Business Problem

Company Details

In the heart of Lisbon, the boutique hotel "**BoutiqueDeLisboa**" was weaving its story of sophistication and charm. As the demand for its unique blend of Portuguese elegance and modern luxury soared, the proprietors found themselves facing the delightful challenge of managing a growing business with three distinct buildings. With a total of 30 rooms spread across these architectural gems, the need for a well-organized and modern data management system became apparent.

Why Does the Company Need DBMS?

The visionary minds behind BoutiqueDeLisboa recognized the pivotal role that technology could play in elevating the guest experience and optimizing internal operations. In pursuit of this, company decided to embark on a journey of digital transformation, beginning with the creation of an Entity-Relationship Diagram (ERD) tailored to the boutique hotel's unique needs. This ERD would serve as the architectural blueprint for a sophisticated data management system, designed to seamlessly handle the intricacies of their multi-building business and ensure a harmonious experience for their esteemed guests.

What DBMS Brings to the Table

Streamlined Customer Information

Efficient Reservation System

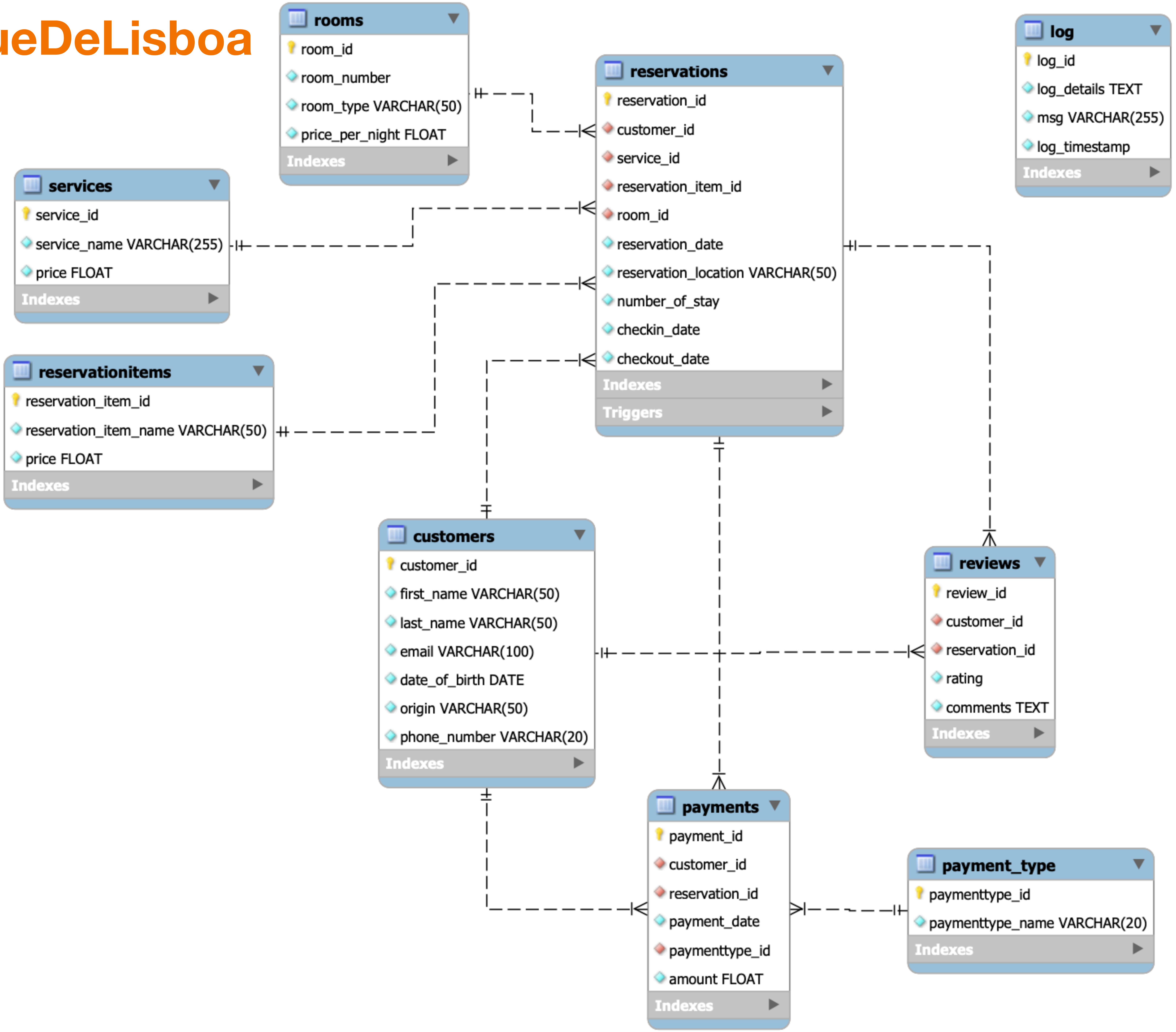
Variable Tracking for Services

Financial Management and Reporting

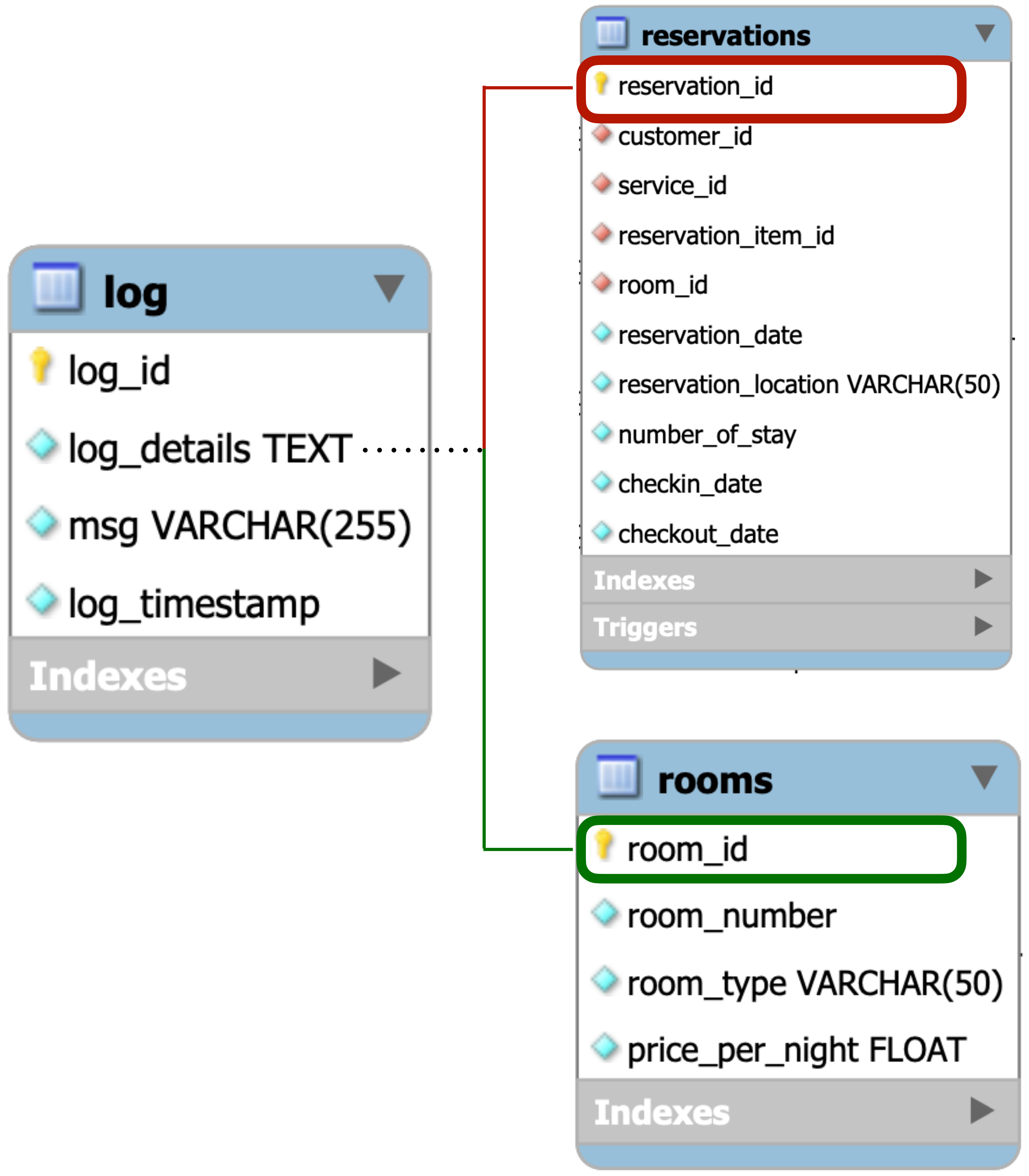
Dynamic Room Assignments

Real-time Feedback Analysis

ERD of **BoutiqueDeLisboa**



Triggers(Log)

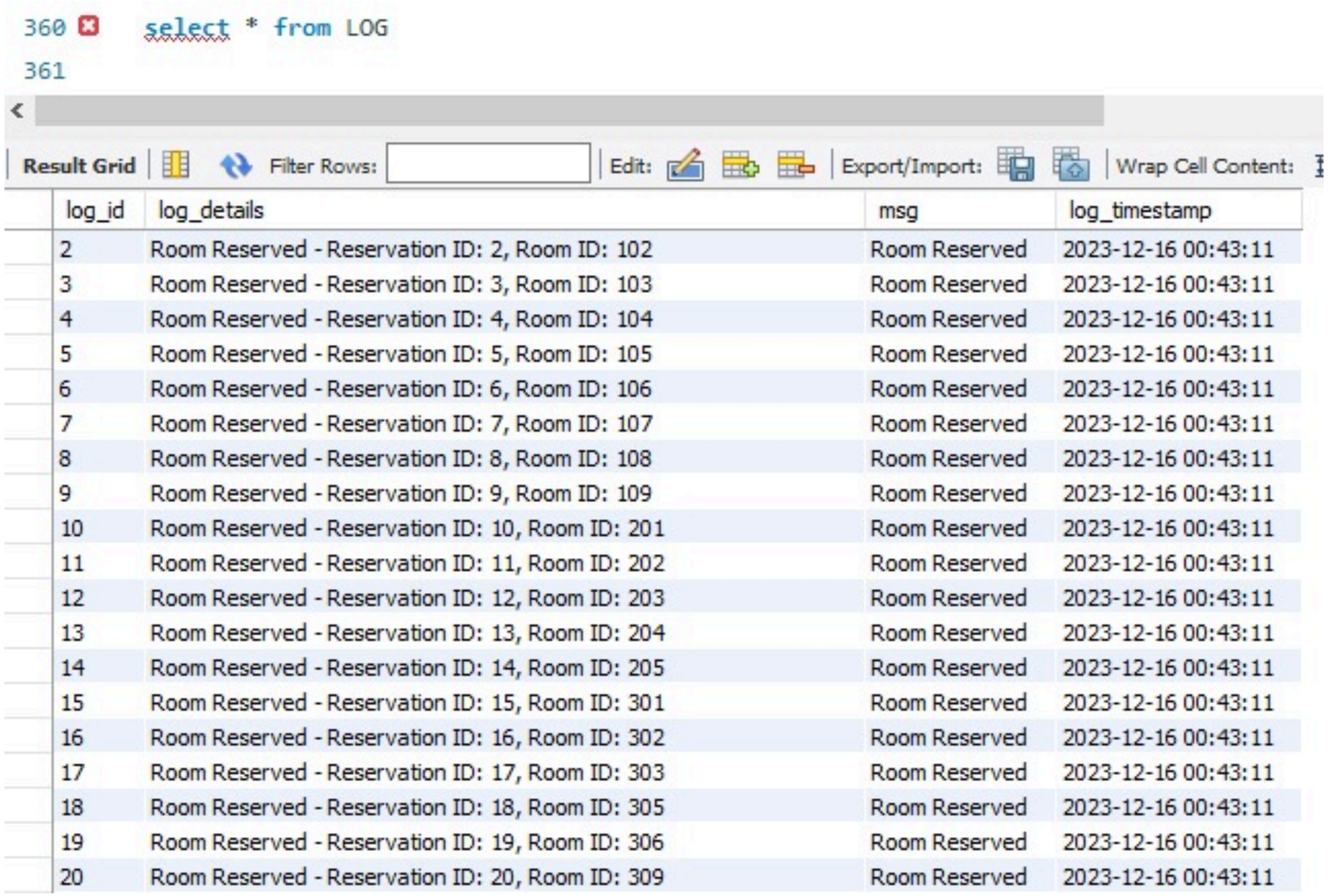


```
DELIMITER //
CREATE TRIGGER tr_room_reservation_log
AFTER INSERT ON RESERVATIONS
FOR EACH ROW
BEGIN
-- Log the room reservation details in the LOG table
INSERT INTO LOG (log_details, msg,
log_timestamp)
VALUES (
CONCAT('Room Reserved - Reservation ID: ',
NEW.reservation_id, ', Room ID: ', NEW.room_id),
'Room Reserved',
NOW()
);
END;
//
DELIMITER ;
```

Example Output

log_id	log_details	msg	log_timestamp
1	Room Reserved - Reservation ID: 1, Room ID: 101	Room Reserved	2023-12-15 12:34:56
2	Room Reserved - Reservation ID: 2, Room ID: 102	Room Reserved	2023-12-15 12:36:00
3	Room Reserved - Reservation ID: 4, Room ID: 104	Room Reserved	Current Timestamp

