# Hotel Booking Management Database for Husky Resort

## Industry Overview – Online Booking Services

The travel booking services industry is undergoing a transformative shift driven by an increasing demand for seamless and efficient booking experiences. While direct bookings through travel agencies hold the lion's share of revenue at 80%, online booking services are growing at a faster rate of 10.7% CAGR. In fact, 72% of customers surveyed said they preferred booking their trips online in 2023. While third party reservation sites like Expedia and Booking.com allow customers to reserve hotel rooms easily, they charge hotels between 10-30% in commission for each reservation made. This significantly reduces the margin of profit hotels can retain. As online bookings continue to become more prevalent, hotels must find a way to reduce the costs associated with acquiring customers.

## About Husky Resort

Husky Resort is a fictional hotel which has experienced lower profit margins due to the commission they have to pay third party booking sites. The company wishes to add a booking functionality to their website so that guests have the option to make a reservation directly with them. To incentivize guests, they also wish to introduce promotional offers for bookings made on their website. For the booking feature, they wish to keep track of each guest, room, payment and promotion that is involved with making a reservation. They also wish to include a 'reviews' feature that guests can use to share their experiences at the hotel.

## Project Objective

The objective of this project is to build a relational database that can effectively store all the relevant information associated with a room reservation. The database will record contact and payment information for guests, the hotel room associated with their reservation, and the type of room they have booked. Since guests can book multiple rooms, and each room can host multiple guests over time – a relational database will be essential to remove data redundancies and avoid data anomalies. A well designed database will make it possible to glean powerful insights about the resort's business performance, and also support quick and efficient data retrieval necessary for routine business operations.

## Requirements

1. One guest can place one or more reservations, each reservation can only be attributed to one guest.
2. One guest can have one or more phone numbers, each phone number is associated with only one guest.
3. One guest can leave zero to many reviews, each review must be attributed to one guest.
4. Each reservation must only be associated with one room, each room can have multiple reservations.
5. Each room belongs to one building, each building has multiple rooms
6. Each room belongs to one type, each type can be associated with one or more rooms.
7. Each reservation can only be associated with one payment, each payment can contain one or more reservations.
8. Each reservation can be associated with zero or one promotion, each promotion can be associated with zero to many reservations.

