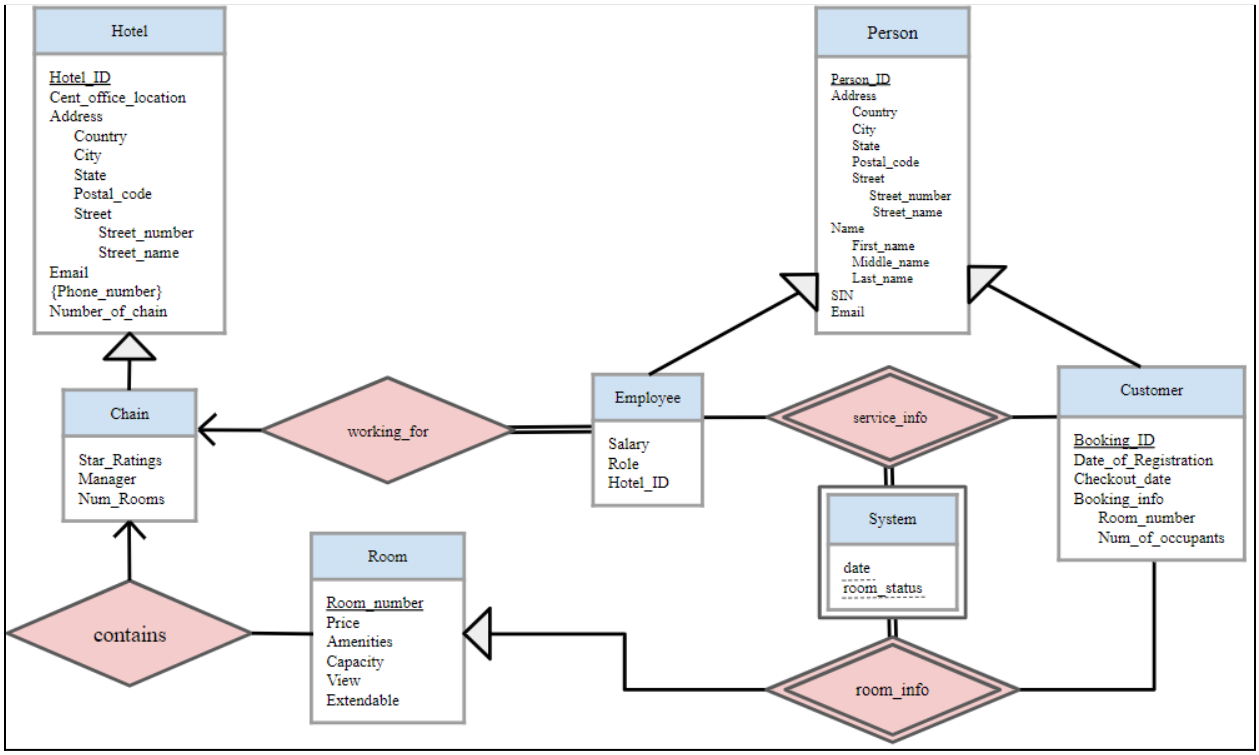
Justin Chun 300131838
Ziyad Gaffar 300148116
Siana Kong 300110500

Deliverable 2 Report | B02 G43

PART 4: Implementation of the Database



As shown above, our group initially had Employee and Customer connected to Person, in order to handle all general data (name, address, etc.), and give specific attributes depending on the role.

As we started to implement the ER diagram in postgresql, we have made some significant changes to fix the errors that we previously had.

- The first thing we did is that we changed the names for the Chain and Hotel table, so now Chain table is the parent of the Hotel table.
- We have changed the relation between customer and room so it can be one to many.
- We have added a primary key for the Hotel Table (Previously Chain table in deliverable 1) named hotel_id so that the strong entity between Hotel and Chain stands.

For the System Table, there isn't any kind of Primary key mainly because it is a weak entity, so it doesn't need a Primary key. We have also added a new attribute in the System table called room_status, where there are only three possible status which are {'Avail', 'Book', 'Rent'}. For the date attribute, we have divided it into two attributes called start_date and end_date.

We have also removed the Person table, as it was much harder to implement, even though it seemed right theoretically. So Customer Table has a Primary Key booking_id that is special to them, and for Employee Table, their Primary Key is employee_id, an attribute that we have added during this deliverable. Both Tables contain the Address attributes, name attributes, SIN and Phone number attributes. However only Employee Table has an email address attribute.

