# Uncovering Trends and Outliers: Airbnb Listing Analysis and Insights

## Introduction:

The Airbnb Outlier Detection and Exploratory Data Analysis (EDA) project focuses on analyzing Airbnb listings across major European cities to uncover valuable insights into pricing patterns, customer satisfaction, and market dynamics. In recent years, Airbnb has transformed the hospitality industry, making it crucial to understand the factors that influence listing performance and customer satisfaction. This analysis leverages SQL to process and analyze a rich dataset of Airbnb listings, helping identify pricing anomalies, booking trends, and key performance indicators across different cities and property types.

## Project Objectives:

1. Perform comprehensive exploratory data analysis to understand listing distribution and characteristics across European cities
2. Identify and analyse price outliers using statistical methods (IQR method) to understand extreme pricing patterns
3. Analyse booking trends and revenue patterns across different cities and room types
4. Evaluate customer satisfaction metrics and their relationship with various listing attributes
5. Assess the impact of location factors (distance to metro, city center) on listing performance
6. Compare weekday versus weekend booking patterns and their pricing implications
7. Provide actionable insights for property managers and market analysts.

## Dataset Overview:

The dataset for this project consists of Airbnb listings across various cities in Europe. The dataset contains information about bookings, room types, pricing, guest satisfaction scores, distance to metro stations and city center, as well as attraction and restaurant indices. The data will be analysed to detect outliers, gain insights into booking trends, room types, pricing, and guest satisfaction, and to help improve decision-making for Airbnb property managers and analysts.

Key Columns:

- CITY: City where the property is located.
- PRICE: Price of the booking.
- DAY: Day of the booking (Weekday/Weekend).
- ROOM_TYPE: Type of the room (e.g., Shared, Private).
- GUEST_SATISFACTION: Rating given by guests.
- CLEANLINESS_RATING: Cleanliness rating given by guests.
- METRO_DISTANCE_KM: Distance to nearest metro station (in kilometers).
- CITY_CENTER_KM: Distance to city center (in kilometers).
- ATTRACTON_INDEX: Index for the attraction level in the area.
- RESTAURANT_INDEX: Index for the restaurant availability in the area.

## Preparing the Airbnb dataset for analysis in SQL:

```sql
-- Create Database
CREATE DATABASE AIRBNB;

USE AIRBNB;

-- Create Table
CREATE TABLE airbnb_data (
    CITY VARCHAR(20),
    PRICE DECIMAL(10, 4),
    DAY VARCHAR(8),
    ROOM_TYPE VARCHAR(20),
    SHARED_ROOM TINYINT,
    PRIVATE_ROOM TINYINT,
    PERSON_CAPACITY INT,
    SUPERHOST TINYINT,
    MULTIPLE_ROOMS TINYINT,
    BUSINESS TINYINT,
    CLEANLINESS_RATING DECIMAL(10, 4),
    GUEST_SATISFACTION INT,
    BEDROOMS INT,
    CITY_CENTER_KM DECIMAL(8, 4),
    METRO_DISTANCE_KM DECIMAL(8, 4),
    ATTRACTION_INDEX DECIMAL(10, 4),
    NORMALISED_ATTRACTION_INDEX DECIMAL(8, 4),
    RESTAURANT_INDEX DECIMAL(10, 4),
    NORMALISED_RESTAURANT_INDEX DECIMAL(8, 4)
);
```

Created a new database named AIRBNB and defined a table called airbnb_data with 19 columns to store key information about Airbnb listings, including city, price, room type, guest satisfaction, and proximity to metro stations and city center. Each column was assigned an appropriate data type to ensure accurate data storage and analysis.

```sql
-- Loading Data
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Airbnb Europe Dataset.csv'
INTO TABLE airbnb_data
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(CITY, PRICE, DAY, ROOM_TYPE, @shared_room, @private_room, PERSON_CAPACITY,
 @superhost, MULTIPLE_ROOMS, BUSINESS, CLEANLINESS_RATING, GUEST_SATISFACTION,
 BEDROOMS, CITY_CENTER_KM, METRO_DISTANCE_KM, ATTRACTION_INDEX, NORMALISED_ATTRACTION_INDEX,
 RESTAURANT_INDEX, NORMALISED_RESTAURANT_INDEX)
SET
    SHARED_ROOM = IF(@shared_room = 'TRUE', 1, 0),
    PRIVATE_ROOM = IF(@private_room = 'TRUE', 1, 0),
    SUPERHOST = IF(@superhost = 'TRUE', 1, 0);

SELECT * FROM airbnb_data
LIMIT 5;
```