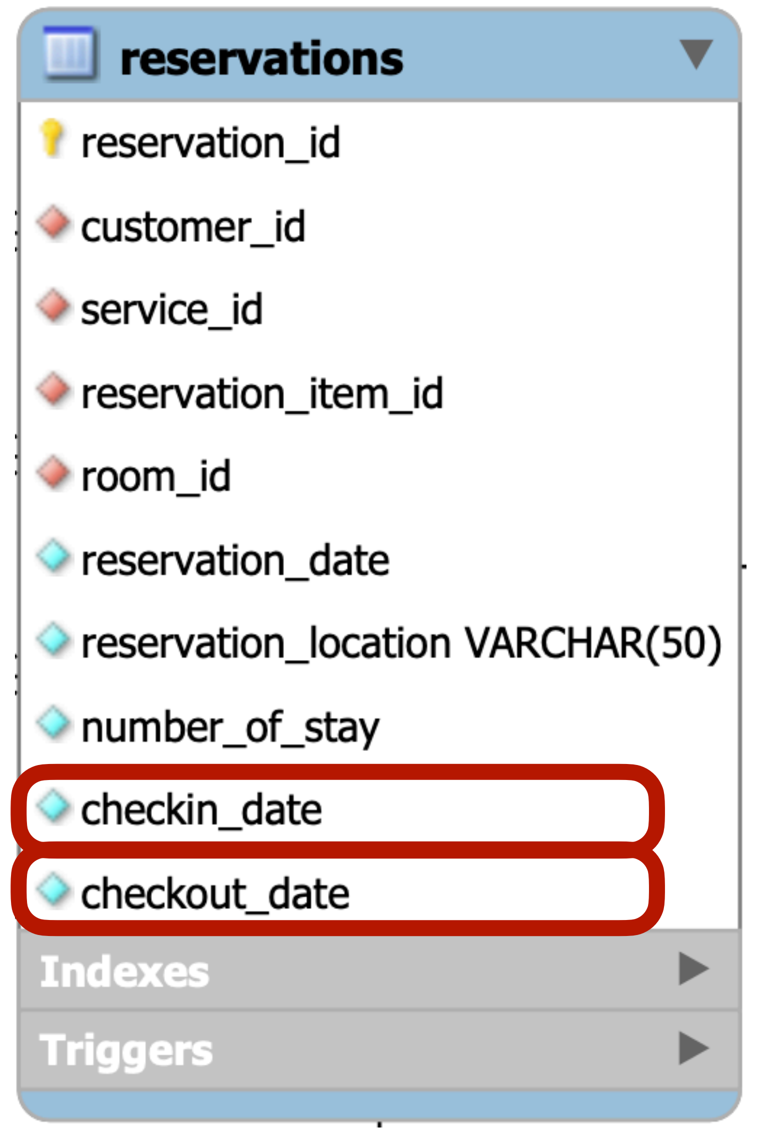
Triggers(Log)



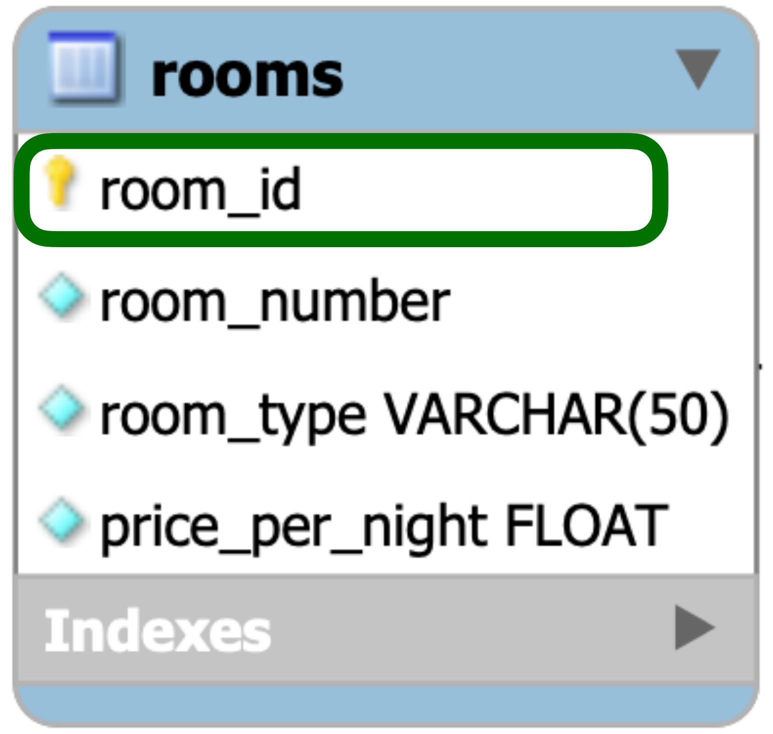
```
select * from LOG
```

log_id	log_details	msg	log_timestamp
2	Room Reserved - Reservation ID: 2, Room ID: 102	Room Reserved	2023-12-16 00:43:11
3	Room Reserved - Reservation ID: 3, Room ID: 103	Room Reserved	2023-12-16 00:43:11
4	Room Reserved - Reservation ID: 4, Room ID: 104	Room Reserved	2023-12-16 00:43:11
5	Room Reserved - Reservation ID: 5, Room ID: 105	Room Reserved	2023-12-16 00:43:11
6	Room Reserved - Reservation ID: 6, Room ID: 106	Room Reserved	2023-12-16 00:43:11
7	Room Reserved - Reservation ID: 7, Room ID: 107	Room Reserved	2023-12-16 00:43:11
8	Room Reserved - Reservation ID: 8, Room ID: 108	Room Reserved	2023-12-16 00:43:11
9	Room Reserved - Reservation ID: 9, Room ID: 109	Room Reserved	2023-12-16 00:43:11
10	Room Reserved - Reservation ID: 10, Room ID: 201	Room Reserved	2023-12-16 00:43:11
11	Room Reserved - Reservation ID: 11, Room ID: 202	Room Reserved	2023-12-16 00:43:11
12	Room Reserved - Reservation ID: 12, Room ID: 203	Room Reserved	2023-12-16 00:43:11
13	Room Reserved - Reservation ID: 13, Room ID: 204	Room Reserved	2023-12-16 00:43:11
14	Room Reserved - Reservation ID: 14, Room ID: 205	Room Reserved	2023-12-16 00:43:11
15	Room Reserved - Reservation ID: 15, Room ID: 301	Room Reserved	2023-12-16 00:43:11
16	Room Reserved - Reservation ID: 16, Room ID: 302	Room Reserved	2023-12-16 00:43:11
17	Room Reserved - Reservation ID: 17, Room ID: 303	Room Reserved	2023-12-16 00:43:11
18	Room Reserved - Reservation ID: 18, Room ID: 305	Room Reserved	2023-12-16 00:43:11
19	Room Reserved - Reservation ID: 19, Room ID: 306	Room Reserved	2023-12-16 00:43:11
20	Room Reserved - Reservation ID: 20, Room ID: 309	Room Reserved	2023-12-16 00:43:11

