WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

Masters Programmes: Group Assignment Cover Sheet

Student Numbers: Please list numbers of all group members	5672230, 2063217, 5673642, 5669260, 5663503,5646570
Module Code:	IB9HP0
Module Title:	Data Management
Submission Deadline:	20 March 2025
Date Submitted:	20 March 2025
Word Count:	2000/2000
Number of Pages:	28
Question Attempted: *(question number/title, or description of assignment)*	This assignment is designed to simulate the development of a data product that provides business insights for a selected company. You will engage in end-to-end data management, including database design, data implementation, synthetic data generation, and report creation. The project will involve the use of SQLite for database management, SQL for querying, and data visualization techniques for reporting
Have you used Artificial Intelligence (AI) in any part of this assignment?	YES

Academic Integrity Declaration

We're part of an academic community at Warwick. Whether studying, teaching, or researching, we're all taking part in an expert conversation which must meet standards of academic integrity. When we all meet these standards, we can take pride in our own academic achievements, as individuals and as an academic community.

Academic integrity means committing to honesty in academic work, giving credit where we've used others' ideas and being proud of our own achievements.

In submitting my work, I confirm that:

- I have read the guidance on academic integrity provided in the Student Handbook and understand the University regulations in relation to Academic Integrity. I am aware of the potential consequences of Academic Misconduct.
- I declare that this work is being submitted on behalf of my group and is all our own, except where I have stated otherwise.
- No substantial part(s) of the work submitted here has also been submitted by me in other credit bearing assessments courses of study (other than in certain cases of a resubmission of a piece of work), and I acknowledge that if this has been done this may lead to an appropriate sanction.
- Where a generative Artificial Intelligence such as ChatGPT has been used I confirm I have abided by both the University guidance and specific requirements as set out in the Student Handbook and the Assessment brief. I have clearly acknowledged the use of any generative Artificial Intelligence in my submission, my reasoning for using it and which generative AI (or AIs) I have used. Except where indicated the work is otherwise entirely my own.
- I understand that should this piece of work raise concerns requiring investigation in relation to any of points above, it is possible that other work I have submitted for assessment will be checked, even if marks (provisional or confirmed) have been published.
- Where a proof-reader, paid or unpaid was used, I confirm that the proof-reader was made aware of and has complied with the University's proofreading policy.

Upon electronic submission of your assessment you will be required to agree to the statements above

Table of Contents

1. Introduction

This report presents an analytical exploration of end-to-end data management with the primary purpose of presenting business insights. The company chosen for this report is Safe Space Hotels, based in the United Kingdom, which is suitable for personal and/or business trip purposes. With locations across the UK, it is beneficial for head management to monitor performance in each branch, identify loyal customers, and analyse staff performance. The following report includes database design, data implementation, synthetic data generation, SQL querying, and data visualisation techniques.

2. Business Understanding

Safe Space Hotels operates multiple properties throughout the UK, offering accommodations for both leisure and business travellers. Safe Space Hotels manages several interconnected business processes across its UK locations, including employee management, property management for various room types, facility operations (gyms, pools, conference rooms, car parks, etc), and a multi-channel reservation system accepting both online and walk-in bookings. The company maintains customer profiles to track preferences, stay history, payments, and collects customer feedback. This integrated approach requires a comprehensive database that connects customer, room, facilities, booking, feedback, payment and employee data to support management decision-making and enhance the revenue as well as guest experience at each location.

The company's success depends on efficient operations, employee performance across all locations, and customer satisfaction. To support these objectives, a robust database system is needed to track and analyse key business insights on three critical areas:

1) Branch Performance Comparison - Identify high and low-performing locations, with particular emphasis on yearly revenue across all properties.
2) Employee Analysis - Location-specific employee analysis including employee profile and turnover rates.
3) Customer Satisfaction and Loyalty - Systematic measurement of guest feedback to identify loyal customers and develop targeted marketing strategies.

3. Data Preparation

3.1 Database Design

3.1.1 *Entity-Relationship* (ER) Diagram

The Entity Relationship Diagram (Figure 1) shows a hotel management system with interconnected entities. The central entity is HOTEL, which connects to SITE (physical location details), HOTEL_FACILITY_ASSIGNMENT (linking hotels to available facilities), ROOM (containing room specifications and rates), EMPLOYEE (staff information), and CONTRACT (employment details). CUSTOMER information feeds into the BOOKING entity, which tracks reservations including check-in/out times and purpose of stay. BOOKING connects to PAYMENT (financial transaction details) and REVIEW (customer feedback). The ER diagram shows primary keys (PK) and foreign keys (FK) with appropriate data types, establishing the relational structure needed related to hotel business processes.