Imported data from the Airbnb Europe Dataset.csv file into the airbnb_data table, specifying the delimiters and formatting for the data. Boolean columns like SHARED_ROOM, PRIVATE_ROOM, and SUPERHOST were converted to binary values (1 for TRUE, 0 for FALSE). Finally, we previewed the first five rows of the table to verify the data import.

---

**1. How many records are there in the dataset?**

```
SELECT COUNT(*) AS no_of_records
FROM airbnb_data;
```

| | no_of_records |
|---|---|
| ▶ | 41714 |

The dataset contains 41,714 records, representing individual Airbnb listings. This sizable dataset provides ample data for exploratory analysis and detecting patterns across various attributes like pricing, room types, and guest satisfaction.

---

**2. How many unique cities are in the European dataset?**

```
SELECT COUNT(DISTINCT CITY) AS cities
FROM airbnb_data;
```

Used COUNT(DISTINCT CITY) function to count only unique occurrences of city names in the dataset, eliminating duplicates in the count.

| | cities |
|---|---|
| ▶ | 9 |

---

**3. What are the names of the cities in the dataset?**

```
SELECT DISTINCT CITY AS cities
FROM airbnb_data;
```

Applied SELECT DISTINCT on the CITY column to retrieve a unique list of city names without any duplicates.

| | cities |
|---|---|
| ▶ | Amsterdam |
| | Athens |
| | Barcelona |
| | Berlin |
| | Budapest |
| | Lisbon |
| | Paris |
| | Rome |
| | Vienna |

---

### 4. How many bookings are there in each city?

```
SELECT city, COUNT(city) AS Number_OF_Booking
FROM airbnb_data
GROUP BY city
ORDER BY 2 DESC;
```

Created a query using COUNT(*) with GROUP BY CITY clause to aggregate booking counts per city, ordered by count in descending order.

| | city | Number_OF_Booking |
|---|---|---|
| ▶ | Rome | 9027 |
| | Paris | 6688 |
| | Lisbon | 5763 |
| | Athens | 5280 |
| | Budapest | 4022 |
| | Vienna | 3537 |
| | Barcelona | 2833 |
| | Berlin | 2484 |
| | Amsterdam | 2080 |

---

### 5. What is the total booking revenue for each city?

```
SELECT CITY, ROUND(SUM(PRICE)) AS total_booking_revenue
FROM airbnb_data
GROUP BY CITY
ORDER BY 2 DESC;
```

Used SUM(PRICE) with GROUP BY CITY to calculate total revenue per city, ordered results by revenue to show highest-earning cities first.

| CITY | total_booking_revenue |
|---|---|
| Paris | 2625250 |
| Rome | 1854073 |
| Lisbon | 1372807 |
| Amsterdam | 1192075 |
| Vienna | 854477 |
| Barcelona | 832204 |
| Athens | 801209 |
| Budapest | 709937 |
| Berlin | 607546 |

---

### 6. What is the average guest satisfaction score for each city?

```
SELECT CITY, ROUND(AVG(GUEST_SATISFACTION),1) AS avg_guest_satisfication
FROM airbnb_data
GROUP BY CITY
ORDER BY 2 DESC;
```

Calculates the average guest satisfaction score for each city by grouping the data by the CITY column. The AVG() function is used to compute the average of GUEST_SATISFACTION, and the result is rounded to one decimal place.

| CITY | avg_guest_satisfication |
|---|---|
| Athens | 95.0 |
| Budapest | 94.6 |
| Amsterdam | 94.5 |
| Berlin | 94.3 |
| Vienna | 93.7 |
| Rome | 93.1 |
| Paris | 92.0 |
| Barcelona | 91.1 |
| Lisbon | 91.1 |

Athens has the highest average guest satisfaction score (95.0), followed closely by Budapest and Amsterdam. The cities with the lowest average scores are Barcelona and Lisbon, both at 91.1, indicating a slight disparity in guest satisfaction across the cities.

