Summary of extra features of my database project

Data visualization

There are a total of three pages of which I implemented data visualization:

- Customer -> track spendings, where I used a bar chart to represent
- Agent -> view top customers, where I used 2 bar chart to represent
- Staff -> view reports, where I used 1 bar chart and 2 pie charts to represent

Security

1. Password (Login) security

I implemented this by hashing the password during the registration, so that the database server will only know the hashed password. Even when the data was leaked, the actual password will still be secured.

When logging in, we apply the same hashing algorithm so that the server may verify whether the password is correct without knowing what the actual password is.

Note: I generated a few accounts with passwords of length 10, their passwords are not hashed for testing purpose

2. Anti-sql injection

Since sql queries are passed to the server in the form of string, if there's no control for what the user inputs, there may be some security issues known as sql injection.

For example, in the login percedure, if we have a sql code like this:

```
SELECT * FROM Customer WHERE email='{email}' AND password='{password}'
```

In this case, if the user input is

```
email="' OR 1=1 --"
```

The final sql query will become:

```
SELECT * FROM Customer WHERE email='' OR 1=1 -- AND password='{password}'
```

The password check in the WHERE clause is not executed, which is why such sql query is dangerous.

To handle this problem, I implemented an anti-injected function in utils.py, which put a escape character `\` in front of every `"\"` or `"\\"`, and delete any `;`. Though this is not handling every case of sql injection, but

this avoids a lot of common potential payloads.

3. Access security

Although in the GUI, we only provides link to the pages that certain user has access to, this does not stop users from directly accessing the page via url (if they know them in advance).

For example, suppose we have a user who hasn't logged in yet, who should only be accessing search flights page (and he is only provided with this link). If he know there is `\manage_staff` page on the server, he could access it directly using `127.0.0.1:5000\manage_staff`, so that we need to prevent this in the backend.

To handle this problem, in the front of each page, we have:

```
if session['role'] != 'Staff':
        return render_template('home.html', msg=[True, "You are not logged
in as a staff"])
```

Which identify the current user's role and block their access if they are not supposed to be there.

4. Login & Log out security

We properly handle log-in and log-out status using session, and the session information is properly released upon log out.

For example, if we didn't execute

```
session.pop('admin', None)
```

The next time a customer logged in, he would have access to the admin privileges even if he shouldn't.

Triggers

In real life, every airline has its own pool of flight number, two seperate airlines may have the same flight number. For example, `AA001` may co-exists with `MU001`.

To implement this, `AUTO INCREMENT` is not enough for the attritube `flight_num`, we need a trigger like this:

```
CREATE TRIGGER auto_flight_num BEFORE INSERT ON flight
FOR EACH ROW
BEGIN
    DECLARE auto_flight_num INT;
    SET auto_flight_num = (SELECT MAX(flight_num) FROM flight WHERE
IATA_code = NEW.IATA_code);
    IF auto_flight_num IS NULL THEN
        SET auto_flight_num = 0;
```

```
    END IF;
    SET NEW.flight_num = auto_flight_num + 1;
END;//
```

Where we select the greatest flight number of the airline being inserted, and set the new flight number to be 1 larger.

Test datas

Designing test datas for such a project may be troublesome, espically there is so many data to generate.

In my project, I utilized `Faker`, to generate a relatively big scale of data (more than 10000 insertions). And the test data is working fine.

PS: Suggested test dummy account:

Suggest customer account: Email: valdezamy@example.com password: !m0oLN8f*E Ticket_count: 34

Suggest agent account: Email: adam98@example.net password: _6RYgQ+s26 Ticket_count: 75

Proper documentation

Like what is presented above

p.s.

This is really a tiring work