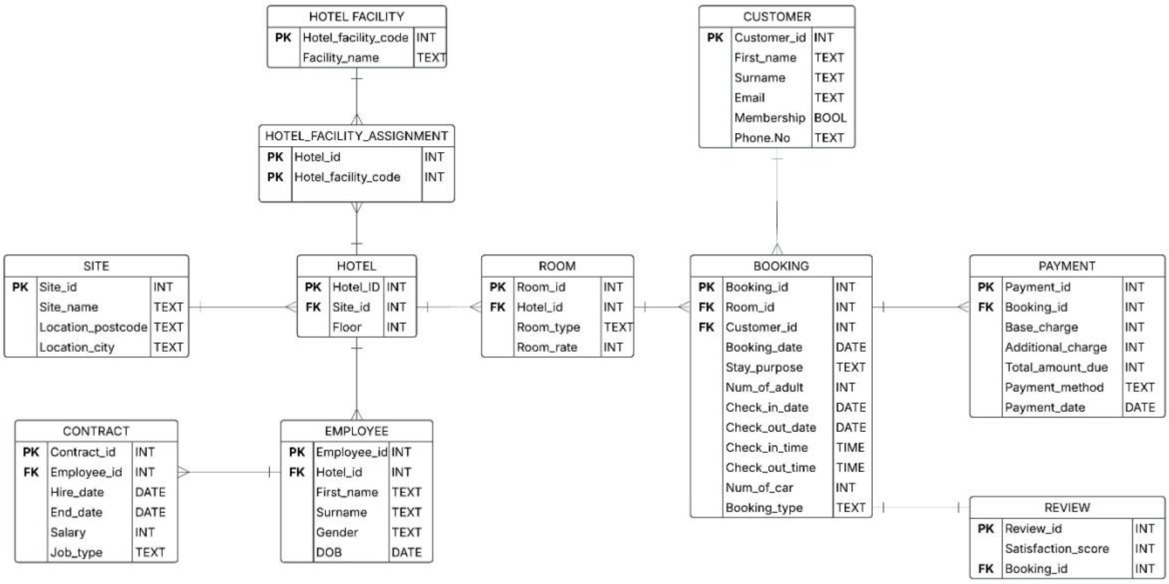


Figure 1 Entity Relationship Diagram

3.1.2 *Logical Design*

The logical design of our hotel management system defines a comprehensive relational structure between entities, where each entity contains specific attributes capturing essential information. Relationships are established through primary and foreign keys to ensure data integrity (Bhojaraju and Koganurmath, 2003). Moreover, the normalized model was implemented to eliminate redundancy while maintaining complete operational information flow across all hotel functions. Table 1 below illustrates the logical design of our database, detailing the entities, relationships, attributes, and cardinality of our hotel management system.

Table 1 Logical Design

Entities	Relationship	Attributes	Cardinality
SITE – HOTEL	A site contains multiple hotels, but each hotel belongs to only one site	**site_id** (PK) site_name location_postcode location_city **hotel_id** (PK) floor site_id (FK)	**1:M**
HOTEL – ROOM	Each hotel has multiple rooms, but each room belongs to only one hotel	**hotel_id** (PK) floor site_id (FK) **room_id** (PK) room_type room_rate hotel_id (FK)	**1:M**
HOTEL – HOTEL FACILITY ASSIGNMENT – HOTEL FACILITY	A hotel can have multiple facilities, and each facility can be available in multiple hotels.	**hotel_id** (PK) floor site_id (FK)	**M:N**
ROOM – BOOKING	A room can be booked multiple times, but each booking is for a specific room.	**room_id** (PK) room_type room_rate hotel_id (FK) **booking_id** (PK) booking_date stay_purpose num_of_adult check_in_date check_out_date check_in_time check_out_time num_of_car booking_type room_id (FK) customer_id (FK)	**1:M**
BOOKING – PAYMENT	A booking can have multiple payments (e.g., deposit + final payment), a payment belongs to only one booking.	**booking_id** (PK) booking_date stay_purpose num_of_adult check_in_date check_out_date check_in_time check_out_time num_of_car booking_type room_id (FK) customer_id (FK) **payment_id** (PK)	**1:M**

Entities	Relationship	Attributes	Cardinality
		base_charge additional_charge total_amount_due payment_method payment_date booking_id (FK)	
BOOKING – CUSTOMER	A customer can make multiple bookings, each booking is made by one customer.	**booking_id** (PK) booking_date stay_purpose num_of_adult check_in_date check_out_date check_in_time check_out_time num_of_car booking_type room_id (FK) customer_id (FK) **customer_id** (PK) first_name surname email membership mobile_phone	**1:M**
BOOKING – REVIEW	One booking can have one review.	**booking_id** (PK) booking_date stay_purpose num_of_adult check_in_date check_out_date check_in_time check_out_time num_of_car booking_type room_id (FK) customer_id (FK) **review_id** (PK) satisfaction_score booking_id (FK)	**1:1**
EMPLOYEE – CONTRACT	Each employee can have multiple contracts over time, a contract belongs to one employee.	**employee_id** (PK) first_name surname gender DOB hotel_id (FK) **contract_id** (PK) hire_date end_date salary job_type	**1:M**

Entities	Relationship	Attributes	Cardinality
		employee_id (FK)	
EMPLOYEE – HOTEL	A hotel has multiple staff members, each staff member works for only one hotel at a time.	**employee_id** (PK) first_name surname gender DOB hotel_id (FK) **hotel_id** (PK) floor site_id (FK)	**1:M**

3.1.3 *SQL Schema Implementation*

The implementation of physical design was accomplished using SQL Data Definition Language (DDL) statements, primarily the CREATE command for defining tables and other database objects (attributes, primary key, and foreign key). Each table was created with appropriate column definitions, data types, and constraints. The complete SQL implementation queries can be found in Appendix A.

All the entities utilised four primary data types: INTEGER for numeric values including IDs and financial figures, VARCHAR for string data like names and descriptions, DATE for calendar dates, and TIME for time values (Bhojaraju and Koganurmath, 2003) as detailed below:

Table 2 Data Types

Data Type	Attributes	Details
INTEGER	Site_id, Hotel_id, Floor, Room_id, Room_rate, Booking_id, Num_of_adult, Num_of_car, Review_id, Satisfaction_score, Employee_id, Contract_id, Salary, Payment_id, Base_charge, Additional_charge, Total_amount_due, Hotelfacility_code	All ID fields use INTEGER to serve as efficient primary keys. Financial values (Room_rate, Salary, Base_charge, etc.) use INTEGER to store amounts to avoid floating-point calculation errors. Count values (Num_of_adult, Num_of_car) use INTEGER as they represent whole numbers.
VARCHAR	Site_name, Location_postcode, Location_city, Room_type, First_name (Customer), Surname (Customer), Email, Membership, Phone_Number, Stay_purpose, Booking_type, First_name (Employee), Surname (Employee), Gender, Job_type, Payment_method, Facility_name	VARCHAR is particularly suitable for fields like names, addresses, and codes where length can vary significantly between records (Cummins, 2008).
DATE	Booking_date, Check_in_date, Check_out_date, DOB, Hire_date, End_date, Payment_date	Used for calendar dates to enable proper date comparisons, calculations and formatting (Cummins, 2008).
TIME	Check_in_time, Check_out_time	Selected specifically for time values.