Triggers(Availability)



reservations
reservation_id
customer_id
service_id
reservation_item_id
room_id
reservation_date
reservation_location VARCHAR(50)
number_of_stay
checkin_date
checkout_date

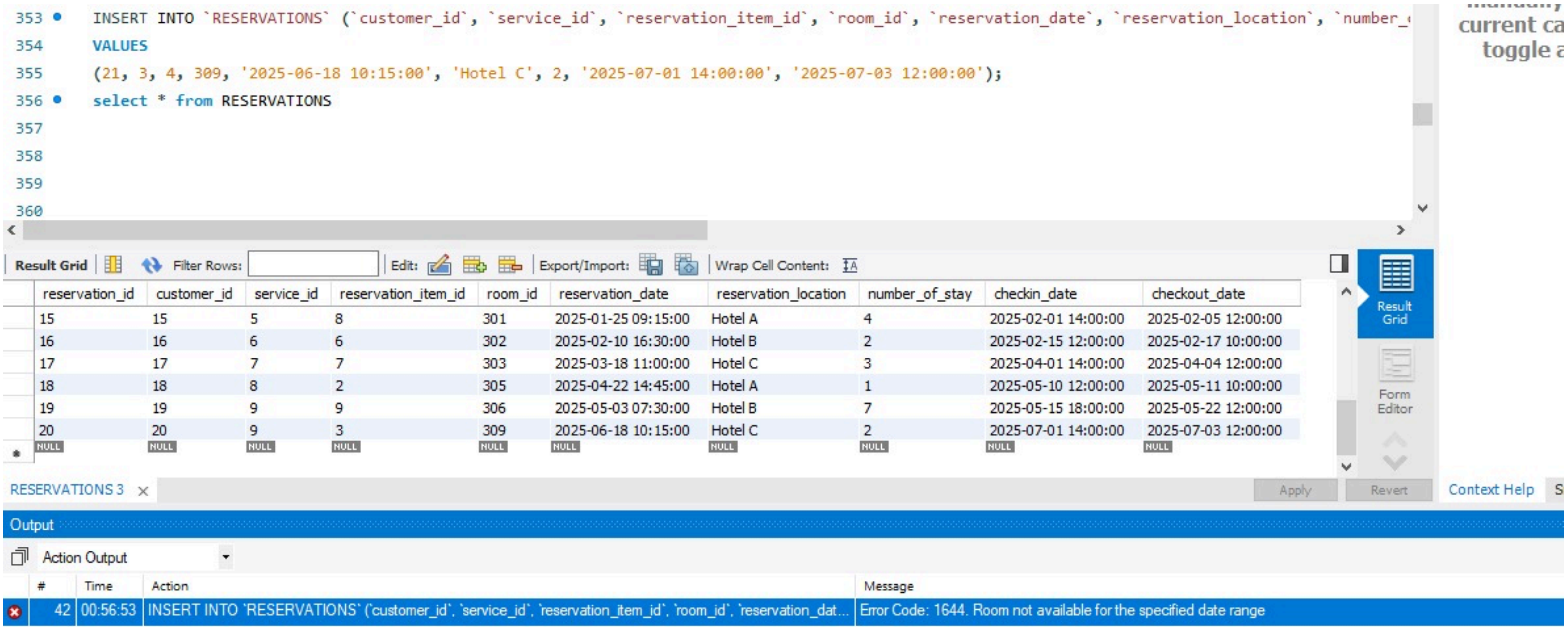


rooms
room_id
room_number
room_type VARCHAR(50)
price_per_night FLOAT

If a room is reserved between specific dates. When a customer wants to reserve it gives an error message

```
DELIMITER //
CREATE TRIGGER check_room_availability
BEFORE INSERT ON RESERVATIONS
FOR EACH ROW
BEGIN
DECLARE room_count INT;
-- Check if the room is available for the specified date range
SELECT COUNT(*) INTO room_count
FROM RESERVATIONS
WHERE room_id = NEW.room_id
AND ((NEW.checkin_date >= checkin_date AND NEW.checkin_date < checkout_date)
OR (NEW.checkout_date > checkin_date AND NEW.checkout_date <= checkout_date)
OR (NEW.checkin_date <= checkin_date AND NEW.checkout_date >= checkout_date));
-- If room_count is greater than 0, then the room is not available
IF room_count > 0 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Room not available for the specified date range';
END IF;
END
//
DELIMITER ;
```

Triggers(Availability)



```
INSERT INTO `RESERVATIONS` (`customer_id`, `service_id`, `reservation_item_id`, `room_id`, `reservation_date`, `reservation_location`, `number_
VALUES
(21, 3, 4, 309, '2025-06-18 10:15:00', 'Hotel C', 2, '2025-07-01 14:00:00', '2025-07-03 12:00:00');
select * from RESERVATIONS
```

current ca
toggle a

reservation_id	customer_id	service_id	reservation_item_id	room_id	reservation_date	reservation_location	number_of_stay	checkin_date	checkout_date
15	15	5	8	301	2025-01-25 09:15:00	Hotel A	4	2025-02-01 14:00:00	2025-02-05 12:00:00
16	16	6	6	302	2025-02-10 16:30:00	Hotel B	2	2025-02-15 12:00:00	2025-02-17 10:00:00
17	17	7	7	303	2025-03-18 11:00:00	Hotel C	3	2025-04-01 14:00:00	2025-04-04 12:00:00
18	18	8	2	305	2025-04-22 14:45:00	Hotel A	1	2025-05-10 12:00:00	2025-05-11 10:00:00
19	19	9	9	306	2025-05-03 07:30:00	Hotel B	7	2025-05-15 18:00:00	2025-05-22 12:00:00
20	20	9	3	309	2025-06-18 10:15:00	Hotel C	2	2025-07-01 14:00:00	2025-07-03 12:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