## Cardinalities Between Entities

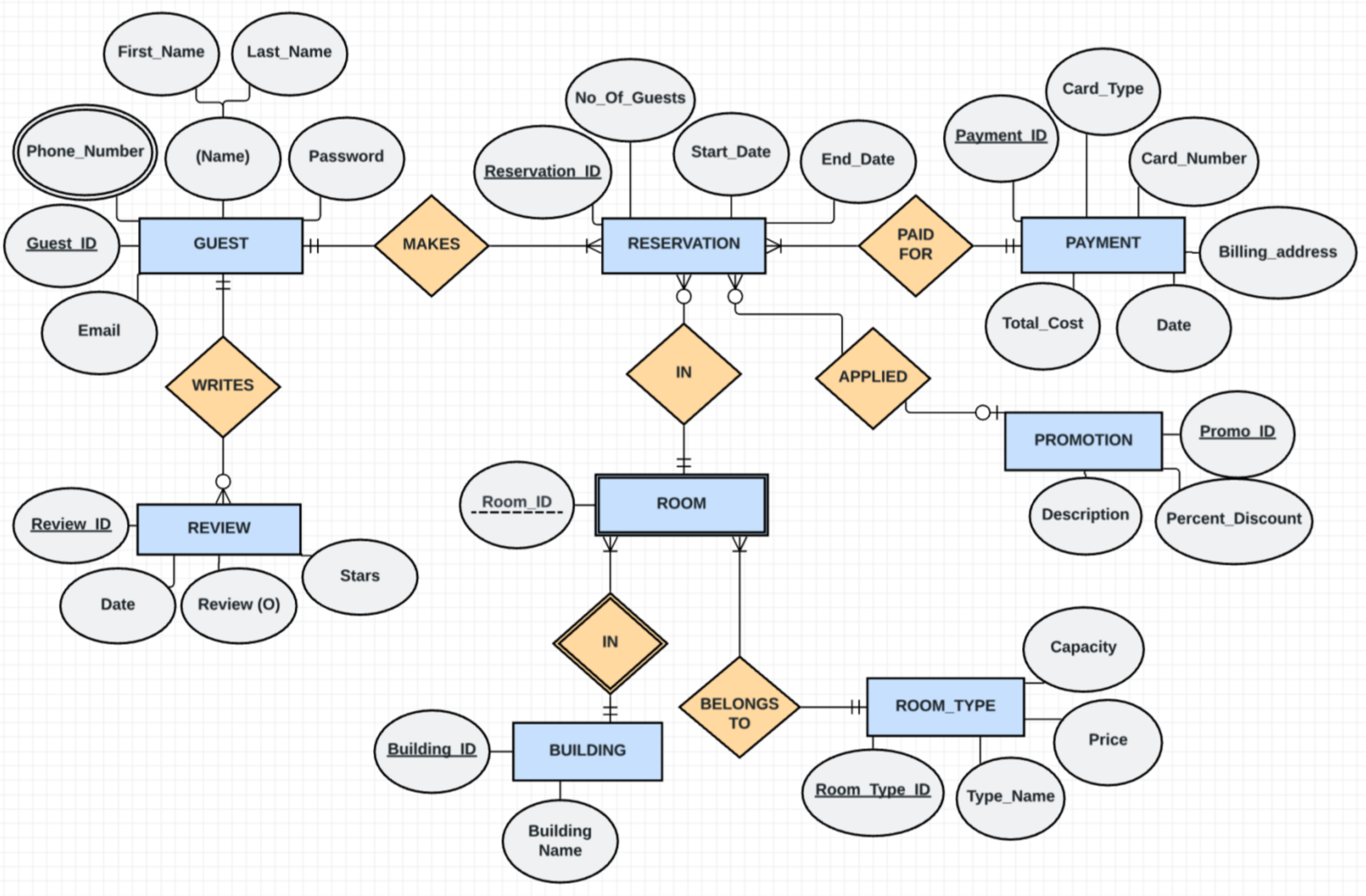
1. **Guest** *(mandatory one)* - **Reservation** *(mandatory many)*
2. **Guest** *(mandatory one)* - **Review** *(optional many)*
3. **Guest** *(mandatory one)* - **Phone_Number** *(mandatory many)*
4. **Payment** *(mandatory one)* - **Reservation** *(mandatory many)*
5. **Promotion** *(optional one)* - **Reservation** *(optional many)*
6. **Room** *(mandatory one)* - **Reservation** *(optional many)*
7. **RoomType** *(mandatory one)* - **Room** *(mandatory many)*
8. **Building** *(mandatory one)* - **Room** *(mandatory many)*

*PTO*

### **Attributes of each entity** (excluding foreign keys)

| Entity | Attributes |
|---|---|
| Guest | Guest_ID, Password, First_Name, Last_Name, Email, Phone_Number |
| Reservation | Reservation_ID, Start_Date, End_Date,  Number_Of_Guests |
| Room | Room_ID, Room_Type_ID |
| Building | Building_ID, Building_Name |
| RoomType | Room_Type_ID, Type_Name, Price, Capacity |
| Payment | Payment_ID, Card_Number, Card_Type, Billing_Address, Date, Total_Cost |
| Review | Review_ID, Review_Date, Review, Stars |
| Promotion | Promo_ID, Percent_Discount, Description |
| Phone | Phone_ID, Phone_Number |