We implemented NOT NULL constraints to protect data integrity by preventing incomplete records and ensuring all critical information is captured consistently across the database. In addition, foreign key constraints are established throughout the schema to maintain referential integrity between related tables (Tupper, 2011). In SQL, REFERENCES command establishes crucial parent-child relationships as follow:

1) HOTEL table: REFERENCES SITE(Site_id) links hotels to locations.
2) ROOM table: REFERENCES HOTEL(Hotel_ID) connects rooms to hotels.
3) BOOKING table: REFERENCES ROOM(Room_id) ensures that every Room_id value entered in the BOOKING table must exist in the ROOM table, and REFERENCES CUSTOMER(Customer_id) to ensure that every Customer_id value in the BOOKING table must exist in the CUSTOMER table.
4) REVIEW table: REFERENCES BOOKING(Booking_id) ensures reviews match bookings.
5) EMPLOYEE table: REFERENCES HOTEL(Hotel_ID) associate staff with workplaces.
6) CONTRACT table: REFERENCES EMPLOYEE(Employee_id) connects contracts to employees.
7) PAYMENT table: REFERENCES BOOKING(Booking_id) links payments to bookings.
8) HOTEL_FACILITY_ASSIGNMENT table: REFERENCES HOTEL(Hotel_ID) and REFERENCES HOTEL_FACILITY(Hotelfacility_code) maintains the many-to-many relationship.

3.2 Synthetic Data Generation

Most of the synthetic data has been generated using generative AI. ChatGPT has been used to populate tables created via SQLite, with multiple logical assumptions in place. Python has been applied to generate UK phone numbers for customers. These assumptions were iteratively tested on earlier dataset versions, leading to further refinements. This process proved lengthy and challenging until a coherent dataset was achieved.

The integrity of the final dataset has been tested and assessed, particularly the logic and consistency of the booking information. This was done to verify whether the assumptions made during data generation would hold. Our assumptions included:

1) The booking date must be on or before the check-in date, and the check-out date must be after the check-in date.
2) A room can only have one active booking at a time, preventing overbooking.
3) The check-in date and time for a guest in a particular room must be after the check-out date and time of the previous guest.
4) The number of adults per booking cannot exceed the room’s capacity based on its type.
5) The number of cars in a booking cannot exceed the number of adults in that booking.

6) The number of reviews must be less than or equal to the number of bookings.
7) Additional charges per booking must be based on the facilities assigned to the hotel and the duration of the booking. Moreover, the charge for breakfast must account for the number of adults in the booking, while the charge for parking must consider the number of cars.
8) The payment date must match the check-in date.

The final dataset consists of over 2,000 records of the focus entity BOOKING and ten supporting entity tables (see Appendix B).

4. Result and Discussion

4.1 Branch Performance Comparison

In 2024, the revenue analysis reveals varied performance across different hotel locations (see Appendix C). London generated the highest revenue at £349,585, making it the top-performing branch financially. Edinburgh followed closely with £306,900. Manchester and Birmingham delivered moderate revenue with £237,510 and £230,495 respectively. Southampton showed the lowest revenue at £226,815.

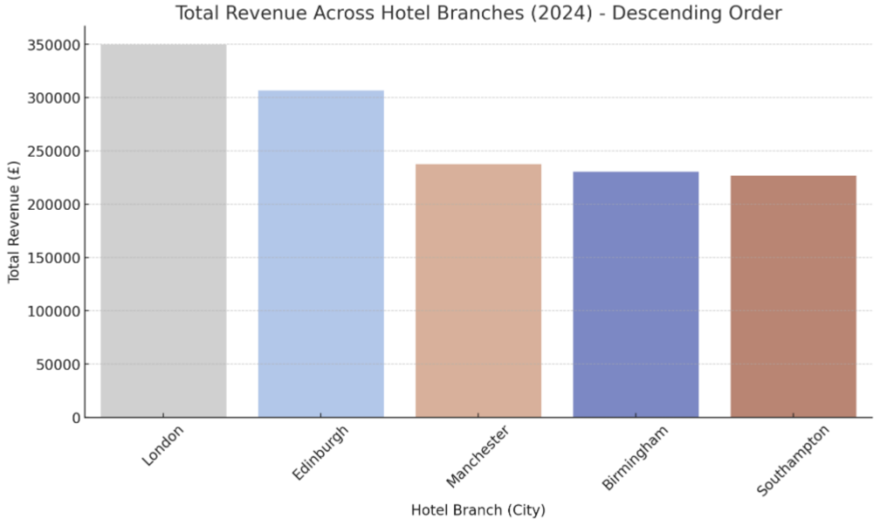


Figure 2 Total Revenue Across Hotel Branches (2024) - Descending Order

London's revenue substantially exceeds other locations, suggesting effective premium pricing strategies, as evidenced by London having the highest average room rates. In addition, London and Edinburgh generate combined revenue of £656,485, representing approximately 49% of total earnings. These leading locations benefit from strong tourism markets, business travel demand, and effective premium pricing strategies.

4.2 Employee Analysis

Salaries for chefs and receptionists remain uniform across locations, while security roles show notable pay disparities (Figure 3). Managers receive the highest salaries across all departments, significantly exceeding other roles. Additionally, part-time receptionists and housekeeping staff earn the lowest salaries, indicating a clear hierarchy in compensation based on job responsibilities and employment type.