Justin Chun 300131838
Ziyad Gaffar 300148116
Siana Kong 300110500

For Chain and Hotel Table, we have created four new tables called hotelEmails, hotelPhones, ChainEmails and ChainPhones which stores the phone numbers and email addresses of Hotel and Chains.

Finally, for the relations, we have only implemented the “service_info” relation since it's an identifying relationship between employees, system and customer.

PART 5: Queries and Triggers

Query 1)

Number of occupants: greater than or equal to 1, less than or equal to room capacity

```
ALTER TABLE System
ADD CONSTRAINT occupants_cap_check (CHECK (number_of_occupants >= 1 AND
number_of_occupants <= capacity);
```

Query 2)

Star rating: 1 to 5

```
ALTER TABLE Hotel
ADD CONSTRAINT rating_check CHECK (star_ratings > 0 AND star_ratings < 6);
```

Trigger 2)

When the room becomes available from rented or booked status, all the customer and room information for that room should become Null or default value.

If Room_status = ‘Avail’: Start_date, End_date, Number_of_occupants, Booking_ID set to null

```
CREATE FUNCTION check_room_status()
      RETURNS trigger AS
$BODY$
BEGIN
UPDATE System
SET Start_date = '2021-01-01',
end_date = '2021-12-31',
Number_of_occupants = NULL,
Booking_ID = NULL,
paid = FALSE
WHERE room_id = OLD.room_id;
RETURN NEW;
END
$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER room_status_check
```

```
AFTER UPDATE OF room_status ON System
FOR EACH ROW
WHEN (NEW.room_status = 'Avail')
EXECUTE PROCEDURE check_room_status();
```

Trigger 2)

We decided that we should only allow the employees to turn the available or booked rooms to rented rooms only if the customer has paid for the room successfully.

Only If paid = true, book or avail can turn into rent

```
CREATE FUNCTION check_room_paid()
      RETURNS trigger AS
$BODY$
BEGIN
IF NEW.room_status IS 'Rent' AND NEW.paid IS FALSE THEN
      RAISE EXCEPTION 'Customer must have paid for the room to rent';
ELSE
      RETURN NEW;
END IF;
END
$BODY$ LANGUAGE plpgsql;

CREATE TRIGGER paid_rent_check
BEFORE INSERT OR UPDATE ON System
FOR EACH ROW
EXECUTE PROCEDURE check_room_paid();
```

Other constraints

Constraints we added to the database other than the 2 queries we provided are:

- Phone number should be 10 digits and cannot start with 0
- SIN number should be 9 digits and cannot start with 0
- Room view can only be either ‘Sea’ or ‘Mountain’
- Room price and capacity has to be greater than 0
- Star rating can only be an integer from 1 to 5
- Email format should be '%_@_%._%'
- Hotels must have at least 5 rooms
- Employee salary must be greater than 0
- Employee role can only be either ‘Manager’, ‘Concierge’ or ‘Housekeeper’
- Room status can only be either ‘Avail’, ‘Book’, or ‘Rent’
- Number of occupants should be greater than or equal to 1 and less than or equal to capacity

The provided constraints are based on the domain integrity, entity integrity, and user-defined constraints we defined in deliverable 1. However, some of those constraints have been changed or omitted as we had to adjust to the capabilities and limitations of the database.

Refriential key constraints

We added the referential key constraints based on the referential integrities we defined in deliverable 1. We implemented these constraints using the following pattern for all foreign key constraints:

```
ALTER TABLE System
ADD CONSTRAINT system_employee_id_fkey FOREIGN KEY (employee_id)
REFERENCES employee(employee_id) MATCH SIMPLE ON UPDATE CASCADE ON DELETE CASCADE
```

PART 6: Populating the Database

The queries used for populating the database are in the attached file: **data.pdf**
The specific data might not match the current data as we have altered and updated datas.