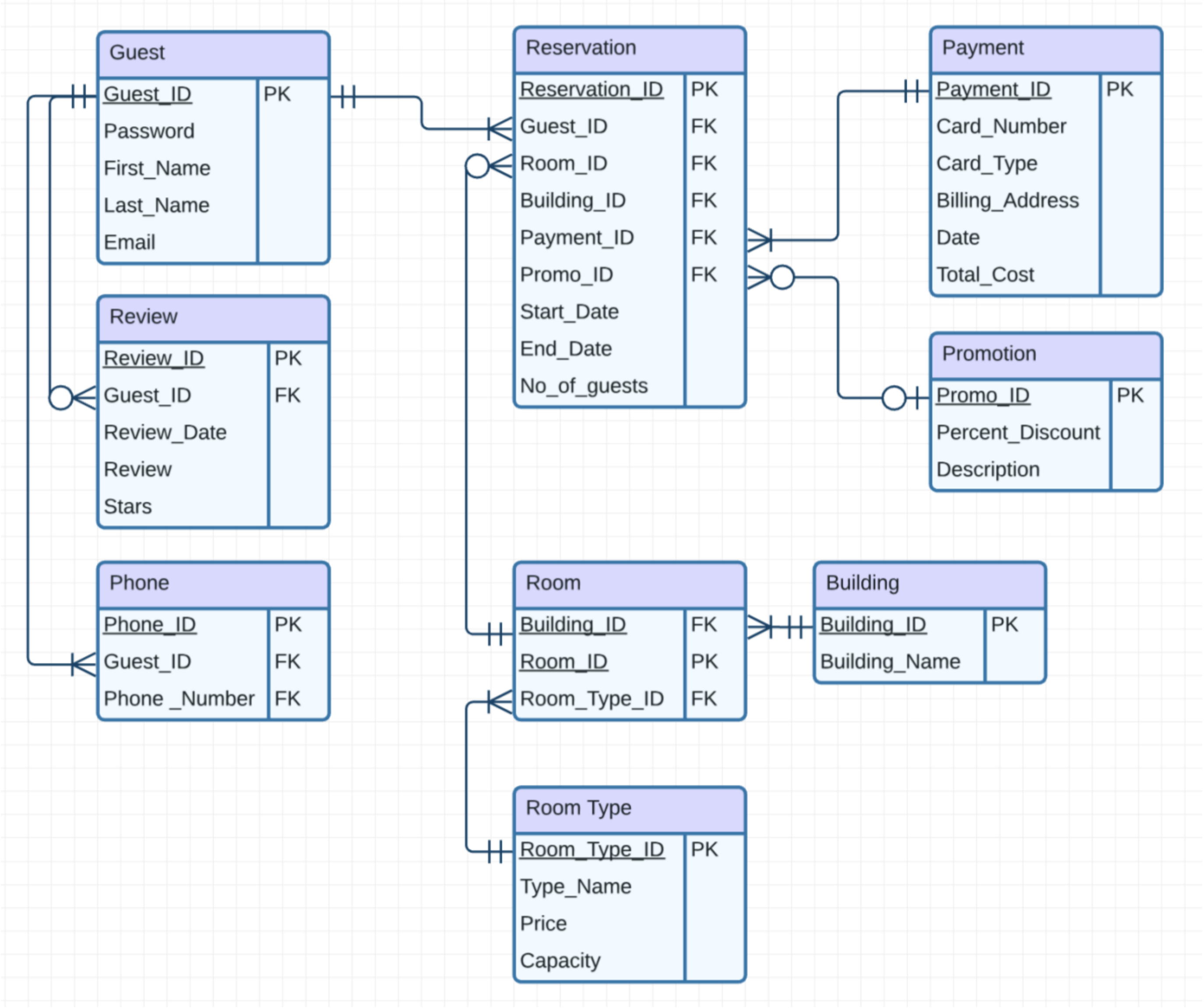
## Entity Relationship Diagram



Factors to note:

1. ‘Phone_Number’ attribute is a multivalued attribute and must be removed from the ‘Guest’ table and stored within a different entity for normalization.
2. ‘Review’ attribute (storing written text) is an optional attribute and must not have a ‘NOT NULL’ constraint.
3. ‘Room’ is a weak entity, and Room_ID (which is the room number for each room) cannot uniquely identify each room in the resort. A composite key with ‘Building_ID’ and ‘Room_ID’ would be required.

## Schema



Factors to note:

1. Multivalued attribute ‘Phone_Number’ removed from ‘Guest’ table, added to new table ‘Phone’ with ‘Guest_ID’ as foreign key.
2. ‘Room’ table has a composite key with ‘Building_ID’ and ‘Room_ID’ to uniquely identify each room.
3. Since there are no multivalued attributes, partial key dependencies or transitive dependencies, this schema is in third normal form.