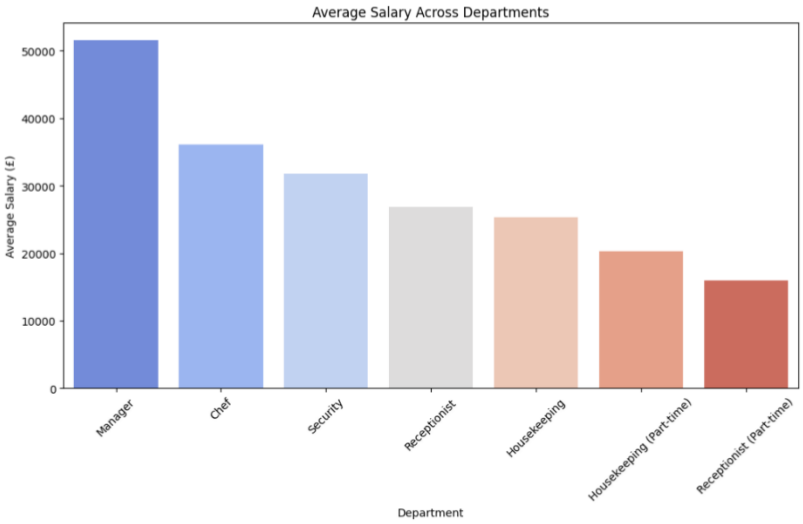


Figure 3 Average Salary Across Department

Beyond salary discrepancies, workforce distribution across branches (Figure 4) reveals variations in active contract types, reflecting different employment structures and staffing needs. Analysing these differences provides valuable insight into operational efficiency and potential areas for workforce realignment. Ensuring an optimal balance of employees across locations can help maintain service quality while controlling labour costs.

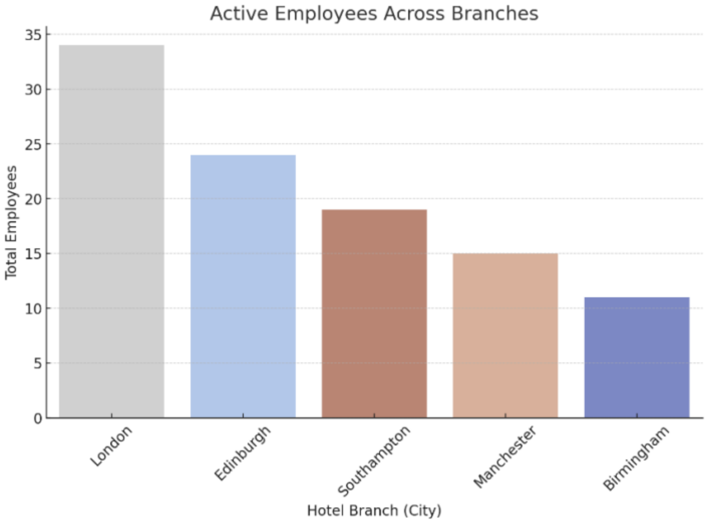


Figure 4 Active Employee Across Branches

In addition, comparing the best and worst-performing branches, Southampton shows higher average salaries for certain job types than top performers. This indicates potential cost inefficiencies that may be impacting Southampton's overall financial performance. These insights suggest an opportunity for cost optimization, particularly in Southampton, to align expenditures more effectively with revenue generation.

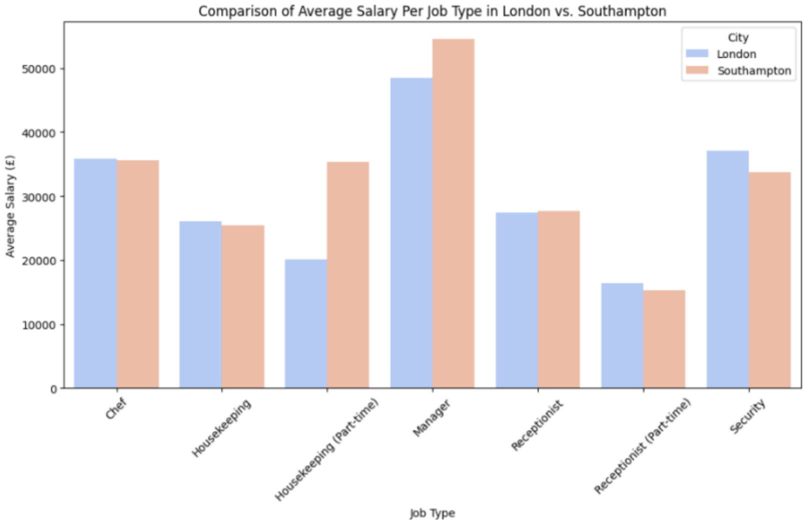


Figure 5 Average Salary Comparison in Top and Worst-Performing Branches

Moreover, the hospitality industry is known for high turnover rates, and this is reflected in our findings.

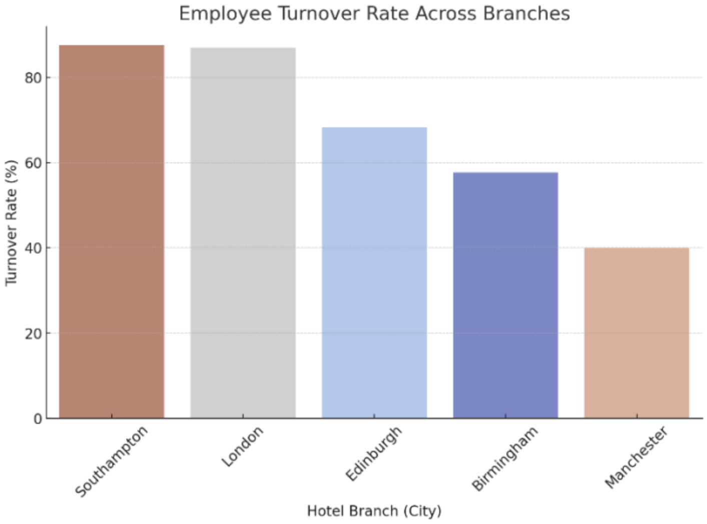


Figure 6 Employee Turnover Rate

4.3 Customer Satisfaction and Loyalty

Looking at the average customer satisfaction score (Figure 6), all five branches are performing relatively similar. This consistency suggests standardized service quality across the hotel network, with all branches maintaining comparable guest experience levels. While small variations exist, with Edinburgh and London showing slightly stronger performance and Manchester rating somewhat lower, the overall pattern indicates uniform customer satisfaction across locations rather than significant disparities between branches.