### 7. What is the minimum, maximum, average, and median booking prices?

```sql
-- MINIMUM PRICE
SELECT ROUND(MIN(PRICE),2) AS min_price
FROM airbnb_data;

-- MAXIMUM PRICE
SELECT ROUND(MAX(PRICE),2) AS max_price
FROM airbnb_data;

-- AVERAGE PRICE
SELECT ROUND(AVG(PRICE),2) AS avg_price
FROM airbnb_data;

-- MEDIAN PRICE
WITH CTE_1 AS(
SELECT PRICE,
          NTILE(2) OVER (ORDER BY PRICE) AS price_group
FROM airbnb_data
)
SELECT ROUND(AVG(PRICE), 2) AS median
FROM CTE_1
WHERE price_group = 1 OR price_group = 2;
```

To find the minimum and maximum booking prices, the query uses the MIN() and MAX() functions, respectively, rounding the results to two decimal places. The average price is calculated using the AVG() function, which also rounds the result to two decimal places. For the median price, I utilized a common SQL approach with a NTILE(2) function, which splits the data into two groups based on price ordering. The median is then computed as the average of the prices in these two groups, resulting in a rounded value of the median price.

| min_price |
| --- |
| 34.78 |

| max_price |
| --- |
| 18545.45 |

| avg_price |
| --- |
| 260.09 |

| median |
| --- |
| 260.09 |

The minimum booking price is 34.78, while the maximum is a very high 18,545.45, suggesting a significant price range across listings. The average booking price is 260.09, and since the median price is the same as the average, it indicates that the distribution of booking prices is relatively symmetric, with no extreme outliers skewing the data.

**8. How many outliers are there in the price field?**

```sql
WITH cte1 AS (
SELECT PRICE,
          NTILE(4) OVER (ORDER BY PRICE) AS quartile
FROM airbnb_data
),
cte2 AS (
SELECT
        MAX(CASE WHEN quartile = 1 THEN PRICE END) AS Q1,
        MAX(CASE WHEN quartile = 3 THEN PRICE END) AS Q3
FROM cte1
),
cte3 AS (
SELECT
        Q1,
        Q3,
        (Q3 - Q1) AS IQR,
        Q1 - 1.5 * (Q3 - Q1) AS lower_bound,
        Q3 + 1.5 * (Q3 - Q1) AS upper_bound
FROM cte2
)
SELECT
    COUNT(*) AS outliers
FROM airbnb_data, cte3
WHERE PRICE < lower_bound OR PRICE > upper_bound;
```

In this query, I first used the NTILE(4) function in a common table expression (CTE) to divide the price data into quartiles. Then, I calculated the first quartile (Q1) and third quartile (Q3) using conditional aggregation. With these values, I computed the Interquartile Range (IQR) and determined the lower and upper bounds for detecting outliers (1.5 times the IQR below Q1 and above Q3). Finally, I counted the number of listings whose prices fall outside these bounds, identifying them as outliers.

| | outliers |
|---|---|
| ▶ | 2891 |

There are 2,891 outliers in the price field, indicating a significant portion of listings with prices that are either extremely low or high compared to the majority of listings, which may need further investigation to understand pricing strategies or anomalies.

**9. What are the characteristics of the outliers in terms of room type, number of bookings, and price?**

```sql
WITH cte1 AS (
SELECT PRICE,
           NTILE(4) OVER (ORDER BY PRICE) AS quartile,
           ROOM_TYPE
FROM airbnb_data
),
cte2 AS (
SELECT
        MAX(CASE WHEN quartile = 1 THEN PRICE END) AS Q1,
        MAX(CASE WHEN quartile = 3 THEN PRICE END) AS Q3
FROM cte1
),
cte3 AS (
SELECT
        Q1,
        Q3,
        (Q3 - Q1) AS IQR,
        Q1 - 1.5 * (Q3 - Q1) AS lower_bound,
        Q3 + 1.5 * (Q3 - Q1) AS upper_bound
FROM cte2
),
outliers AS (
SELECT
        ROOM_TYPE,
        PRICE
FROM airbnb_data, cte3
WHERE PRICE < lower_bound OR PRICE > upper_bound
)
SELECT
   ROOM_TYPE AS "Room_Type",
    COUNT(*) AS "No_Of_Booking",
    ROUND(MIN(PRICE), 1) AS "minimun_outlier_price",
    ROUND(MAX(PRICE), 1) AS "maximum_outlier_price",
    ROUND(AVG(PRICE), 1) AS "average_outlier_price"
FROM outliers
GROUP BY ROOM_TYPE;
```