PART 7: Application and SQL codes

Programming language used: Java
DBMS: PostgreSQL
We decided that it is best for us to not provide our credentials in the code for security reasons, as there are crucial informations such as the SIN number that can be accessed using our university accounts. The way of installment is therefore affected by this decision. We have granted access to all TAs on the syllabus, but if you still do not have access to our DB, please let us know. Also, please make sure to have all requirements satisfied to run JDBC library(instruction in lab5)

DB Administrator

How to install: Compile and run **Admin.java** in the command line or any IDE that you are using to run the app. You will need to enter your pgAdmin ID and password to run the rest of the app.

This program allows the administrator to input queries through the app in order to insert, update, or delete the values inputted to the tables in the Hotel Management database. Also checks if an error has occurred or not after the admin submits the query. We created this application so that whenever a change in record is necessary, an admin can directly interact with the datas. We considered that there may be a situation like cancellation of booking or cancellation of the customer or employee's registration.

Employee

How to install: Compile and run **Employee.java** in the command line or any IDE that you are using to run the app. You will need to enter your pgAdmin ID and password to run the rest of the app.

The DB hotel app for employees allows the employees to log in with their SIN number.

```
SELECT e.name, c.hotel_name, h.city
FROM "Project".em
```

Once the login is successfully done, the app welcomes the user with their name and showing their employee number. It also saves the name of the hotel chain that the employee works at, as well as the city of the hotel location.

```
SELECT e.name, c.hotel_name, h.city
FROM employee e, chain c, hotel h
WHERE employee_id = ?
AND e.hotel_id = h.hotel_id
AND h.chain_id = c.chain_id
```

The employee is given two options to view the rooms: view available rooms or view booked rooms. For viewing the available rooms:

```
SELECT s.room_id, r.room_number, r.price, r.view, r.extendable, r.capacity,
r.amenities
FROM employee E, Hotel H, Room R, Chain C, System S
WHERE e.employee_id = ?
AND H.hotel_id = E.hotel_id AND R.hotel_id = H.hotel_id AND S.room_id = R.room_id
AND C.chain_id = H.chain_id AND S.room_status = 'Avail'
ORDER BY r.room_number
```

For viewing the booked rooms:

```
SELECT s.room_id, r.room_number, r.price, r.view, r.extendable, r.capacity,
r.amenities
FROM employee E, Hotel H, Room R, Chain C, System S
WHERE e.employee_id = ?
AND H.hotel_id = E.hotel_id AND R.hotel_id = H.hotel_id AND S.room_id = R.room_id
AND C.chain_id = H.chain_id AND S.room_status = 'Book'
ORDER BY R.room_number
```

When viewing the rooms that are available, the employee needs to fill in the information such as number of occupants and the booking ID of a customer. This step is omitted for viewing the booked rooms as these informations should be already filled in by the customer when booking. Instead, the customer information can be retrieved from using the room ID of the room that is booked.

```
SELECT capacity FROM system WHERE room_id = ?
SELECT name FROM customer WHERE booking_id = ?

SELECT c.name
FROM customer c, system s
WHERE s.room_id = ?
```

```
AND c.booking_id = s.booking_id
```

Then the app prompts the user if the customer who is renting the room has paid for the room, and the employee can either reply yes or no to change the paid status of the customer. If the answer is no, the program is terminated as the customer must have paid for the room to rent it. If the answer is yes the program updates the Paid boolean in the database to TRUE.

```
UPDATE system SET paid = TRUE WHERE room_id = ?
```

Afterwards, if the employee is turning an available room to a booked room, it takes another extra step to update the booking ID, the check in and check out dates, and the number of occupants for the selected room. These information are all ready at this stage.

```
UPDATE system SET booking_id = ?, start_date = ?, end_date = ?,
number_of_occupants = ? WHERE room_id = ?
```

Finally, the app updates the room status to ‘Rent’ for the selected room. The room number of the selected room is retrieved from the database to finally display a message that the rent process has been completed for the specific room number.

```
UPDATE system SET employee_id = ? room_status = 'Rent' WHERE room_id=?
SELECT room_number FROM room WHERE room_id=?
```

Customer

How to install: Compile and run **Customer.java** in the command line or any IDE that you are using to run the app. You will need to enter your pgAdmin ID and password to run the rest of the app.

This application allows customers to search for hotels either by the name of the hotel chain or by the location. It also allows the customer to register to the system after checking whether the customer is registered or not.