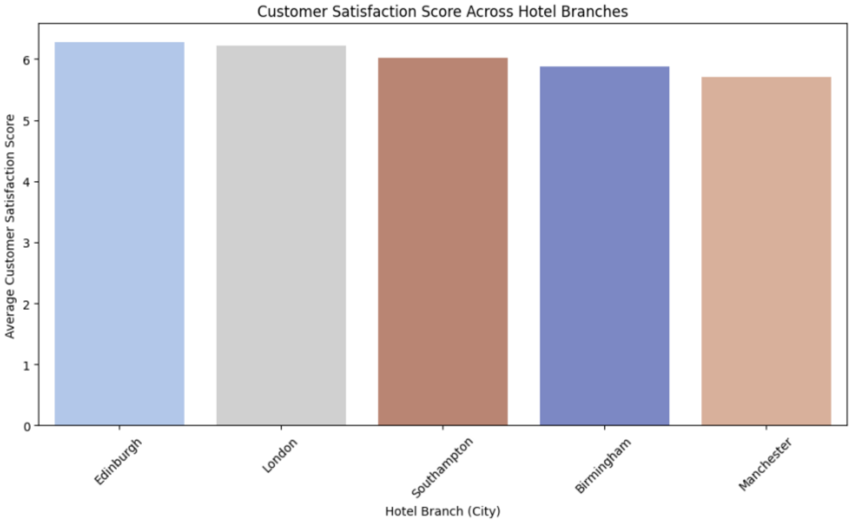


Figure 7 Customer Satisfaction Score

Based on the membership type, this pattern in Figure 5 suggests that while geographic location has minimal impact on guest satisfaction, membership status substantially enhances the experience, with gold membership creating particularly outstanding satisfaction score.

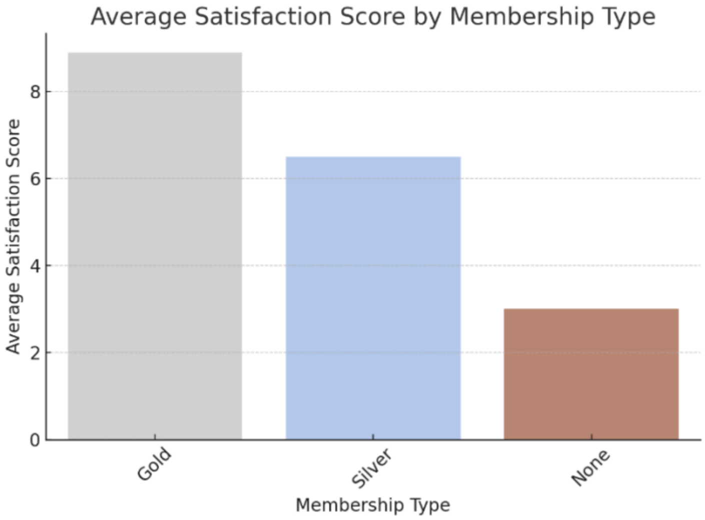


Figure 8 Average Customer Satisfaction Score by Membership

These findings suggest two key improvement strategies: implementing personalized marketing approaches and actively encouraging membership enrolment through enhanced benefits, which would likely improve overall satisfaction ratings and strengthen the hotel's reputation.

5. Conclusion

This report examines the complete database design and implementation process for Safe Space Hotels, starting from conceptual entity-relationship diagrams to logical design and SQL-based execution. A database was created, populated with synthetic data, and queried to generate meaningful business insights. The analysis of this data provided valuable findings on hotel operations, which were then compiled into a report for business decision-making.

6. References

G, Bhojaraju. and MM, Koganurmath. (2003) Database Management: Concepts and Design. In: Amitabha Chatterjee. *XXIV All India Conference of IASLIC, Survey of India, Dehra Dun, 15-18 December.* IASLI: Reasearch Gate.

Fred Cummins. (2008) *Building the Agile Enterprise: With SOA, BPM and MBM*. London: Morgan Kaufmann*.*

Charles D. Tupper. (2011) *Data Architecture: From zero to reality.* Germany: Morgan Kaufmann*.*

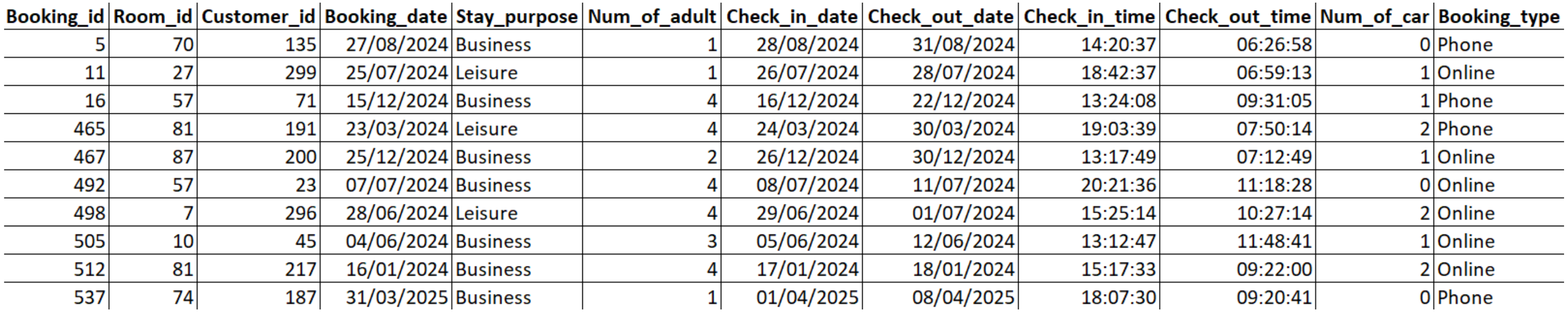
Roostedhr.com. (2025). *8 Reasons for Hospitality Staff Turnover | Roosted*. [online] Available at: https://www.roostedhr.com/blog/hospitality-jobs-have-some-of-the-highest-turnover-rates.

