CSI 2132 Databases I, Winter 2023
Professor: Verena Kantere

Hotel Assignment Part II
March 31, 2023

Group 33
Steven Wilson - 300220675
Edgar Acosta - 300246710

Part I: Queries

Query 1:

```sql
SELECT hotel.hotel_id,hotel.owner_name,star_rating, country
FROM hotel_addresses,hotel
WHERE star_rating >= 4 AND country = 'Canada';
```

The following query takes in the hotel_id, the owner_name, the star_rating, and country from hotel_addresses, and hotel as all these keys can be found in either table or both such that one can obtain the star ratings of each hotel in Canada

Query 2:

```sql
SELECT room.hotel_id,owner_name,room_number, price
FROM room, hotel
WHERE price <= 200;
```

The following query takes in the hotel_id, the owner_name, room_number, and the price from room, and hotels as all these keys can be found in either table or both such that the price of the hotel room is less than or equal to 200.

Query 3:

```sql
SELECT hotel_id, room_number, outdoor_view
FROM room
WHERE outdoor_view = 'Mountain';
```

The following query takes in the hotel_id, room_number, and the outdoor_view of room such that the outdoor view of the room is equal to mountain.

Query 4:

```sql
SELECT room.hotel_id,room_number,owner_name, problems
FROM room, hotel
WHERE problems = '{"none"}' AND owner_name = 'Goodnite Hostels';
```

The following query takes in the hotel_id, owner_name, and problems from room, and hotel as all these keys can be found in either table or both such that the one can obtain the hotels and its chain owner where there are no problems.

Part II: Views

View 1: Areas

```sql
CREATE VIEW rooms_in_usa AS
    SELECT
        COUNT(DISTINCT room.room_id) AS num_rooms
    FROM room
    JOIN hotel_addresses ON room.hotel_id = hotel_addresses.hotel_id
    WHERE
        hotel_addresses.country = 'USA' AND room.isAvailable = 'yes';

CREATE VIEW rooms_in_canada AS
    SELECT
        COUNT(DISTINCT room.room_id) AS num_rooms
    FROM room
    JOIN hotel_addresses ON room.hotel_id = hotel_addresses.hotel_id
    WHERE
        hotel_addresses.country = 'Canada' AND room.isAvailable = 'yes';
```

The following view(s) counts the available hotel rooms per area and displays them. As a group, it has been established that each country is an area. For some reason there is a syntax error on CREATE, we could not figure out as to why.

View 2: Room capacity

```sql
CREATE VIEW capacity_in_rooms AS
    SELECT hotel_id,room_number,capacity
    FROM room;
```

The following view displays the capacity in each room in a given hotel.

Part III: Indexes

Index I:

```sql
CREATE INDEX hotel_rating ON hotel(star_rating);
CREATE INDEX price_hotel_room_availability ON room(isAvailable) INCLUDE
(price);
```

This index delivers the price, star rating, and availability of a hotel room.

Index II:

```sql
CREATE UNIQUE INDEX customer_booking ON
booking(customer_id,booking_id);
```

This index delivers the bookings of each customer, the third index is used in one of the triggers

Part IV: Triggers

Trigger 1: Assigning employees to a manager

```sql
CREATE OR REPLACE FUNCTION if_manager()
    RETURNS TRIGGER
    AS
$$
BEGIN
    IF emp_role <> 'manager' THEN
        INSERT INTO employee(manager_id)
        VALUES (NEW.emp_id);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER manager_event
AFTER UPDATE ON employee
FOR EACH ROW
EXECUTE FUNCTION if_manager();
```

This trigger inputs the emp_id of the manager for each non-manager employee

Trigger 2: Changing the availability of a room

```
CREATE OR REPLACE FUNCTION change_availability() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.room_id <> NULL AND NEW.customer_id <> NULL THEN
        UPDATE room SET isAvailable = 'no';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER unavailable
AFTER INSERT OR UPDATE ON booking
FOR EACH ROW
EXECUTE FUNCTION change_availability();
```

The point of this trigger is when room bookings are being made, and the customer_id and the room_id is already filled, a booking is made and the specific room that was booked would no longer be available such that each room can be booked once.

Index 3:

```
CREATE UNIQUE INDEX rent_booking_key ON rent(booking_id);
```

This index is used to count unique booking_ids for each room, this index will be used in the trigger.

Trigger 3:

```
CREATE OR REPLACE FUNCTION update_rentals() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO rent(booking_id,room_number, rent_date, customer_id)
    VALUES(NEW.booking_id,NEW.room_number,NEW.book_date,NEW.customer_id)
    ON CONFLICT (booking_id, room_number) DO UPDATE
    SET rent_date = EXCLUDED.rent_date, customer_id =
EXCLUDED.customer_id;
    RETURN NEW;
END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER book_to_rent
AFTER INSERT OR UPDATE ON booking
```

```
FOR EACH ROW
EXECUTE FUNCTION update_rentals();
```

This trigger updates the rentals' table whenever the booking table is being updated.

Part V: Installation

- **See attached README.md**