer_id	customer_fname	customer_lname	phone	email	street	city	zipcode
10039	Alice	Johnson	3456789012	alice.j@versacare.com	789 Pine Rd.	San Fransico	80229

**16. Find the top 5 service providers with the highest number of completed bookings at first:**

```
SELECT sp.employee_id, sp.Fname, sp.Lname, COUNT(b.booking_id) AS total_bookings
FROM employee sp INNER JOIN Booking b
ON sp.employee_id = b.employee_id
WHERE b.status = 'Confirmed'
GROUP BY sp.employee_id, sp.Fname, sp.Lname
ORDER BY total_bookings DESC;
```

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'z1978058'. The main window shows a SQL query titled 'omissql4.z1978058 - dbo.Employee' with the following text:

```

/* Find the top 5 service providers with
the highest number of completed bookings at first: */

SELECT sp.employee_id,
       sp.Fname, sp.Lname,
       COUNT(b.booking_id) AS total_bookings
FROM employee sp
INNER JOIN Booking b ON sp.employee_id = b.employee_id
WHERE b.status = 'Confirmed'
GROUP BY sp.employee_id, sp.Fname, sp.Lname
ORDER BY total_bookings DESC;

```

The Results pane at the bottom shows the following data:

employee_id	Fname	Lname	total_bookings
13178059	Lucas	Grey	2
13183439	Liam	Neeson	1
13172728	Sarah	Connor	1

## 17. Retrieve the services with the highest average rating from customer feedback:

```

SELECT s.service_id, s.service_name, AVG(f.rating) AS average_rating
FROM Services s
INNER JOIN Booking b ON s.service_id = b.service_id
INNER JOIN Feedback f ON b.booking_id = f.booking_id
GROUP BY s.service_id, s.service_name
ORDER BY average_rating DESC;

```

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure for 'z1978058'. The main window shows a SQL query titled 'omissql4.z1978058 - dbo.Employee' with the following text:

```

/*Retrieve the services with the
highest average rating from customer feedback: */

SELECT s.service_id, s.service_name, AVG(f.rating) AS average_rating
FROM Services s
INNER JOIN Booking b ON s.service_id = b.service_id
INNER JOIN Feedback f ON b.booking_id = f.booking_id
GROUP BY s.service_id, s.service_name
ORDER BY average_rating DESC;

```

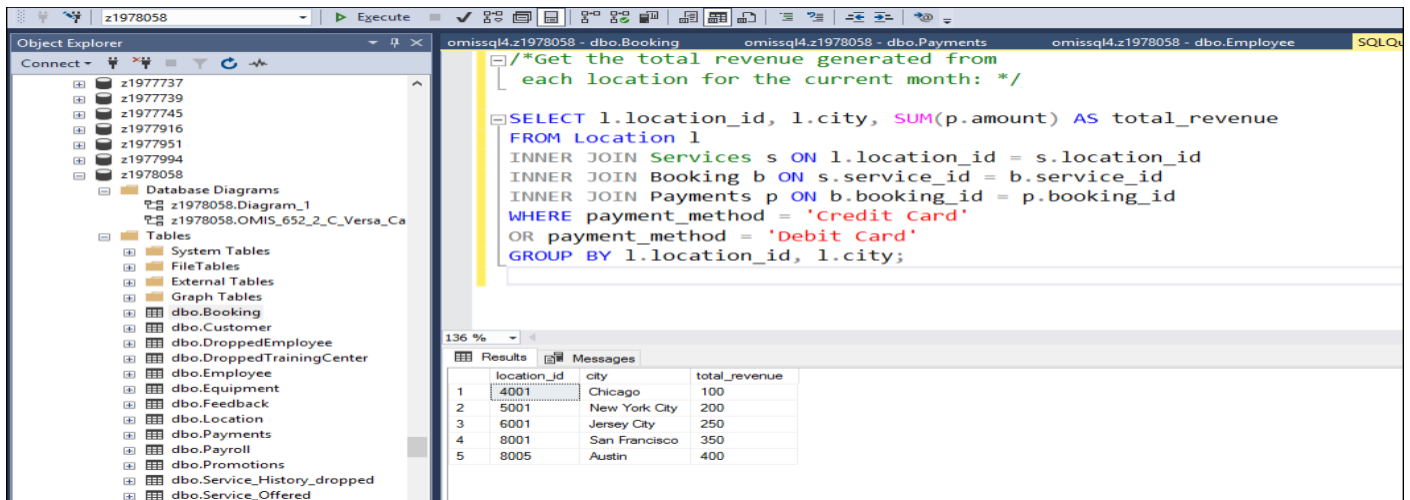
The Results pane at the bottom shows the following data:

service_id	service_name	average_rating
100126	Saloon	5
129983	Smart locks installation	5
190142	Wall decor and house painters	5
145016	Full Home Cleaning	4
100458	Stress Relief	4
112879	Appliance repair	3
167230	Disinfection Services	2