Appendix A: Physical Design – SQL Implementation

```
import sqlite3

# Connect to SQLite database
conn = sqlite3.connect("hotel_db.sqlite") # Connects to or creates
'hotel_db.sqlite'
cursor = conn.cursor()

# List of SQL table creation queries
create_queries = [
"""CREATE TABLE SITE (
Site_id INTEGER PRIMARY KEY,
Site_name TEXT NOT NULL,
Location_postcode TEXT NOT NULL,
Location_city TEXT NOT NULL
);""",

"""CREATE TABLE HOTEL (
Hotel_id INTEGER PRIMARY KEY,
Site_id INTEGER NOT NULL,
Floor INTEGER NOT NULL CHECK (Floor >= 0),
FOREIGN KEY (Site_id) REFERENCES SITE(Site_id) ON DELETE CASCADE
);""",

"""CREATE TABLE ROOM (
Room_id INTEGER PRIMARY KEY,
Hotel_id INTEGER NOT NULL,
Room_type TEXT NOT NULL,
Room_rate REAL NOT NULL CHECK (Room_rate >= 0),
FOREIGN KEY (Hotel_id) REFERENCES HOTEL(Hotel_id) ON DELETE CASCADE
);""",

"""CREATE TABLE CUSTOMER (
Customer_id INTEGER PRIMARY KEY,
First_name TEXT NOT NULL,
Surname TEXT NOT NULL,
Email TEXT UNIQUE NOT NULL,
Membership TEXT CHECK (Membership IN ('None', 'Silver', 'Gold')),
Phone_Number TEXT NOT NULL CHECK (length(Phone_Number) >= 10)
);""",

"""CREATE TABLE BOOKING (
Booking_id INTEGER PRIMARY KEY,
Room_id INTEGER NOT NULL,
Customer_id INTEGER NOT NULL,
Booking_date DATE NOT NULL,
Stay_purpose TEXT CHECK (Stay_purpose IN ('Business', 'Leisure')),
Num_of_adult INTEGER NOT NULL CHECK (Num_of_adult >= 0),
Check_in_date DATE NOT NULL,
Check_out_date DATE NOT NULL,
Check_in_time TIME NOT NULL,
Check_out_time TIME NOT NULL,
Num_of_car INTEGER CHECK (Num_of_car >= 0),
Booking_type TEXT CHECK (Booking_type IN ('Online', 'Phone', 'Walk-in')),
Room_rate REAL NOT NULL CHECK (Room_rate >= 0),
Duration INTEGER NOT NULL CHECK (Duration >= 0),
Base_charge REAL NOT NULL CHECK (Base_charge >= 0),
FOREIGN KEY (Room_id) REFERENCES ROOM(Room_id) ON DELETE CASCADE,
FOREIGN KEY (Customer_id) REFERENCES CUSTOMER(Customer_id) ON DELETE CASCADE
);""",

"""CREATE TABLE REVIEW (
Review_id INTEGER PRIMARY KEY,
Booking_id INTEGER NOT NULL,
Satisfaction_score INTEGER NOT NULL,
FOREIGN KEY (Booking_id) REFERENCES BOOKING(Booking_id) ON DELETE CASCADE
);""",

"""CREATE TABLE EMPLOYEE (
Employee_id INTEGER PRIMARY KEY,
Hotel_id INTEGER NOT NULL,
First_name TEXT NOT NULL,
Surname TEXT NOT NULL,
Gender TEXT,
DOB DATE NOT NULL,
FOREIGN KEY (Hotel_id) REFERENCES HOTEL(Hotel_id) ON DELETE CASCADE
);""",

"""CREATE TABLE CONTRACT (
Contract_id INTEGER PRIMARY KEY,
Employee_id INTEGER NOT NULL,
Hire_date DATE NOT NULL,
End_date DATE,
Salary REAL NOT NULL CHECK (Salary > 0),
Job_type TEXT NOT NULL,
FOREIGN KEY (Employee_id) REFERENCES EMPLOYEE(Employee_id) ON DELETE CASCADE
);""",

"""CREATE TABLE PAYMENT (
Payment_id INTEGER PRIMARY KEY,
Booking_id INTEGER NOT NULL,
Base_charge REAL NOT NULL CHECK (Base_charge >= 0),
Total_Additional_Charges REAL DEFAULT 0 CHECK
(Total_Additional_charges >= 0),
Total_amount_due REAL NOT NULL CHECK (Total_amount_due >= 0),
Payment_method TEXT,
Payment_date DATE NOT NULL,
FOREIGN KEY (Booking_id) REFERENCES BOOKING(Booking_id) ON DELETE
CASCADE
);""",

"""CREATE TABLE HOTEL_FACILITY (
Hotelfacility_code INTEGER PRIMARY KEY,
Facility_name TEXT NOT NULL
);""",

"""CREATE TABLE HOTEL_FACILITY_ASSIGNMENT (
Hotel_id INTEGER NOT NULL,
Hotelfacility_code INTEGER NOT NULL,
PRIMARY KEY (Hotel_id, Hotelfacility_code),
FOREIGN KEY (Hotel_id) REFERENCES HOTEL(Hotel_id) ON DELETE CASCADE,
FOREIGN KEY (Hotelfacility_code) REFERENCES
HOTEL_FACILITY(Hotelfacility_code) ON DELETE CASCADE
);"""
]

# Execute each query separately to avoid SQLite multi-statement
execution errors
for query in create_queries:
cursor.execute(query)

# Commit changes and close connection
conn.commit()
conn.close()

# Confirm table creation
print("All tables created successfully in SQLite database
'hotel_db.sqlite'!")
```

Appendix B: Final Dataset (display of 10 rows in each table)