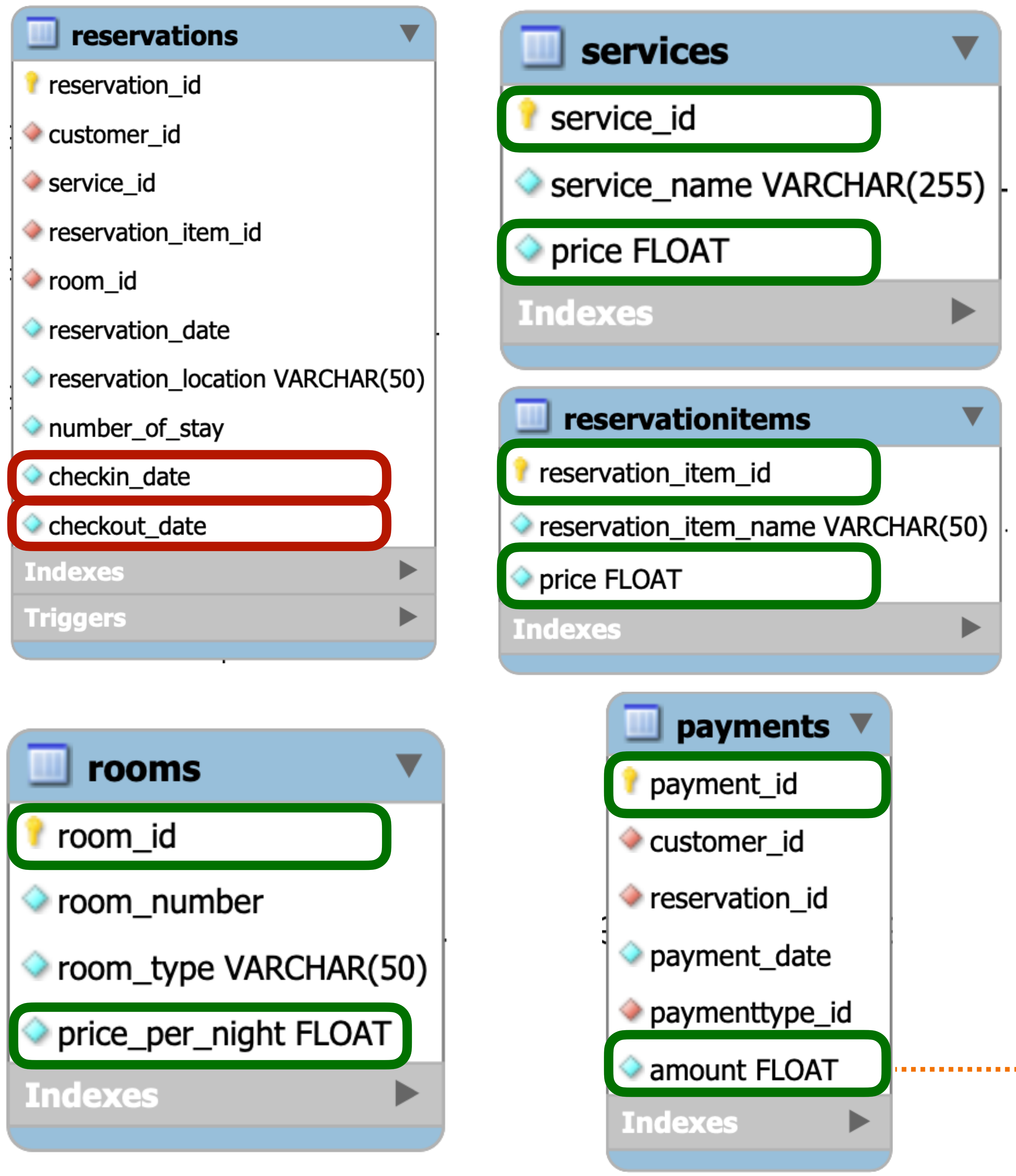
RESERVATIONS 3

Output

Action Output

#	Time	Action	Message
42	00:56:53	INSERT INTO `RESERVATIONS` (`customer_id`, `service_id`, `reservation_item_id`, `room_id`, `reservation_dat...	Error Code: 1644. Room not available for the specified date range

Triggers(Total Price)



reservations
- reservation_id
- customer_id
- service_id
- reservation_item_id
- room_id
- reservation_date
- reservation_location VARCHAR(50)
- number_of_stay
- checkin_date
- checkout_date
- Indexes
- Triggers

services
- service_id
- service_name VARCHAR(255)
- price FLOAT
- Indexes

reservationitems
- reservation_item_id
- reservation_item_name VARCHAR(50)
- price FLOAT
- Indexes

rooms
- room_id
- room_number
- room_type VARCHAR(50)
- price_per_night FLOAT
- Indexes

payments
- payment_id
- customer_id
- reservation_id
- payment_date
- paymenttype_id
- amount FLOAT
- Indexes

In our payment table, the total amount is calculated automatically;

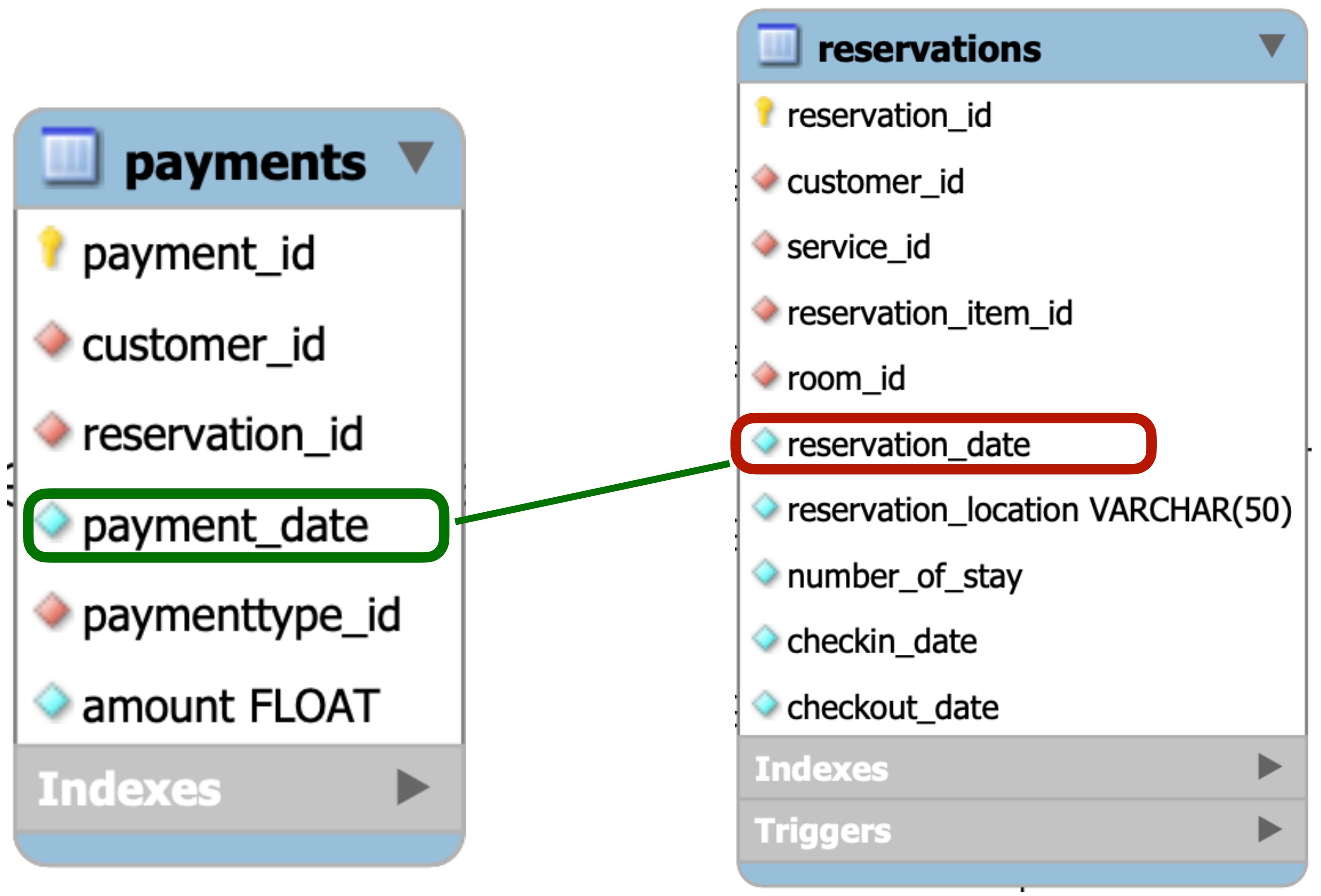
Room Price * Number of Stays
Services Price * Number of Stays
+ Reservation Item Price * Number of StayStays