If registered, it asks for the phone number of the customer, then displays the booking ID.

```
SELECT booking_id FROM customer WHERE phone_number = ?
SELECT name FROM customer WHERE booking_id = booking_id
```

Registration is done with the following query, where the first query gives a new booking ID to a new customer.

```
SELECT MAX(booking_id) FROM Customer
INSERT INTO "Project".customer (Booking_ID, Date_of_registration, Country, City,
State, Postal_code, Street_number, Street_name, name, SIN, Phone_number)
VALUES "(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)")
```

Once the registration is complete, the application asks the customer to decide whether to choose by the name of the chain or by location. For the name of the chain, it uses the following query to display the list of chains to the customer:

```
SELECT hotel_name, Country FROM "Project".chain
```

Once the customer chooses a chain, it displays a list of hotels accordingly to the name of the chain.

```
SELECT DISTINCT C.hotel_name, H.city, R.room_number, R.price, R.view,
R.extendable, R.capacity, R.amenities
FROM "Project".chain C, "Project".hotel H, "Project".system S, "Project".room R
WHERE S.hotel_id = H.hotel_id AND S.hotel_id = R.hotel_id AND C.hotel_name = ?
AND C.chain_id = H.chain_id AND S.room_id = R.room_id AND S.room_status = 'Avail'
ORDER BY H.city
```

Afterwards, the app asks the customer to choose a city and ask for information on that specific room desired (room number, check-in/out date, number of occupants, etc.).

```
UPDATE System
SET room_status = 'Book', start_date = ?, end_date = ?, booking_id = ?,
number_of_occupants = ?
FROM (SELECT R.room_id AS id FROM Room R, Hotel H, Chain C WHERE H.city = ? AND
C.hotel_name = ? AND H.hotel_id = R.hotel_id AND H.chain_id = C.chain_id AND
R.room_number = ?) AS B WHERE room_id = B.id
```

For another case when the customer chooses to search by location, the following query is executed to display the datas to the customer.

```
SELECT DISTINCT country, city FROM hotel
```

After the customer chooses a city, the following query is executed:

```
SELECT DISTINCT C.hotel_name, R.room_number, R.price, R.view, R.extendable,
R.capacity, R.Amenities
FROM hotel H, chain C, System S, Room R
WHERE S.hotel_id=H.hotel_id AND S.hotel_id=R.hotel_id AND H.chain_id=C.chain_id
AND H.city=? AND S.room_status='Avail' AND S.room_id = R.room_id ORDER BY
c.hotel_name
```

After the user inputs all the necessary information to book a room, the following query is executed to update the table accordingly.

```
UPDATE System
SET room_status = 'Book', start_date = ?, end_date = ?, booking_id = ?,
number_of_occupants = ?
```

```
FROM (SELECT R.room_id AS id FROM Room R, Hotel H, Chain C WHERE H.city = ? AND
C.hotel_name = ? AND H.hotel_id = R.hotel_id AND H.chain_id = C.chain_id AND
R.room_number = ?) AS B WHERE room_id = B.id
```

Error Handling

We have implemented some error handling for inputs in our Java code, such that the program would prompt the user until they put in a valid input. For example, when asking to enter a city from the list of cities of different hotels, the user will be prompted to re-enter the city name if they put in a city name that is not from the list. This is applied to different inputs such as entering a room number, SIN, phone number, and etc. However, we did not add error handling for cases where the user inputs wrong data types, such as the user entering a string when asked to enter a room number, which should be an integer.

Customer

For checking customer's phone number:

```
SELECT booking_id FROM customer WHERE phone_number = ?
```

For checking the room number:

```
SELECT R.room_number FROM room R, chain C, Hotel H, System S
WHERE R.room_id = S.room_id AND H.Chain_id = C.Chain_id AND C.Hotel_name = ?
AND S.hotel_id=H.hotel_id AND S.hotel_id=R.hotel_id AND H.city = ?
AND S.room_status='Avail';
```

For number of occupants:

```
SELECT R.capacity
FROM room R, hotel H, chain C
WHERE R.room_number=? AND H.city=? AND C.hotel_name=? AND C.chain_id = H.chain_id
AND H.hotel_id = R.hotel_id
```

The three provided java files above are all the code necessary to implement all the user interfaces

Justin Chun 300131838
Ziyad Gaffar 300148116
Siana Kong 300110500

PART 8: Queries and results

You might not get the same results when running the queries as we have altered and updated the tables for testing the app and queries after completing this part.