BOOKING



Booking_id	Room_id	Customer_id	Booking_date	Stay_purpose	Num_of_adult	Check_in_date	Check_out_date	Check_in_time	Check_out_time	Num_of_car	Booking_type
5	70	135	27/08/2024	Business	1	28/08/2024	31/08/2024	14:20:37	06:26:58	0	Phone
11	27	299	25/07/2024	Leisure	1	26/07/2024	28/07/2024	18:42:37	06:59:13	1	Online
16	57	71	15/12/2024	Business	4	16/12/2024	22/12/2024	13:24:08	09:31:05	1	Phone
465	81	191	23/03/2024	Leisure	4	24/03/2024	30/03/2024	19:03:39	07:50:14	2	Phone
467	87	200	25/12/2024	Business	2	26/12/2024	30/12/2024	13:17:49	07:12:49	1	Online
492	57	23	07/07/2024	Business	4	08/07/2024	11/07/2024	20:21:36	11:18:28	0	Online
498	7	296	28/06/2024	Leisure	4	29/06/2024	01/07/2024	15:25:14	10:27:14	2	Online
505	10	45	04/06/2024	Business	3	05/06/2024	12/06/2024	13:12:47	11:48:41	1	Online
512	81	217	16/01/2024	Business	4	17/01/2024	18/01/2024	15:17:33	09:22:00	2	Online
537	74	187	31/03/2025	Business	1	01/04/2025	08/04/2025	18:07:30	09:20:41	0	Phone

CUSTOMER

Customer_id	First_name	Surname	Email	Membership	Phone_Number
1	Calvin	Medina	calvin.medina@example.com	None	+447788848950
2	Brian	Hodge	brian.hodge@example.com	None	+448406448780
3	Heather	Duran	heather.duran@example.com	Silver	+441845655697
4	Nicole	Gill	nicole.gill@example.com	Gold	+449890459856
5	Todd	Taylor	todd.taylor@example.com	Gold	+443543222617
6	Jessica	Wright	jessica.wright@example.com	Gold	+442171497396
7	Brianna	Beasley	brianna.beasley@example.com	None	+443429051635
8	John	Morales	john.morales@example.com	None	+441979760391
9	Doris	Williams	doris.williams@example.com	None	+441682869290
10	Chad	Joseph	chad.joseph@example.com	Silver	+446780441843

SITE

Site_id	Site_name	Location_postcode	Location_city
1	Safe Space London	SE1 9SG	London
2	Safe Space Manchester Airport	M1 1AF	Manchester
3	Safe Space Birmingham	B1 1BB	Birmingham
4	Safe Space Edinburgh	EH1 1BB	Edinburgh
5	Safe Space Southampton	SO14 7FP	Southampton

HOTEL

Hotel_id	Site_id	Floor
1	1	4
2	2	3
3	3	3
4	4	2
5	5	3

ROOM

Room_id	Hotel_id	Room_type	Room_rate
1	1	Double	150
2	1	Single	100
3	1	Single	100
4	1	Triple	220
5	1	Triple	220
6	1	Triple	220
7	1	Quad	350

8	1	Single	100
9	1	Double	150
10	1	Triple	220

PAYMENT

Payment_id	Booking_id	Base_charge	Total_Additional_Charges	Total_amount_due	Payment_method	Payment_date
1	1	595	175	770	Bank Transfer	2024-11-06
2	2	85	25	110	Cash	2024-06-20
3	3	510	300	810	Cash	2024-12-19
4	4	85	40	125	Credit Card	2024-07-03
5	5	255	45	300	Cash	2024-08-28
6	6	595	210	805	Bank Transfer	2024-01-11
7	7	425	250	675	Credit Card	2024-07-22
8	8	85	15	100	Bank Transfer	2024-06-15
9	9	85	30	115	Cash	2024-07-19
10	10	510	180	690	Cash	2025-04-01

HOTEL_FACILITY

Hotelfacility_code	Facility_name
1	Gym
2	Swimming Pool
3	Free Wi-Fi
4	Parking
5	Restaurant
6	Bar
7	Conference Room
8	Breakfast

HOTEL_FACILITY_ASSIGNMENT

Hotel_id	Hotelfacility_code
1	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
2	1
2	2

EMPLOYEE

Employee_id	Hotel_id	First_name	Surname	Gender	DOB
1	1	Shari	Thompson	Male	7/25/1969 0:00
2	1	Tony	Hudson	Female	10/22/1978 0:00
3	1	Thomas	Andrews	Male	11/24/1988 0:00
4	1	Mariah	Warren	Female	2/24/1965 0:00
5	1	Kristin	Young	Male	12/28/1968 0:00
6	1	Richard	Steele	Female	3/1/1982 0:00
7	1	Jeffrey	Bauer	Male	6/21/1967 0:00
8	1	Jennifer	Campbell	Male	4/27/1970 0:00
9	1	Elizabeth	Greene	Female	8/29/1972 0:00
10	1	Jared	Rodgers	Male	11/21/1989 0:00

CONTRACT

Contract_id	Employee_id	Hire_date	End_date	Salary	Job_type
1	1	9/27/2021		16124	Receptionist (Part-time)
2	1	7/28/2022	11/13/2024	13831	Receptionist (Part-time)
3	1	10/7/2021		15739	Receptionist (Part-time)
4	1	2/6/2025	2/18/2025	13454	Receptionist (Part-time)
5	2	1/22/2025		32043	Chef
6	2	3/30/2022	12/29/2022	41316	Chef
7	2	11/14/2020		38911	Chef
8	2	2/26/2021		42334	Chef
9	3	11/14/2023		28180	Receptionist
10	3	4/18/2022		23178	Receptionist
11	3	2/8/2022	6/13/2024	31922	Receptionist

REVIEW

Review_id	Booking_id	Satisfaction_score
1	985	8
2	68	6
3	733	4
4	892	6
5	1681	10
6	421	4
7	1210	10
8	1398	7
9	1356	1
10	84	2