Used the same approach as the previous outlier detection by dividing the price data into quartiles and calculating the Interquartile Range (IQR) to find the lower and upper bounds for outliers. Then, I selected the room type and price of the outlier listings. Finally, I aggregated the results by ROOM_TYPE, calculating the number of bookings (COUNT()), and the minimum, maximum, and average prices for the outliers in each room type.

| | Room_Type | No_Of_Booking | minimun_outlier_price | maximum_outlier_price | average_outlier_price |
|---|---|---|---|---|---|
| ▶ | Private room | 353 | 528.2 | 13664.3 | 822.8 |
| | Entire home/apt | 2536 | 527.5 | 18545.5 | 853.2 |
| | Shared room | 2 | 556.2 | 591.2 | 573.7 |

**10. How does the average price differ between the main dataset and the dataset with outliers removed?**

```sql
CREATE OR REPLACE VIEW CLEANED_AIRBNB_DATA AS
(
    WITH PRICE_QUARTILES AS (
        SELECT
            PRICE,
            NTILE(4) OVER (ORDER BY PRICE) AS quartile
        FROM AIRBNB_DATA
    ),
    FIVE_NUMBER_SUMMARY AS (
        SELECT
            MAX(CASE WHEN quartile = 1 THEN PRICE END) AS Q1,
            MAX(CASE WHEN quartile = 2 THEN PRICE END) AS MEDIAN,
            MAX(CASE WHEN quartile = 3 THEN PRICE END) AS Q3
        FROM PRICE_QUARTILES
    ),
    HINGES AS (
        SELECT
            Q1,
            Q3,
            (Q3 - Q1) AS IQR,
            (Q1 - 1.5 * (Q3 - Q1)) AS LOWER_HINGE,
            (Q3 + 1.5 * (Q3 - Q1)) AS UPPER_HINGE
        FROM FIVE_NUMBER_SUMMARY
    )
    -- Select rows where prices are within the bounds to form the cleaned dataset
    SELECT
        *
    FROM AIRBNB_DATA
    WHERE PRICE >= (SELECT LOWER_HINGE FROM HINGES)
      AND PRICE <= (SELECT UPPER_HINGE FROM HINGES)
);

SELECT
    'Original Data' AS DATASET,
    ROUND(AVG(PRICE), 2) AS AVG_PRICE
FROM AIRBNB_DATA
UNION ALL
SELECT
    'Cleaned Data' AS DATASET,
    ROUND(AVG(PRICE), 2) AS AVG_PRICE
FROM CLEANED_AIRBNB_DATA;
```

First created a view called CLEANED_AIRBNB_DATA that filters out outliers by using the Interquartile Range (IQR) method. The data is divided into quartiles, and the lower and upper hinges are calculated to identify the price bounds. The cleaned dataset consists of listings with prices within these bounds. Then, I compared the average price of the original dataset (AVG(PRICE)) with the cleaned dataset, which excludes outliers, using a UNION ALL query to display both results.

| | DATASET | AVG_PRICE |
|---|---|---|
| ▶ | Original Data | 260.09 |
| | Cleaned Data | 216.22 |

The average price for the original dataset is 260.09, while the average price for the cleaned dataset (with outliers removed) is 216.22. This indicates that outliers, particularly high-priced listings, were significantly inflating the average price, and removing them provides a more accurate representation of typical booking prices.

---

### 11. What is the average price for each room type?

```sql
SELECT ROOM_TYPE, ROUND(AVG(price), 1) AS average_price
FROM airbnb_data
GROUP BY ROOM_TYPE
ORDER BY 2 DESC;
```

Calculates the average price for each room type by grouping the data by the ROOM_TYPE column and applying the AVG() function to the PRICE field. The result is rounded to one decimal place and ordered in descending order based on the average price, allowing for easy comparison of room types.

| | ROOM_TYPE | average_price |
|---|---|---|
| ▶ | Entire home/apt | 290.1 |
| | Private room | 198.4 |
| | Shared room | 137.8 |

The "Entire home/apt" room type has the highest average price (290.1), followed by "Private room" (198.4), and "Shared room" (137.8). This suggests that entire homes or apartments are significantly more expensive on average than private and shared rooms, which is expected given the larger space and greater privacy offered.