Query 1)

```
SELECT C.name, R.capacity, R.price, S.start_date, R.view, Ch.hotel_name
FROM Customer C, Room R, System S, Chain Ch, Hotel H
WHERE S.room_status = 'Rent'
AND S.booking_id = C.booking_id
AND S.room_id = R.room_id
AND R.hotel_id = H.hotel_id
AND H.chain_id = Ch.chain_id
ORDER BY R.price ASC, S.start_date DESC;
```

	name character varying (2	capacity integer	price numeric	start_date date	view character varying (	hotel_name character varying (20)
1	Samuel James	1	56.29	2021-06-21	Mountain	Sandman
2	Ziyad Gaffar	1	58.12	2021-04-01	Mountain	Sandman
3	Barack Obama	2	82.35	2021-04-01	Mountain	Sandman
4	James American	4	92.75	2021-03-31	Mountain	Sandman
5	Senna Lux	2	130.29	2021-03-28	Mountain	Hilton
6	Andrea Bellerin	2	215.99	2021-04-02	Mountain	Les Suites
7	Alex Panarin	2	220.96	2021-04-02	Mountain	Marriott
8	Mark Stone	1	242.30	2021-04-01	Sea	Marriott
9	Ben Tina	5	290.99	2021-04-05	Mountain	Fairmont
10	Leo Polo	6	350.35	2021-04-04	Sea	Fairmont
11	Lucas Lopez	4	399.99	2021-04-01	Mountain	Les Suites
12	Eugene Lee	5	412.21	2021-04-03	Mountain	Andaz
13	Xavier Louca	6	434.92	2021-04-04	Sea	Andaz
14	Huston Posner	4	440.99	2021-04-08	Sea	Fairmont
15	Barack Obama	6	444.44	2021-04-06	Mountain	Andaz
16	Natalie Ming	3	445.78	2021-05-24	Sea	Andaz

Room type in the question is treated as capacity; we represented the single, double, etc as the maximum number of customers allowed in the room instead.

Query 2)

```
CREATE VIEW CustomerListView AS
	SELECT DISTINCT C.*, Ch.hotel_name
	FROM Customer C, Room R, System S, Chain Ch, Hotel H
	WHERE S.booking_id = C.booking_id
	AND S.hotel_id = H.hotel_id
	AND H.chain_id = Ch.chain_id
	ORDER BY Ch.hotel_name;
```