Appendix C: SQL Queries for Branch Performance Comparison

Revenue Analysis

```
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

# Correct way to execute SQL query in Colab
query = """

SELECT
    s.Location_city AS Branch,
    strftime('%Y', b.Booking_date) AS Year,  -- Extracts only the
year
    SUM(p.Base_charge + COALESCE(p.total_Additional_charges, 0)) AS
Total_Revenue
FROM HOTEL h
JOIN SITE s ON h.Site_id = s.Site_id
JOIN ROOM r ON h.Hotel_ID = r.Hotel_ID
JOIN BOOKING b ON r.Room_id = b.Room_id
LEFT JOIN PAYMENT p ON b.Booking_id = p.Booking_id
GROUP BY Branch, Year
ORDER BY Year DESC, Total_Revenue DESC;

"""

# Execute the query
cursor.execute(query)

# Fetch and print results
results = cursor.fetchall()
for row in results:
    print(row)

# Close connection
conn.close()
```

Average Room Rates

```
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

query = """
SELECT
    s.Location_city,
    AVG(r.Room_rate) AS Average_Price_Per_Room
FROM SITE s
JOIN HOTEL h ON s.Site_id = h.Site_id
JOIN ROOM r ON h.Hotel_id = r.Hotel_id
GROUP BY s.Location_city;
"""

cursor.execute(query)
results = cursor.fetchall()

for row in results:
    print(row)

conn.close()
```

Appendix D: SQL Queries for Employee Performance

Average Salary Across Departments

```
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

query = """
SELECT
    c.Job_type,
    ROUND(AVG(c.Salary), 0) AS Average_Salary
FROM CONTRACT c
GROUP BY c.Job_type
ORDER BY Average_Salary DESC;
"""

cursor.execute(query)
results = cursor.fetchall()

df = pd.DataFrame(results, columns=['Job Type', 'Average Salary'])
print(df)

conn.close()
```

Active Employees Across Branches

```
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

query = """
SELECT
    s.Location_city AS Branch,
    COUNT(e.Employee_id) AS Active_Employees
FROM EMPLOYEE e
JOIN HOTEL h ON e.Hotel_id = h.Hotel_id
JOIN SITE s ON h.Site_id = s.Site_id
JOIN CONTRACT c ON e.Employee_id = c.Employee_id
WHERE c.End_date IS NULL
GROUP BY Branch;
"""

cursor.execute(query)
results = cursor.fetchall()

for row in results:
    print(row)

conn.close()
```

Salary Comparison in London and Southampton Based on Job Type

```
import sqlite3

conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

# Query to compare salaries in London and Southampton based on job type
query = """
SELECT
    c.Job_type,
    s.Location_city,
    AVG(c.Salary) AS Average_Salary
FROM CONTRACT c
JOIN EMPLOYEE e ON c.Employee_id = e.Employee_id
JOIN HOTEL h ON e.Hotel_id = h.Hotel_id
JOIN SITE s ON h.Site_id = s.Site_id
WHERE s.Location_city IN ('London', 'Southampton')
GROUP BY c.Job_type, s.Location_city
ORDER BY c.Job_type, s.Location_city;
"""

cursor.execute(query)
results = cursor.fetchall()

# Create a Pandas DataFrame for better visualization
df = pd.DataFrame(results, columns=['Job Type', 'Location', 'Average Salary'])

# Pivot the table for easier comparison
pivot_df = df.pivot(index='Job Type', columns='Location', values='Average Salary')

print(pivot_df)

conn.close()
```

Employee Turnover Rate

```
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

query_turnover = """
SELECT
    s.Location_city AS Hotel_Location,
    COUNT(DISTINCT CASE WHEN c.End_date IS NOT NULL THEN
e.Employee_id END) AS Employee_Turnover,
    COUNT(DISTINCT CASE WHEN c.End_date IS NOT NULL THEN
e.Employee_id END) * 100.0 /
    COUNT(DISTINCT e.Employee_id) AS Turnover_Rate
FROM EMPLOYEE e
JOIN HOTEL h ON e.Hotel_id = h.Hotel_ID
JOIN SITE s ON h.Site_id = s.Site_id
JOIN CONTRACT c ON e.Employee_id = c.Employee_id
GROUP BY Hotel_Location
ORDER BY Employee_Turnover DESC;

"""

# Execute query
cursor.execute(query_turnover)
results = cursor.fetchall()

# Convert results to DataFrame
df_turnover = pd.DataFrame(results, columns=['Hotel Location',
'Employee Turnover', 'Turnover Rate (%)'])
print(df_turnover)
# Close the connection
conn.close()
```

Appendix E: SQL Queries for Customer Satisfaction and Loyalty

Customer Satisfaction Score

```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

# SQL query to calculate the average guest review score in each
location
query = """
SELECT
    s.Location_city,
    AVG(r.Satisfaction_score) AS Average_Review_Score
FROM REVIEW r
JOIN BOOKING b ON r.Booking_id = b.Booking_id
JOIN ROOM ro ON b.Room_id = ro.Room_id
JOIN HOTEL h ON ro.Hotel_id = h.Hotel_ID
JOIN SITE s ON h.Site_id = s.Site_id
GROUP BY s.Location_city;
"""

# Execute the query
cursor.execute(query)

# Fetch and print the results
results = cursor.fetchall()
for row in results:
  print(row)

# Close the connection
conn.close()
```

Customer Satisfaction Score by Membership

```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect("hotel_db.sqlite")
cursor = conn.cursor()

# SQL query to calculate average customer satisfaction score by
membership
query = """
SELECT
    c.Membership,
    AVG(r.Satisfaction_score) AS Average_Satisfaction_Score
FROM CUSTOMER c
JOIN BOOKING b ON c.Customer_id = b.Customer_id
JOIN REVIEW r ON b.Booking_id = r.Booking_id
GROUP BY c.Membership;
"""

# Execute the query
cursor.execute(query)

# Fetch the results into a Pandas DataFrame
results = cursor.fetchall()
df = pd.DataFrame(results, columns=['Membership',
'Average_Satisfaction_Score'])

# Print the DataFrame
print(df)

# Close the connection
conn.close()
```