Total Price

And the total price automatically imputing to the payment's table amount column.

```
CREATE TRIGGER calculate_amount_trigger BEFORE INSERT ON PAYMENTS
FOR EACH ROW
BEGIN
DECLARE room_price FLOAT;
DECLARE reservation_item_price FLOAT;
DECLARE service_price FLOAT;
DECLARE nights_stayed INT;
-- Fetch reservation details for the payment
SELECT
r.room_id,
r.reservation_item_id,
r.service_id,
r.number_of_stay
INTO
room_price,
reservation_item_price,
service_price,
nights_stayed
FROM RESERVATIONS r
WHERE r.reservation_id = NEW.reservation_id;
-- Calculate total amount based on the number of nights stayed and prices
SET NEW.amount = nights_stayed * (
(SELECT price_per_night FROM ROOMS WHERE room_id = room_price) +
(SELECT price FROM RESERVATIONITEMS WHERE reservation_item_id = reservation_item_price) +
(SELECT price FROM SERVICES WHERE service_id = service_price)
);
END;
//
```

Triggers(Auto Match)

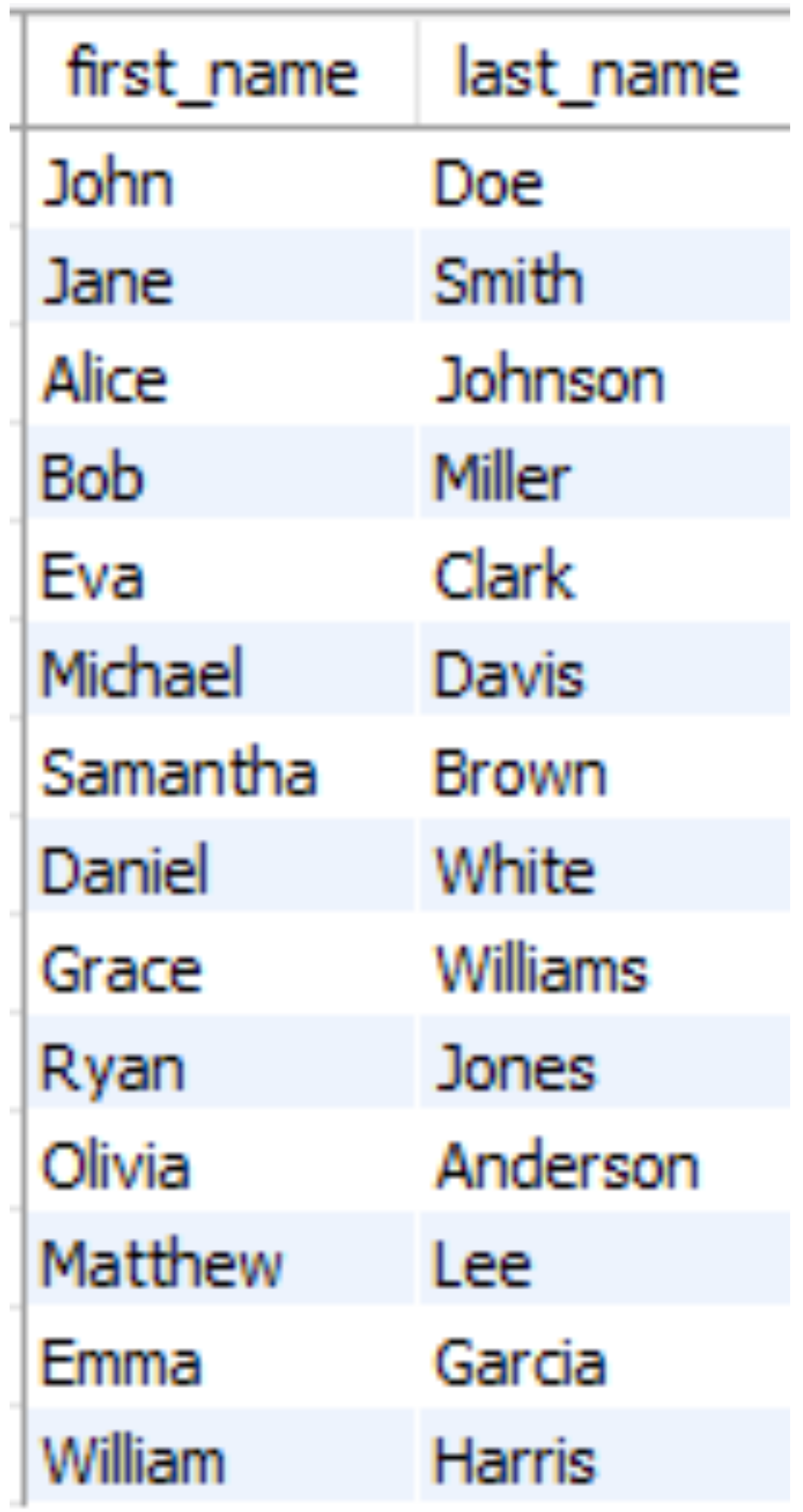


Once the reservation is done, payment_date is automatically filled with reservation date

```
DELIMITER //
CREATE TRIGGER ensure_matching_dates_trigger BEFORE INSERT ON PAYMENTS
FOR EACH ROW
BEGIN
-- Fetch reservation_date for the given reservation_id
DECLARE reservation_date_check DATETIME;
SELECT reservation_date INTO reservation_date_check
FROM RESERVATIONS
WHERE reservation_id = NEW.reservation_id;
-- Check if reservation_date matches payment_date or set payment_date to reservation_date
IF NEW.payment_date IS NULL THEN
SET NEW.payment_date = reservation_date_check;
ELSE
IF NEW.payment_date <> reservation_date_check THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Reservation date must match payment date';
END IF;
END IF;
END;
//
DELIMITER ;
```