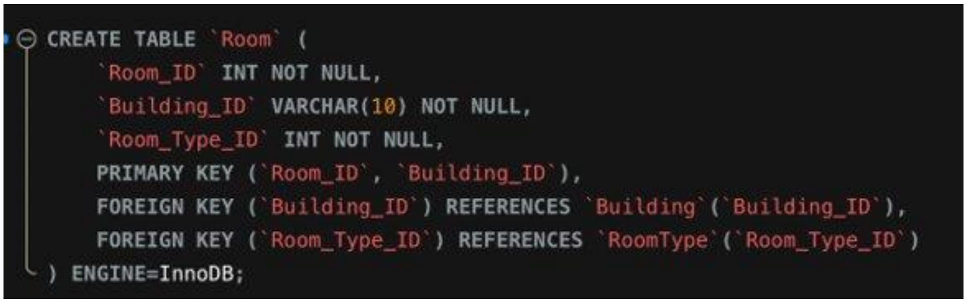
## Database Creation

The normalized database schema is now implemented through the creation of tables described in the schema. The following process is followed:

1. Creating tables with no foreign keys and inserting observations.
2. Creating tables with foreign keys and inserting observations.
3. Creating a trigger to ensure dates of new reservations do not overlap with dates of existing reservations.

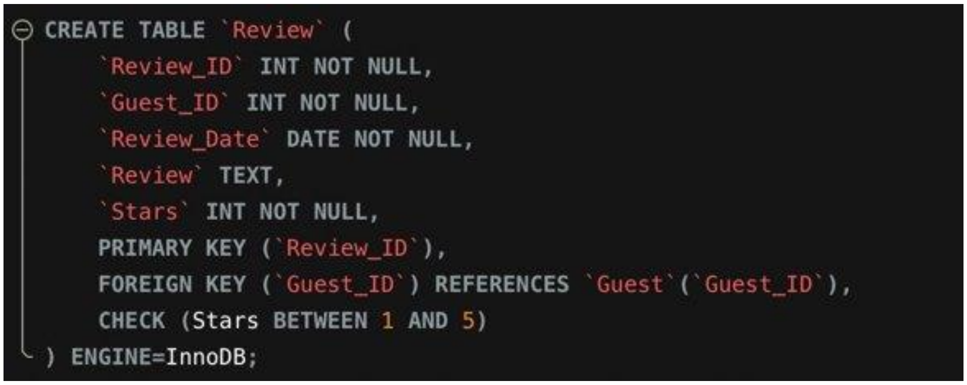
Constraints

To ensure entity integrity and referential integrity, Primary Key, Foreign Key and Not Null constraints are applied to tables wherever they are relevant.



```sql
CREATE TABLE `Room` (
    `Room_ID` INT NOT NULL,
    `Building_ID` VARCHAR(10) NOT NULL,
    `Room_Type_ID` INT NOT NULL,
    PRIMARY KEY (`Room_ID`, `Building_ID`),
    FOREIGN KEY (`Building_ID`) REFERENCES `Building`(`Building_ID`),
    FOREIGN KEY (`Room_Type_ID`) REFERENCES `RoomType`(`Room_Type_ID`)
) ENGINE=InnoDB;
```