## 18. Get the total revenue generated from each location from credit card and debit card:

```
SELECT l.location_id, l.city, SUM(p.amount) AS total_revenue
FROM Location l
INNER JOIN Services s ON l.location_id = s.location_id
INNER JOIN Booking b ON s.service_id = b.service_id
INNER JOIN Payments p ON b.booking_id = p.booking_id
WHERE payment_method = 'Credit Card' OR payment_method = 'Debit Card'
GROUP BY l.location_id, l.city;
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database 'z1978058'. The right pane shows a query window with the following SQL code:

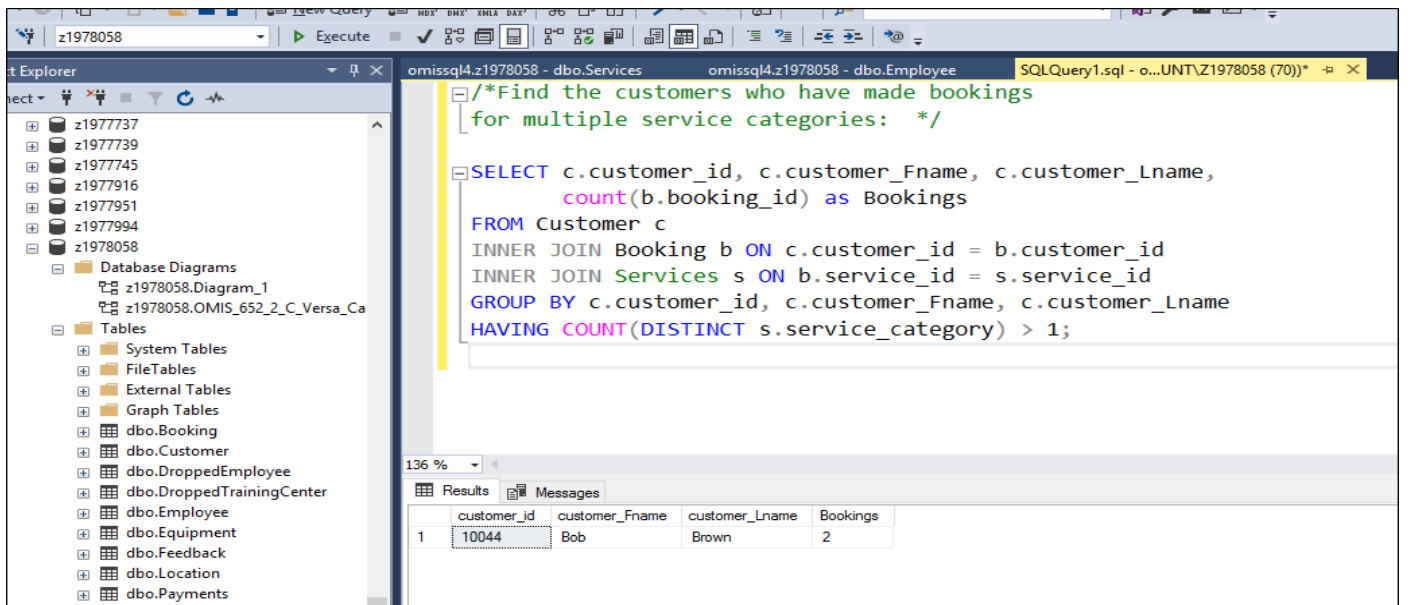
```
/*Get the total revenue generated from
each location for the current month: */
SELECT l.location_id, l.city, SUM(p.amount) AS total_revenue
FROM Location l
INNER JOIN Services s ON l.location_id = s.location_id
INNER JOIN Booking b ON s.service_id = b.service_id
INNER JOIN Payments p ON b.booking_id = p.booking_id
WHERE payment_method = 'Credit Card'
OR payment_method = 'Debit Card'
GROUP BY l.location_id, l.city;
```

Below the query window, the 'Results' pane shows the output of the query:

location_id	city	total_revenue
4001	Chicago	100
5001	New York City	200
6001	Jersey City	250
8001	San Francisco	350
8005	Austin	400

## 19. Find the customers who have made bookings for multiple service categories:

```
SELECT c.customer_id, c.customer_Fname, c.customer_Lname, count(b.booking_id) as
Bookings
FROM Customer c
INNER JOIN Booking b ON c.customer_id = b.customer_id
INNER JOIN Services s ON b.service_id = s.service_id
GROUP BY c.customer_id, c.customer_Fname, c.customer_Lname
HAVING COUNT(DISTINCT s.service_category) > 1;
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database 'z1978058'. The right pane shows a query window with the following SQL code:

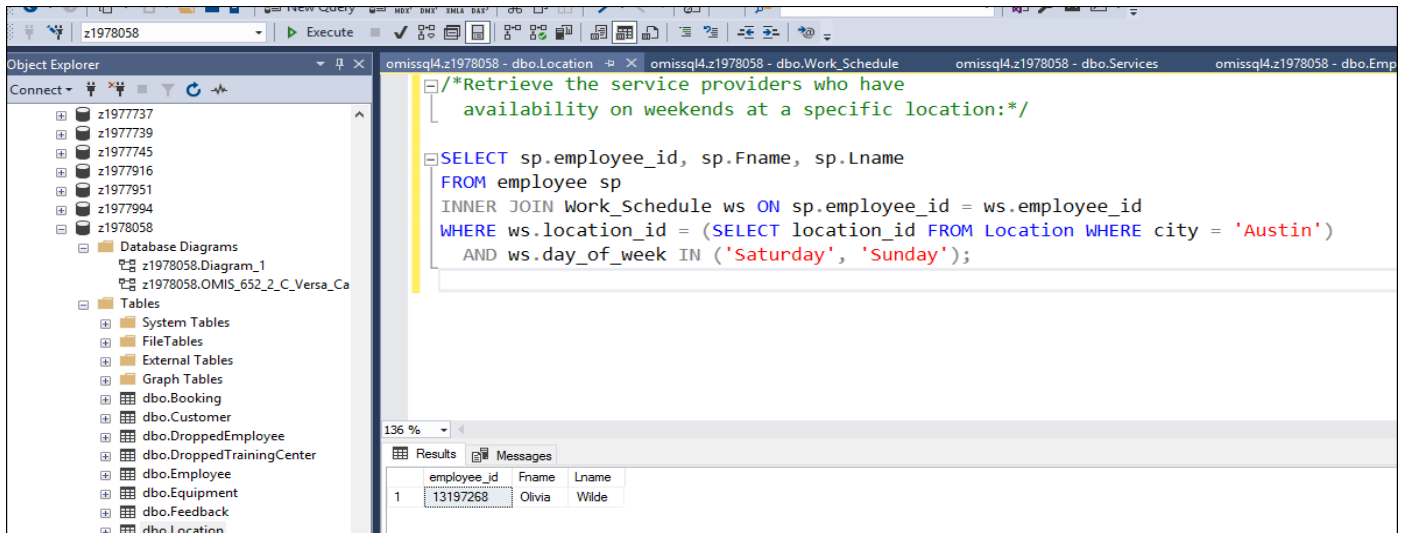
```
/*Find the customers who have made bookings
for multiple service categories: */
SELECT c.customer_id, c.customer_Fname, c.customer_Lname,
count(b.booking_id) as Bookings
FROM Customer c
INNER JOIN Booking b ON c.customer_id = b.customer_id
INNER JOIN Services s ON b.service_id = s.service_id
GROUP BY c.customer_id, c.customer_Fname, c.customer_Lname
HAVING COUNT(DISTINCT s.service_category) > 1;
```

Below the query window, the 'Results' pane shows the output of the query:

customer_id	customer_Fname	customer_Lname	Bookings
10044	Bob	Brown	2

## 20. Retrieve the service providers who have availability on weekends at a specific location:

```
SELECT sp.employee_id, sp.Fname, sp.Lname
FROM employee sp
INNER JOIN Work_Schedule ws ON sp.employee_id = ws.employee_id
WHERE ws.location_id = (SELECT location_id FROM Location WHERE city = 'Austin')
AND ws.day_of_week IN ('Saturday', 'Sunday');
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database 'z1978058'. The right pane shows a query window with the following SQL code:

```
/*Retrieve the service providers who have
availability on weekends at a specific location:*/

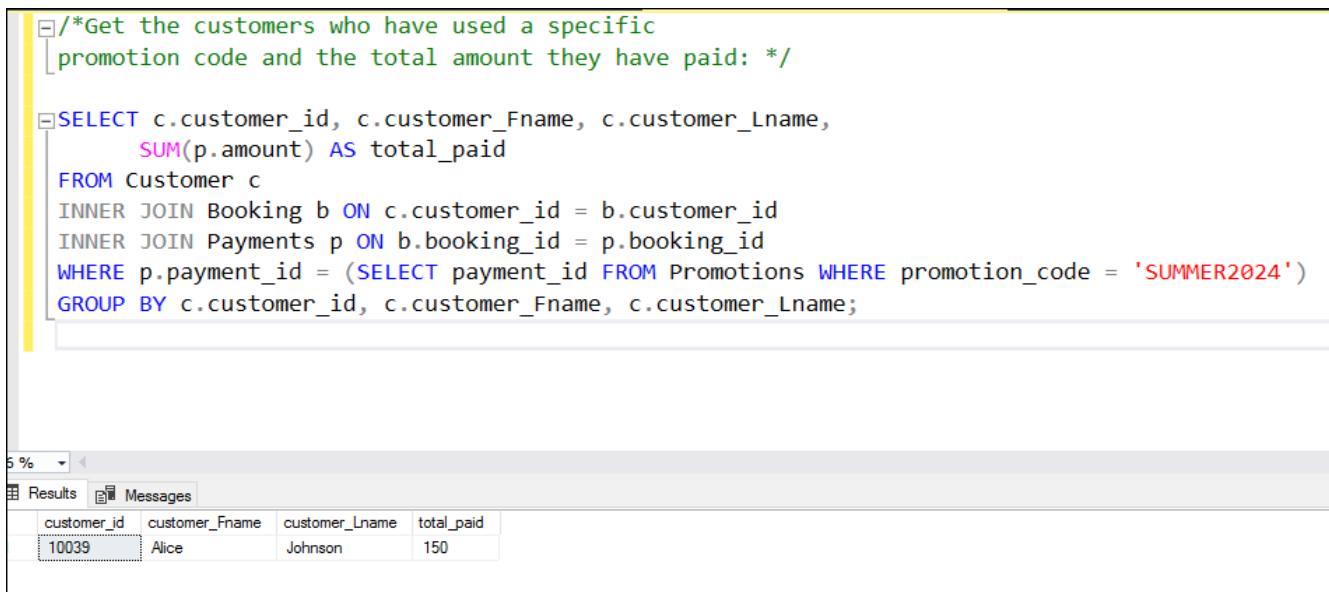
SELECT sp.employee_id, sp.Fname, sp.Lname
FROM employee sp
INNER JOIN Work_Schedule ws ON sp.employee_id = ws.employee_id
WHERE ws.location_id = (SELECT location_id FROM Location WHERE city = 'Austin')
AND ws.day_of_week IN ('Saturday', 'Sunday');
```

Below the query window, the 'Results' tab is active, displaying a single row of data:

employee_id	Fname	Lname
13197268	Olivia	Wilde

## 21. Get the customers who have used a specific promotion code and the total amount they have paid:

```
SELECT c.customer_id, c.customer_Fname, c.customer_Lname, SUM(p.amount) as total_paid
FROM Customer c
INNER JOIN Booking b ON c.customer_id = b.customer_id
INNER JOIN Payments p ON b.booking_id = p.booking_id
WHERE p.payment_id = (SELECT payment_id FROM Promotions WHERE promotion_code =
'SUMMER2024')
GROUP BY c.customer_id, c.customer_Fname, c.customer_Lname;
```



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with a tree view of the database 'z1978058'. The right pane shows a query window with the following SQL code:

```
/*Get the customers who have used a specific
promotion code and the total amount they have paid: */