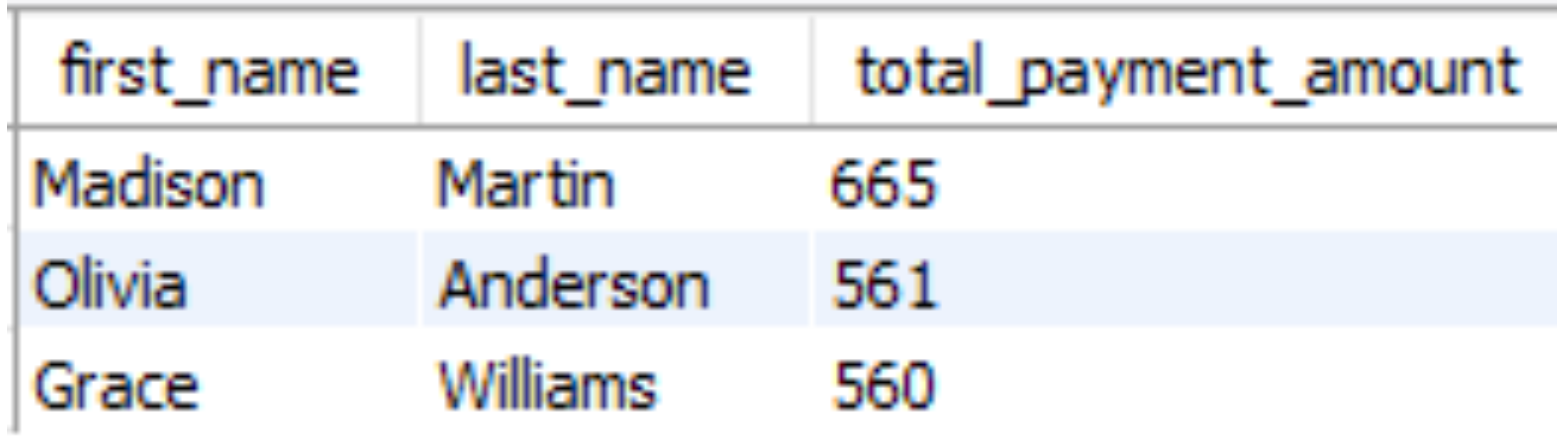
Queries and Their Outputs

```
-- QUERY 1 --
SELECT
customers.first_name,
customers.last_name
FROM CUSTOMERS
INNER JOIN reservations ON
customers.customer_id =
reservations.customer_id
WHERE reservations.reservation_date
BETWEEN '2023-01-01 00:00:00' AND
'2025-01-01 00:00:00';
```



first_name	last_name
John	Doe
Jane	Smith
Alice	Johnson
Bob	Miller
Eva	Clark
Michael	Davis
Samantha	Brown
Daniel	White
Grace	Williams
Ryan	Jones
Olivia	Anderson
Matthew	Lee
Emma	Garcia
William	Harris

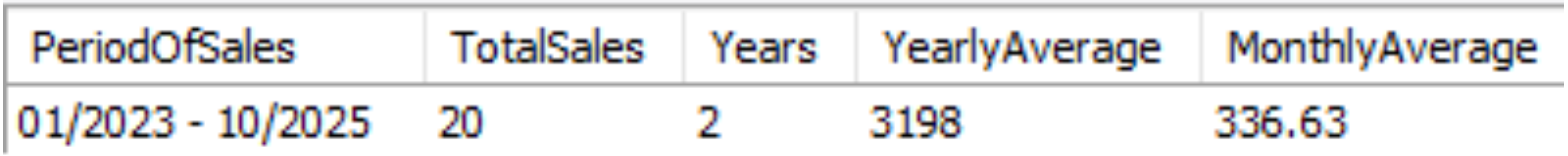
```
-- QUERY 2 --
SELECT
c.first_name,
c.last_name,
SUM(p.amount) AS total_payment_amount
FROM  CUSTOMERS c
INNER JOIN PAYMENTS p ON c.customer_id = p.customer_id
GROUP BY  c.customer_id
ORDER BY total_payment_amount DESC
LIMIT 3;
```



first_name	last_name	total_payment_amount
Madison	Martin	665
Olivia	Anderson	561
Grace	Williams	560

Queries and Their Outputs

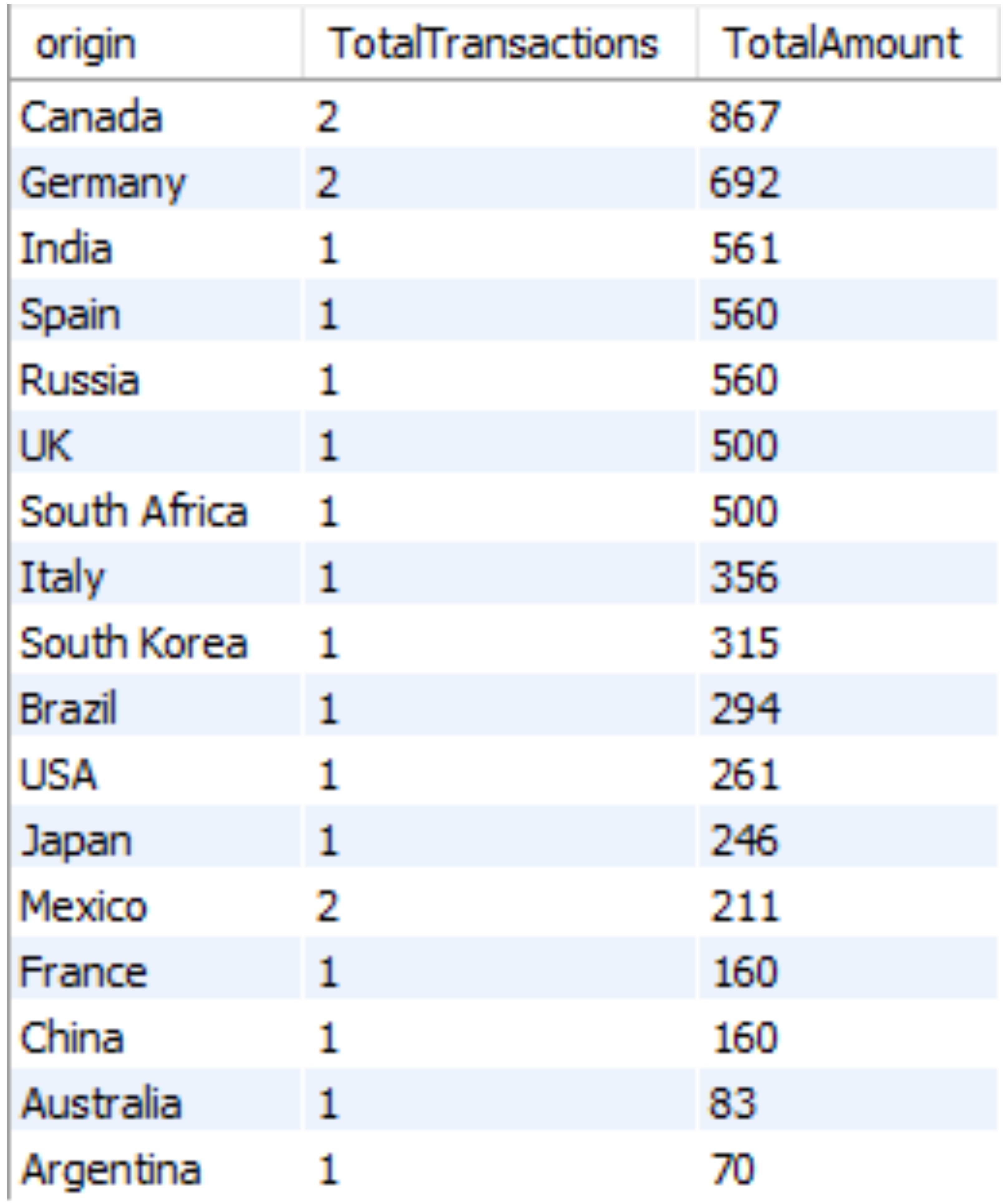
```
-- QUERY 3 --
SELECT    '01/2023 - 10/2025' AS PeriodOfSales,    COUNT(amount) AS TotalSales,
TIMESTAMPDIFF(YEAR, MIN(payment_date), MAX(payment_date)) + 1 AS Years,
ROUND(SUM(amount) / (TIMESTAMPDIFF(YEAR, MIN(payment_date),
MAX(payment_date)) + 1), 2) AS YearlyAverage,
ROUND(SUM(amount) / TIMESTAMPDIFF(MONTH, MIN(payment_date),
MAX(payment_date)), 2) AS MonthlyAverage
FROM  PAYMENTS
WHERE payment_date BETWEEN '2023-01-01' AND '2025-10-31';
```