Justin Chun 300131838
Ziyad Gaffar 300148116
Siana Kong 300110500

	bookir integ	date_of_registrat date	country character varying (2	city character varying (	state character varying	postal_cod character v	street_nur integer	street_name character varying (25	sin integer	phone_number bigint	name character varying (25	hotel_name character v
1	7	2021-01-02	Canada	Oakville	Ontario	C8O5O1	7823	Kerr Street	147384924	6478782984	Taliah Koolo	Andaz
2	10	2021-03-28	United States	Seattle	Washington	J5S8E3	77	King Street	837482938	4348726374	Luke King	Andaz
3	26	2021-01-01	United States	Washington DC	Washington	W4S2H4	1600	Pennsylvania Aven...	574829031	7948561735	Barack Obama	Andaz
4	9	2021-03-14	United States	San Francisco	California	Z8L2D7	453	Xander Street	473627473	3842348756	Xavier Louca	Andaz
5	8	2021-02-22	Canada	Ottawa	Ontario	M9Z7X2	1782	Prada Road	485732845	6047483623	Natalie Ming	Andaz
6	6	2021-01-05	United States	Los Angeles	California	S9L2C1	563	Terra Road	857465738	2835647567	Eugene Lee	Andaz
7	1	2021-01-21	Canada	Toronto	Ontario	L3J1A2	423	Lake Street	348291348	4738295647	James Smith	Fairmont
8	4	2021-03-25	United States	Houston	Texas	U8H2T5	8721	Nova Blvd	389473293	4825847261	Lily James	Fairmont
9	2	2021-02-23	United States	Chicago	Illinois	U1C9L3	6743	Pondo Street	384716593	4264857384	Ben Tina	Fairmont
10	3	2021-01-15	Canada	Burlington	Ontario	C9B2O7	1432	Appleby Road	758493847	6472837462	Leo Polo	Fairmont
11	5	2021-03-11	Canada	Hamilton	Ontario	C8H8S4	24	Postman Street	958473612	9056784958	Huston Posner	Fairmont
12	24	2021-02-12	Canada	Mississauga	Ontario	M2R5F1	340	Weird Street	954023465	2928563522	Abdul Abde	Hilton
13	22	2021-01-27	Canada	Toronto	Ontario	Q2I7O1	7632	St Clair Avenue	345212849	3456102854	Marcos Cav	Hilton
14	25	2020-12-01	United States	Seattle	Washington	W4S1N5	9900	Some Street	758365323	1948735261	James Rohan	Hilton
15	21	2020-12-22	Canada	Toronto	Ontario	M2D8W2	342	Yonge Street	321364565	2347436108	Jennifer Shaq	Hilton
16	23	2021-03-01	United States	Manhattan	New York	N3W7O4	23	Main Street	324886909	2342758463	Senna Lux	Hilton
17	12	2020-12-31	United States	New Mexico	Santa Fe	K2X3E0	120	Cordova St	431975023	5059231121	Lucas Lopez	Les Suites
18	15	2021-02-28	United States	California	San Jose	P3OZ0M	102	Williams Road	426313231	2102421242	Marie-Ann Methot	Les Suites
19	11	2021-03-16	Canada	Quebec	Gatineau	J8Z1T9	41	Plateau Blvd	901242125	8193025231	Danny Moura	Les Suites
20	13	2021-01-28	Canada	British Columbia	Kelowna	V1P0D3	420	Water St	920124210	9302134201	Martin Daccord	Les Suites
21	14	2020-10-31	Mexico	Mexico City	Mexico City	W3D0B1	910	Tripoli	312052352	1920925501	Andrea Bellerin	Les Suites
22	16	2021-01-02	Canada	Ontario	Ottawa	K1W2S4	12	Elgin St	242502963	6130294444	Mark Stone	Marriott
23	17	2020-11-09	Canada	Ontario	Toronto	M2Z0D5	453	Spadina Rd	452134221	2222222222	Joey Pageau	Marriott
24	18	2021-03-14	United States	Minnesota	St. Paul	A2B3C6	103	Front Ave	531421999	2052210000	Danny Heatly	Marriott
25	19	2021-04-01	Canada	Ontario	London	S3T0S2	302	Oxford St W	999109299	3503212142	Omer Abubakar	Marriott
26	20	2020-12-20	United States	District of Colu...	Washington D.C	L2Z0D3	4324	Florida Ave NW	999999999	3012013201	Alex Panarin	Marriott
27	27	2020-11-30	Canada	Toronto	Ontario	M1A0T2	3320	Bloor Street	493029321	8909300221	Jeremy Bucktee	Sandman
28	26	2021-01-01	United States	Washington DC	Washington	W4S2H4	1600	Pennsylvania Aven...	574829031	7948561735	Barack Obama	Sandman
29	29	2021-01-22	Canada	Ottawa	Ontario	K3P0S2	430	Bank Street	493029462	2093256483	Ziyad Gaffar	Sandman
30	28	2021-01-30	United States	Chicago	Illinois	S3H7S1	432	St Alban Road	340209324	9035467284	James American	Sandman
31	30	2021-02-11	Canada	London	Ontario	L0N6O7	212	St Patrick Street	320140339	9020156293	Samuel James	Sandman

Query 3)

```
SELECT R.*
FROM Room R, (SELECT MIN(R.price) AS minP FROM Room R) AS B
WHERE R.price = B.minP;
```

	room_number integer	price numeric	amenities character varying (23	capacity integer	view character	extendable boolean	hotel_id integer	room_id [PK] integer
1	301	53.23	air condition, fridge	1	Sea	false	26	128