SELECT c.customer_id, c.customer_Fname, c.customer_Lname,
SUM(p.amount) AS total_paid
FROM Customer c
INNER JOIN Booking b ON c.customer_id = b.customer_id
INNER JOIN Payments p ON b.booking_id = p.booking_id
WHERE p.payment_id = (SELECT payment_id FROM Promotions WHERE promotion_code = 'SUMMER2024')
GROUP BY c.customer_id, c.customer_Fname, c.customer_Lname;
```

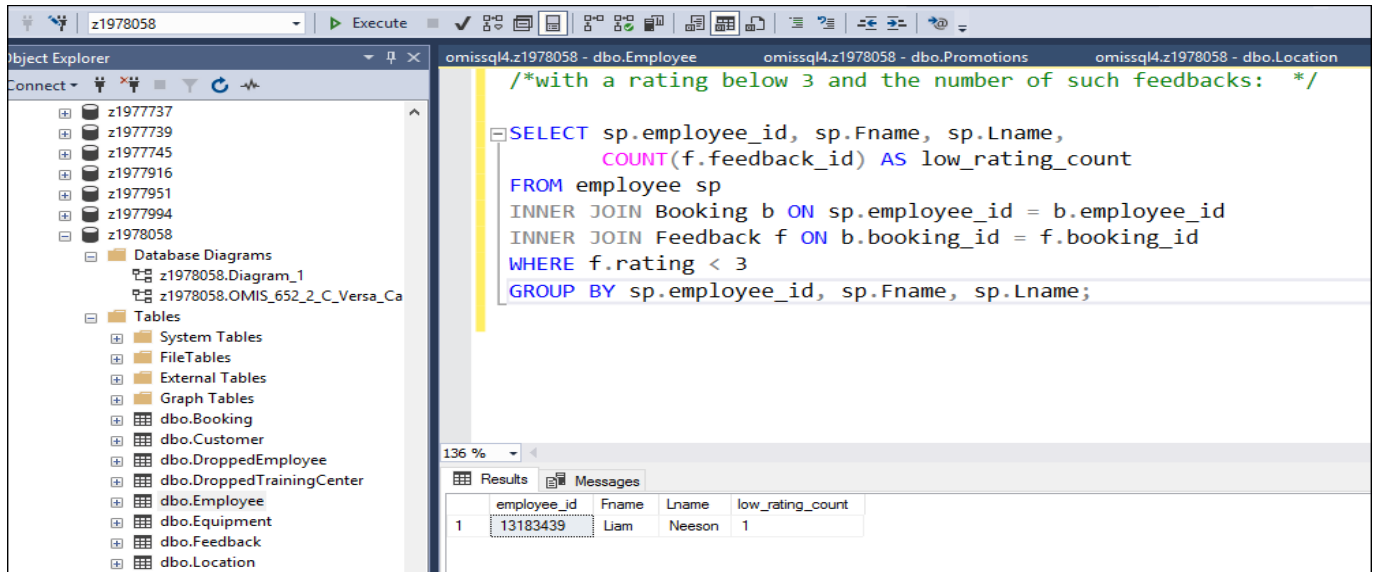
Below the query window, the 'Results' tab is active, displaying a single row of data:

customer_id	customer_Fname	customer_Lname	total_paid
10039	Alice	Johnson	150



**22. Find the service providers who have received feedback with a rating below 3 and the number of such feedbacks:**

```
SELECT sp.employee_id, sp.Fname, sp.Lname, COUNT(f.feedback_id) as low_rating_count
FROM employee sp
INNER JOIN Booking b ON sp.employee_id = b.employee_id
INNER JOIN Feedback f ON b.booking_id = f.booking_id
WHERE f.rating < 3
GROUP BY sp.employee_id, sp.Fname, sp.Lname;
```



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'z1978058', including tables like 'dbo.Employee', 'dbo.Feedback', and 'dbo.Location'. The main pane shows a SQL query window with the following query:

```
/*with a rating below 3 and the number of such feedbacks: */
SELECT sp.employee_id, sp.Fname, sp.Lname,
COUNT(f.feedback_id) AS low_rating_count
FROM employee sp
INNER JOIN Booking b ON sp.employee_id = b.employee_id
INNER JOIN Feedback f ON b.booking_id = f.booking_id
WHERE f.rating < 3
GROUP BY sp.employee_id, sp.Fname, sp.Lname;
```

Below the query window, the Results tab shows the output of the query:

	employee_id	Fname	Lname	low_rating_count
1	13183439	Liam	Neeson	1