PeriodOfSales	TotalSales	Years	YearlyAverage	MonthlyAverage
01/2023 - 10/2025	20	2	3198	336.63

Queries and Their Outputs

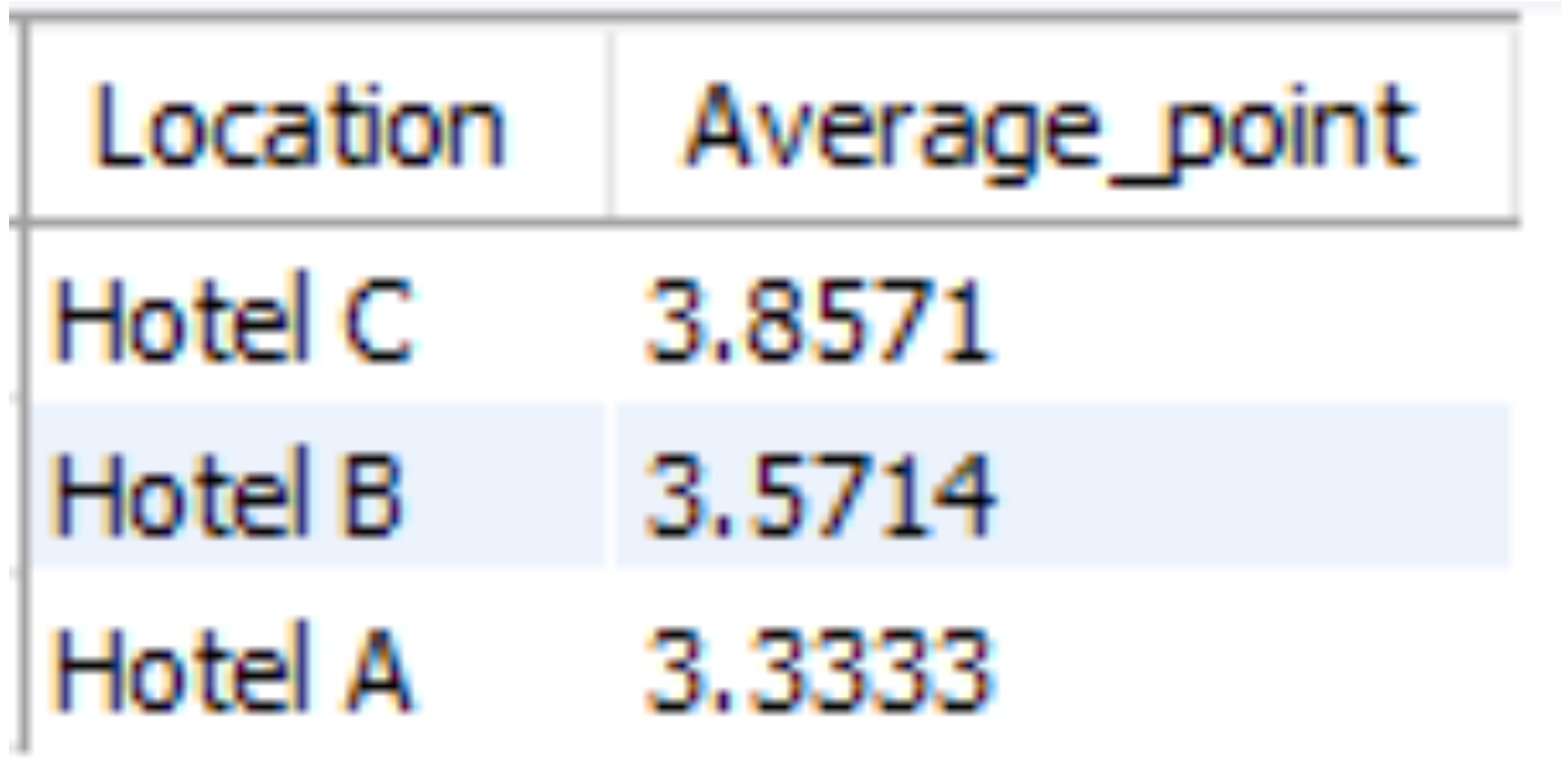
```
-- QUERY 4 --
SELECT customers.origin,
COUNT(*) AS TotalTransactions,
SUM(amount) AS TotalAmount FROM RESERVATIONS
JOIN PAYMENTS ON RESERVATIONS.reservation_id =
PAYMENTS.reservation_id
JOIN CUSTOMERS ON customers.customer_id =
RESERVATIONS.customer_id
GROUP BY CUSTOMERS.origin
ORDER BY  TotalAmount DESC;
```



origin	TotalTransactions	TotalAmount
Canada	2	867
Germany	2	692
India	1	561
Spain	1	560
Russia	1	560
UK	1	500
South Africa	1	500
Italy	1	356
South Korea	1	315
Brazil	1	294
USA	1	261
Japan	1	246
Mexico	2	211
France	1	160
China	1	160
Australia	1	83
Argentina	1	70

Queries and Their Outputs

```
-- QUERY 5 --
FIFTH ONE
SELECT     r.reservation_location AS Location,
AVG(rv.rating) AS Average_pointFROM
RESERVATIONS r
JOIN  REVIEWS rv ON r.reservation_id = rv.reservation_id
GROUP BY  r.reservation_location
ORDER BY     Average_point desc;
```



Location	Average_point
Hotel C	3.8571
Hotel B	3.5714
Hotel A	3.3333