---

### 12. How do weekend and weekday bookings compare in terms of average price and number of bookings?

```sql
SELECT
    DAY AS Day_Type,
    ROUND(AVG(PRICE), 0) AS Avg_Price,
    COUNT(ROOM_TYPE) AS Num_Of_Booking
FROM cleaned_airbnb_data
GROUP BY DAY;
```

Grouped the data by DAY (representing weekdays and weekends) and calculated two key metrics: the average price of bookings (AVG(PRICE)) and the total number of bookings (COUNT(ROOM_TYPE)) for each day type. The results are rounded to zero decimal places for price and grouped by the DAY field, which indicates whether the booking is for a weekday or weekend.

| | Day_Type | Avg_Price | Num_Of_Booking |
|---|---|---|---|
| ▶ | Weekday | 213 | 19505 |
| | Weekend | 219 | 19318 |

The average price for weekend bookings (219) is slightly higher than for weekday bookings (213), indicating that weekend stays tend to be priced a little higher. However, the number of bookings is quite close, with weekdays having 19,505 bookings and weekends having 19,318 bookings, showing that demand remains strong during both weekdays and weekends.

---

**13. What is the average distance from metro and city center for each city?**

```sql
SELECT
    CITY,
    ROUND(AVG(METRO_DISTANCE_KM), 2) AS AVG_METRO_DISTANCE_KM,
    ROUND(AVG(CITY_CENTER_KM), 2) AS AVG_CITY_CENTER_KM
FROM airbnb_data
GROUP BY CITY;
```

Calculated the average distance from the metro station (AVG(METRO_DISTANCE_KM)) and from the city center (AVG(CITY_CENTER_KM)) for each city. The data is grouped by CITY, and the results are rounded to two decimal places for better clarity.

| | CITY | AVG_METRO_DISTANCE_KM | AVG_CITY_CENTER_KM |
|---|---|---|---|
| ▶ | Amsterdam | 1.09 | 2.83 |
| | Athens | 0.48 | 1.80 |
| | Barcelona | 0.44 | 2.12 |
| | Berlin | 0.84 | 5.26 |
| | Budapest | 0.54 | 1.87 |
| | Lisbon | 0.71 | 1.97 |
| | Paris | 0.23 | 3.00 |
| | Rome | 0.82 | 3.03 |
| | Vienna | 0.53 | 3.14 |

### 14. What is the booking revenue for each room type on weekdays vs weekends?

```
SELECT
    CASE
        WHEN DAY IN ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday') THEN 'Weekday'
        ELSE 'Weekend'
    END AS DAY_TYPE,
    ROOM_TYPE,
    ROUND(SUM(PRICE), 2) AS TOTAL_REVENUE
FROM AIRBNB_DATA
GROUP BY DAY_TYPE, ROOM_TYPE
ORDER BY DAY_TYPE, ROOM_TYPE;
```

Classified the days into "Weekday" and "Weekend" using a CASE statement. Then, I grouped the data by both DAY_TYPE and ROOM_TYPE to calculate the total booking revenue for each room type on weekdays and weekends. The SUM(PRICE) function was used to compute the total revenue, and the results were rounded to two decimal places for precision.

| | DAY_TYPE | ROOM_TYPE | TOTAL_REVENUE |
|---|---|---|---|
| ▶ | Weekend | Entire home/apt | 8200285.29 |
| | Weekend | Private room | 2605739.28 |
| | Weekend | Shared room | 43554.18 |

The highest revenue on weekends comes from "Entire home/apt" bookings, amounting to 8.2 million, followed by "Private room" at 2.6 million, and "Shared room" at a much lower 43.5 thousand. This suggests that larger accommodations (entire homes or apartments) generate significantly more revenue, especially on weekends when demand is higher.