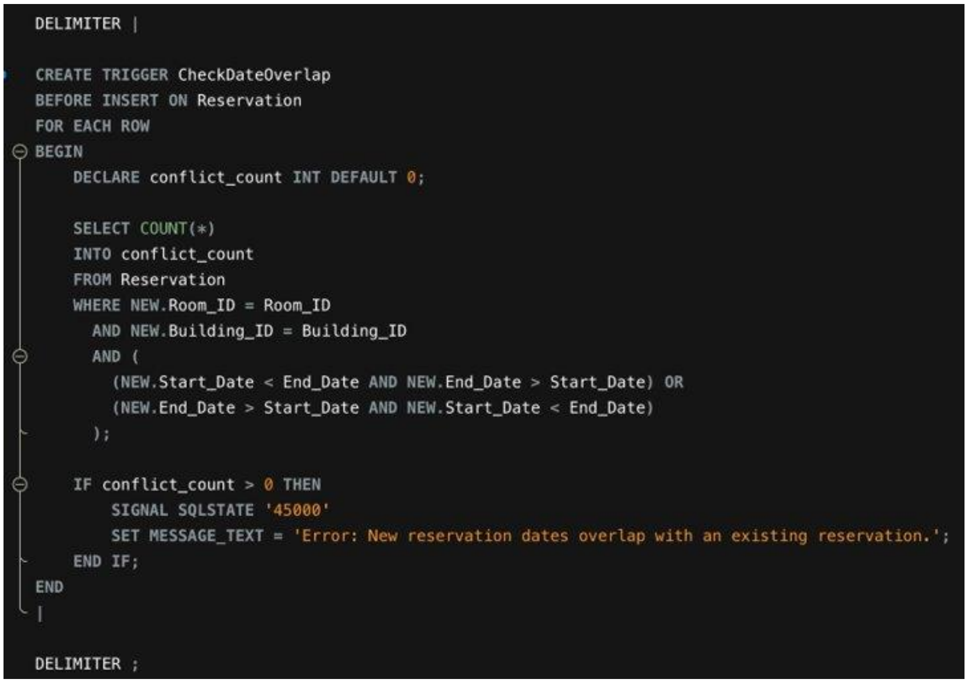
To ensure accuracy of data entry, the Check constraint is used in the ‘Stars’ column in the ‘Review’ table to ensure values are between 1 and 5.



```sql
CREATE TABLE `Review` (
    `Review_ID` INT NOT NULL,
    `Guest_ID` INT NOT NULL,
    `Review_Date` DATE NOT NULL,
    `Review` TEXT,
    `Stars` INT NOT NULL,
    PRIMARY KEY (`Review_ID`),
    FOREIGN KEY (`Guest_ID`) REFERENCES `Guest`(`Guest_ID`),
    CHECK (Stars BETWEEN 1 AND 5)
) ENGINE=InnoDB;
```

Triggers

'CheckDateOverlap' trigger is created to avoid double booking rooms, by ensuring there is no overlap in reservation dates for the same room.



```
DELIMITER |

CREATE TRIGGER CheckDateOverlap
BEFORE INSERT ON Reservation
FOR EACH ROW
BEGIN
    DECLARE conflict_count INT DEFAULT 0;

    SELECT COUNT(*)
    INTO conflict_count
    FROM Reservation
    WHERE NEW.Room_ID = Room_ID
      AND NEW.Building_ID = Building_ID
      AND (
        (NEW.Start_Date < End_Date AND NEW.End_Date > Start_Date) OR
        (NEW.End_Date > Start_Date AND NEW.Start_Date < End_Date)
      );

    IF conflict_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: New reservation dates overlap with an existing reservation.';
    END IF;
END
|

DELIMITER ;
```

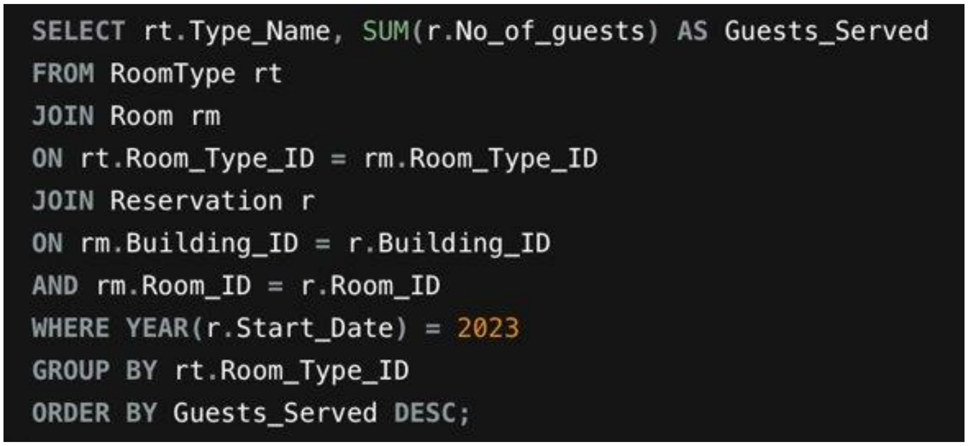
## Business Questions

To demonstrate how Husky Resort can use the database to gain insights about their business, we formulate some sample business questions that can be answered through querying the database:

**Question 1:** How many guests have stayed in rooms of each room type in 2023?

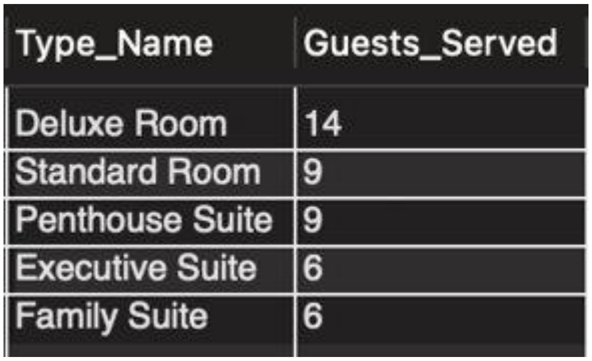
*PTO*

**Query:**



```sql
SELECT rt.Type_Name, SUM(r.No_of_guests) AS Guests_Served
FROM RoomType rt
JOIN Room rm
ON rt.Room_Type_ID = rm.Room_Type_ID
JOIN Reservation r
ON rm.Building_ID = r.Building_ID
AND rm.Room_ID = r.Room_ID
WHERE YEAR(r.Start_Date) = 2023
GROUP BY rt.Room_Type_ID
ORDER BY Guests_Served DESC;
```

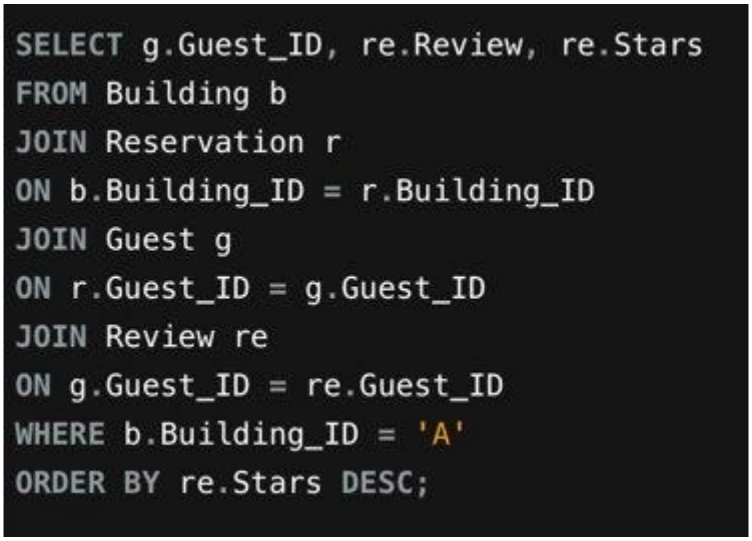
**Output:**



| Type_Name | Guests_Served |
|---|---|
| Deluxe Room | 14 |
| Standard Room | 9 |
| Penthouse Suite | 9 |
| Executive Suite | 6 |
| Family Suite | 6 |

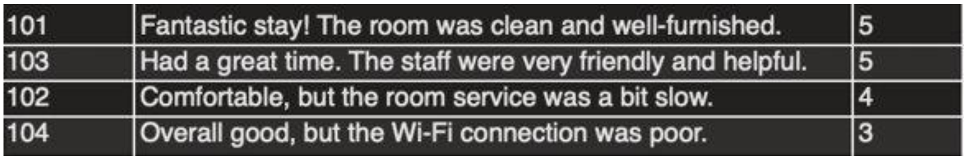
**Question 2:** What were the reviews and ratings left by guests staying in building 'A'?

