**CSI 2132 - Deliverable 2 Report**


| | | |
|---|---|---|
| Oluwafeyisayo Adesanya | 300120992 | Group 30 |
| Emily Lu | 300114727 | Lab B03 |
| Nika Ribnitski | 300121606 | 01/04/2021 |


## The DBMS and Coding Languages We Used

To create this application, we used pgAdmin (which implements postgresql) and Java. The database holds all information relating to parent brands, hotels, rooms, customers, employees, renting, bookings, and archived information. It also contains the relationships between those listed entities. To ensure the integrity of our data, we implemented constraints on relevant fields. For example, there is a constraint on hotel's star rating that ensures each and every hotel has a rating between 1 and 5, inclusive. We also added triggers to take care of automatically transmitting booking and renting information to the archives. The two triggers make a copy of the information, so even if a booking or renting is deleted from the database it's trace will remain in the archives. Our user interface is a simple command-line interface created using Java. We made it user-friendly by adding plenty of instructions that guide the user and exit options to allow smooth navigation between its functionalities without needing to restart the program.

## Guide to Install and Run Our Program

1. download the java files
2. ensure you have access to the database
3. run *connection.java* in the system shell of your choice
4. Here are some sample SIN that will allow you to login to their respective accounts, you may also create a new account for Employees and Customers in Admin:
    a. Employee: 102938162
    b. Customer: 119537075

## SQL code and Java code for UI

Please view the attached files to see how we implemented the database and the user interface.

## Who Did What

All three of us collaborated at each stage of the project. Together, we were able to problem-solve connection issues, create the database, insert the demanded data, create a command line interface, and finally write this report. This was a great group experience!

- Nika, Emily, Feyi

**Code Answers for Questions 1-8**

**1. Give the details of all currently rented rooms in a specific hotel. Please display the columns as customer name, room type, room price, rental start date, room view, hotel chain.Sort by the room price in ascending order and rental start date in descending order.**

```
SELECT DISTINCT (c.first_name , c.middle_name , c.last_name) AS customer_name,
        r.capacity AS room_type,
        r.price AS room_price,
        t.arrival_date AS rental_start_date,
        r.view_type AS room_view,
        h.pbname
        FROM customer c, room r, renting t, hotel h
        WHERE h.hotel_id = r.hotel_id
        AND r.hotel_id = 22
        AND t.hotel_id = r.hotel_id
        AND c.sin = t.sin
        ORDER BY room_price ASC, rental_start_date DESC;
```

**2. Create a view named CustomerListView that gives the details of all the customers. Please, sort the customers by hotel chain.**

```
CREATE VIEW CustomerListView AS
        SELECT DISTINCT (c.*),  h.pbname
        FROM customer c, hotel h, booking b, renting r
        WHERE (c.sin = b.sin AND h.hotel_id = b.hotel_id)
        OR (c.sin = r.sin AND h.hotel_id = r.hotel_id)
        ORDER BY h.pbname;
```

**3. Display the details of the cheapest hotel room of all hotel chains.**

```
        SELECT * FROM room WHERE price in ( SELECT min(price) FROM room);
```

**4. List all the rooms in all hotels in Ottawa and sort them based on the hotel stars and price.**

```
        SELECT r.* FROM room r, hotel h
        WHERE r.hotel_id = h.hotel_id AND h.physical_address LIKE  '%Ottawa%'
        ORDER BY h.star_catagory,r.price;
```

**5. List all the details of all rooms rented on the 10th day of a month of your choice. Ensure to insert dates in your table that correspond to that month in order to run your query.**

```
SELECT r.* FROM room r, renting t
        WHERE r.room_num = t.room_num
         AND r.hotel_id = t.hotel_id
         AND EXTRACT(DAY FROM t.arrival_date) = 10
         AND EXTRACT(MONTH FROM t.arrival_date) = 03;
```

**6. Update the phone number of a customer.**

```
        UPDATE customer SET phone_number = 5763842346
        WHERE sin = 123456789;
```

**7. Which category hotels (1 star to 5 star) are most preferred by the customers?**

```
        SELECT star_catagory
        FROM hotel, booking, renting
        WHERE hotel.hotel_id = booking.hotel_id
        AND hotel.hotel_id = renting.hotel_id
        GROUP BY (hotel.star_catagory)
        ORDER BY COUNT(*) DESC;
```

**8. Find the second highest salary from the employee table.**

```
        SELECT MAX(salary) AS salary  FROM employee
        WHERE salary < (SELECT MAX(salary) FROM employee);
```

**Triggers Implemented**

To automatically add a copy of booking into booking archives at the moment of the booking's creation:

```
create function add_booking_to_archive()
    returns trigger as $BODY$
    begin
    insert into "public".booking_archives VALUES
        (new.booking_id, new.view_type, new.total_occupants, new.arrival_date,
         new.departure_date, new.duration_of_stay, new.room_num, new.sin);
    return NEW;

    end
    $BODY$ language plpgsql;

CREATE TRIGGER trigger_add_booking_to_archive
BEFORE INSERT ON booking
FOR EACH ROW
EXECUTE PROCEDURE add_booking_to_archive();
```

To automatically add a copy of renting into renting archives at the moment of the renting's creation:

```
create function add_renting_to_archive()
    returns trigger as $BODY$
    begin
    insert into "public".renting_archives VALUES
        (new.renting_id, new.arrival_date, new.departure_date, new.duration_of_stay, new.room_num, new.sin);
    return NEW;

    end
    $BODY$ language plpgsql;

CREATE TRIGGER trigger_add_renting_to_archive
BEFORE INSERT ON renting
FOR EACH ROW
EXECUTE PROCEDURE add_renting_to_archive();
```

To automatically delete a renting after it has been paid for:

```
create function delete_renting()
    returns trigger as $BODY$
    begin

    if new.paid_for = true then
        delete from renting where paid_for = true;
    end if;
    return new;
    end
    $BODY$ language plpgsql;

CREATE TRIGGER trigger_delete_renting
AFTER UPDATE ON renting
FOR ROW
EXECUTE PROCEDURE delete_renting();
```

**Queries**

This is an example of a query that inserts into booking.
Before the insert, *trigger_add_booking_to_archive* will add this new booking into the booking_archives table.

```
INSERT INTO booking VALUES
(13, 'mountain', 2, '2021-04-01', '2021-04-04', 3, 1, 21, 453778456);
```

This is an example of a query that inserts into renting. Before the insert, *trigger_add_renting_to_archive* will add this new renting into the renting_archives table.

```
INSERT INTO renting VALUES
(15, 702.87, false, '2021-04-01', '2021-04-04', 3, 1, 21, 453778456);
```