---

### 15. What is the overall average, minimum, and maximum guest satisfaction score?

```
SELECT
    ROUND(AVG(GUEST_SATISFACTION), 2) AS AVG_GUEST_SATISFACTION,
    MIN(GUEST_SATISFACTION) AS MIN_GUEST_SATISFACTION,
    MAX(GUEST_SATISFACTION) AS MAX_GUEST_SATISFACTION
FROM airbnb_data;
```

The AVG() function is used to calculate the average, while the MIN() and MAX() functions were used to determine the lowest and highest guest satisfaction scores, respectively.

| | AVG_GUEST_SATISFACTION | MIN_GUEST_SATISFACTION | MAX_GUEST_SATISFACTION |
|---|---|---|---|
| ▶ | 93.10 | 20 | 100 |

The overall average guest satisfaction score is 93.10, indicating generally high satisfaction among guests. The minimum satisfaction score recorded is 20, which may suggest some extremely poor experiences, while the maximum score is 100, reflecting perfect satisfaction from some guests.

---

**16 What is the average booking value across all cleaned data?**

```
SELECT
    ROUND(AVG(PRICE), 2) AS AVG_BOOKING_VALUE
FROM CLEANED_AIRBNB_DATA;
```

Calculate the average booking value by applying the AVG() function on the PRICE field in the cleaned dataset (where outliers were removed). The result was rounded to two decimal places for precision.

| | AVG_BOOKING_VALUE |
|---|---|
| ▶ | 216.22 |

The average booking value across the cleaned data is 216.22. This value reflects the typical booking price when outliers are removed, providing a more reliable estimate of average pricing in the dataset.

---

**17. What is the average cleanliness score across all cleaned data?**

```
SELECT
    ROUND(AVG(CLEANLINESS_RATING), 2) AS AVG_CLEANLINESS_SCORE
FROM CLEANED_AIRBNB_DATA;
```

Calculate the average cleanliness rating by applying the AVG() function on the CLEANLINESS_RATING field in the cleaned dataset, which excludes outliers. The result was rounded to two decimal places to provide a more accurate measure.

| | AVG_CLEANLINESS_SCORE |
|---|---|
| ▶ | 9.44 |

The average cleanliness score across the cleaned data is 9.44, suggesting that guests generally rate cleanliness highly.

---

### 18. How do cities rank in terms of total revenue?

```
WITH CTE1 AS (
SELECT
        CITY,
        SUM(PRICE) AS TOTAL_REVENUE
FROM AIRBNB_DATA
GROUP BY CITY
)
SELECT
    CITY,
    TOTAL_REVENUE,
    ROW_NUMBER() OVER (ORDER BY TOTAL_REVENUE DESC) AS REVENUE_RANK
FROM CTE1
ORDER BY REVENUE_RANK;
```

Calculate the total revenue for each city by summing the PRICE for all listings in that city. The ROW_NUMBER() function was then used to assign a rank to each city based on its total revenue, ordered in descending order. This allows for a clear ranking of cities in terms of revenue.

| | CITY | TOTAL_REVENUE | REVENUE_RANK |
|---|---|---|---|
| ▶ | Paris | 2625250.0143 | 1 |
| | Rome | 1854073.1395 | 2 |
| | Lisbon | 1372806.9997 | 3 |
| | Amsterdam | 1192074.6162 | 4 |
| | Vienna | 854477.2420 | 5 |
| | Barcelona | 832204.2448 | 6 |
| | Athens | 801208.9483 | 7 |
| | Budapest | 709937.4894 | 8 |
| | Berlin | 607546.0477 | 9 |

The cities are ranked based on total revenue, with Paris leading at approximately 2.63 million, followed by Rome and Lisbon. This indicates that cities like Paris and Rome are key revenue-generators, which could be linked to their popularity or higher pricing for listings.

## Conclusion:

The project focuses on identifying outliers and analysing various statistics in the Airbnb dataset. By removing outliers and comparing the results, we can better understand the trends in pricing, booking, and guest satisfaction. The insights gained will help Airbnb property managers optimize their listings, adjust prices, and improve guest experience.

1. Paris and Rome are the top revenue-generating cities and should be prioritized for high-end listings and marketing efforts.
2. Room types like "Entire home/apt" are the highest revenue-generators and should be emphasized in key cities.
3. Cities with high bookings but moderate revenue should optimize their pricing strategies to improve profitability.
4. Weekend bookings generate more revenue, and targeted weekend promotions can enhance profitability.
5. Continuous monitoring of guest satisfaction and cleanliness ratings can foster loyalty and long-term business growth.