**Query:**



```sql
SELECT g.Guest_ID, re.Review, re.Stars
FROM Building b
JOIN Reservation r
ON b.Building_ID = r.Building_ID
JOIN Guest g
ON r.Guest_ID = g.Guest_ID
JOIN Review re
ON g.Guest_ID = re.Guest_ID
WHERE b.Building_ID = 'A'
ORDER BY re.Stars DESC;
```

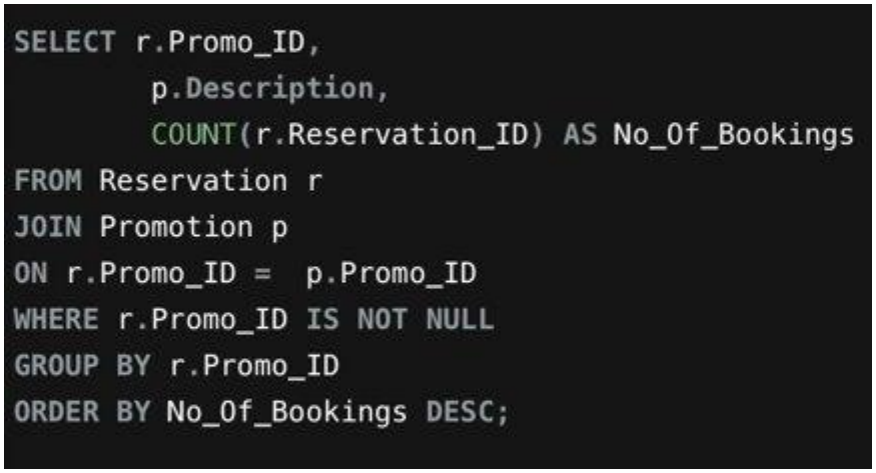
**Output:**



| | | |
|---|---|---|
| 101 | Fantastic stay! The room was clean and well-furnished. | 5 |
| 103 | Had a great time. The staff were very friendly and helpful. | 5 |
| 102 | Comfortable, but the room service was a bit slow. | 4 |
| 104 | Overall good, but the Wi-Fi connection was poor. | 3 |

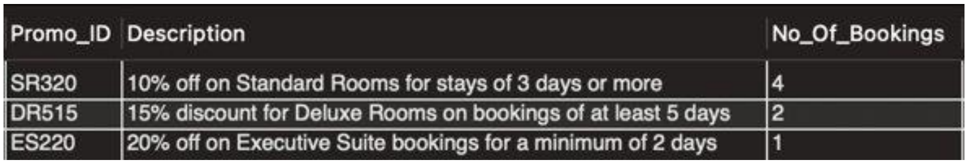
**Question 3:** How many reservations were associated with each promotional offer?

**Query:**



```
SELECT r.Promo_ID,
       p.Description,
       COUNT(r.Reservation_ID) AS No_Of_Bookings
FROM Reservation r
JOIN Promotion p
ON r.Promo_ID =  p.Promo_ID
WHERE r.Promo_ID IS NOT NULL
GROUP BY r.Promo_ID
ORDER BY No_Of_Bookings DESC;
```

**Output:**



| Promo_ID | Description | No_Of_Bookings |
|---|---|---|
| SR320 | 10% off on Standard Rooms for stays of 3 days or more | 4 |
| DR515 | 15% discount for Deluxe Rooms on bookings of at least 5 days | 2 |
| ES220 | 20% off on Executive Suite bookings for a minimum of 2 days | 1 |

## Conclusion

The reservations database has been designed keeping in mind the unique needs and requirements of Husky Resort. It can effectively store information about reservations, promotions and reviews and can serve as the back end for the resort's updated website. The database has been normalized to ensure easy updating, deletion and creation of new observations while avoiding data anomalies or redundancies. Check constraints and triggers have been used to ensure data validity and curtail the possibility of double-booking rooms for overlapping dates. As demonstrated above, it can also be easily queried to answer highly specific questions and support the resort as their business strategy evolves.

## Sources:

1. Armstrong, M., & Richter, F. (2023, March 31). *Infographic: Travel bookings: Online vs. agency*. Statista Daily Data. https://www.statista.com/chart/29622/travel-bookings-online-vs-agency/
2. *Online travel booking service market size, share report, 2030*. Online Travel Booking Service Market Size, Share Report, 2030. (n.d.). https://www.grandviewresearch.com/industry-analysis/online-travel-booking-service-market-report
3. *Understanding Ota Commission rates: A Guide for Hotel Managers*. Preno HQ - Hotel Management Software. (n.d.). https://prenohq.com/blog/understanding-ota-commission-rates-a-guide-for-hotel-managers/#:~:text=Booking.com%2C%20one%20of%20the,of%203%25%20for%20every%20booking.