Query 4)

```
SELECT C.hotel_name, R.*, H.star_ratings
FROM Room R, Hotel H, Chain C
WHERE H.city = 'Ottawa'
AND R.hotel_id = H.hotel_id
AND C.chain_id = H.chain_id
ORDER BY H.star_ratings, R.price;
```

	hotel_name character varying	room_number integer	price numeric	amenities character varying (235)	capacity integer	view character varyi	extendable boolean	hotel_id integer	room_id integer	star_ratings integer
1	Sandman	501	73.32	air condition, fridge	2	Mountain	false	27	135	4
2	Sandman	401	76.31	air condition, fridge	2	Mountain	false	27	134	4
3	Sandman	101	82.57	TV, air condition, fridge	3	Sea	false	27	131	4
4	Sandman	201	87.23	TV, air condition, fridge	3	Mountain	false	27	132	4
5	Sandman	301	98.11	TV, air condition, fridge, micro...	3	Sea	true	27	133	4
6	Marriott	301	242.30	TV, air condition, fridge	1	Sea	false	16	78	5
7	Fairmont	501	269.69	TV, air condition, fridge	3	Mountain	false	1	5	5
8	Marriott	201	280.29	TV, air condition, fridge	2	Sea	false	16	77	5
9	Marriott	401	288.89	TV, air condition, fridge, wine, ...	4	Mountain	true	16	79	5
10	Fairmont	301	299.99	TV, air condition, fridge	4	Mountain	false	1	3	5
11	Marriott	501	300.12	TV, air condition, fridge, wine, ...	5	Sea	true	16	80	5
12	Fairmont	201	309.99	TV, air condition, fridge, wine, ...	4	Sea	true	1	2	5
13	Marriott	101	345.12	TV, air condition, fridge, wine, ...	6	Mountain	true	16	76	5
14	Fairmont	401	399.99	TV, air condition, fridge, wine, ...	3	Sea	true	1	4	5
15	Fairmont	101	430.24	TV, air condition, fridge, wine, ...	5	Mountain	true	1	1	5

Query 5)

```
SELECT R.*, S.start_date, S.end_date
FROM Room R, System S
WHERE S.room_status = 'Rent'
AND S.room_id = R.room_id
AND S.start_date <= '2021-04-10'
AND S.end_date >= '2021-04-10';
```

	room_number integer	price numeric	amenities character varying (235)	capacity integer	view character varyir	extendable boolean	hotel_id integer	room_id integer	start_date date	end_date date
1	401	440.99	TV, air condition, fridge, win...	4	Sea	true	5	24	2021-04-08	2021-04-10
2	101	444.44	TV, air condition, fridge, win...	6	Mountain	true	6	26	2021-04-06	2021-04-11
3	501	399.99	TV, air condition, fridge, win...	4	Mountain	false	12	60	2021-04-01	2021-04-14
4	301	242.30	TV, air condition, fridge	1	Sea	false	16	78	2021-04-01	2021-04-10
5	301	220.96	TV, air condition, fridge	2	Mountain	true	20	98	2021-04-02	2021-04-21
6	501	130.29	TV, air condition, fridge	2	Mountain	true	23	115	2021-03-28	2021-04-10
7	501	82.35	TV, air condition, fridge, mic...	2	Mountain	true	26	130	2021-04-01	2021-04-21
8	301	92.75	TV, air condition, fridge, mic...	4	Mountain	true	28	138	2021-03-31	2021-04-19

Selected month: April 2021

Query 6)

```
UPDATE Customer
SET phone_number = 4738295647
WHERE booking_id = 1;
```

	booking_id [PK] integer	phone_number bigint
1	1	4738295647

Query for viewing the result:

```
SELECT booking_id, phone_number
FROM customer
WHERE booking_id = 1;
```

Justin Chun 300131838
Ziyad Gaffar 300148116
Siana Kong 300110500

Query 7)

4 Star category hotels are most preferred by the customers.

Query for viewing the result:

```
SELECT COUNT(H.hotel_id), H.star_ratings
FROM Hotel H, System S
WHERE (S.room_status = 'Book' OR S.room_status
'Rent')
AND S.hotel_id = H.hotel_id
GROUP BY H.star_ratings
ORDER BY COUNT(H.hotel_id) DESC;
```

"Most preferred by the customers" is defined by most number of rooms that are booked or rented from the hotels with each star rating.

	count bigint	star_ratings integer
1	11	4
2	8	3
3	8	5
4	4	2
5	1	1

=

Query 8)

```
SELECT MAX(E.salary) AS Salary
FROM Employee E
WHERE E.salary NOT IN
(SELECT MAX(E.salary)
FROM Employee E);
```

	salary numeric
1	93021.24