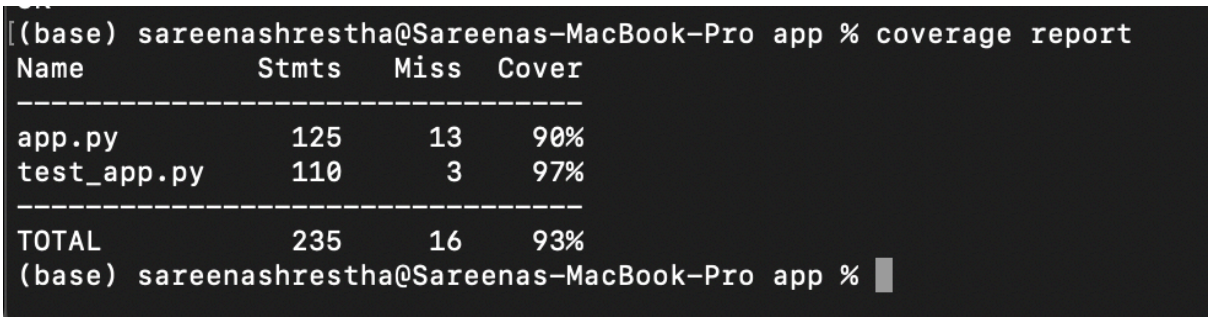
1. 

2. Explanations for Missing Statements in Code Coverage Testing:
    a. app.py
        i. check_email function:
            1. The entire check_email function is being skipped because we don't have any test cases that directly target this route. This function is meant to handle POST requests to the /check_email endpoint, where it checks if a given email exists in our database. Although we do thoroughly test the /register, /login, and /cancelBooking routes, we do not have any tests for the /check_email route. Without any tests that make a POST request to this endpoint, the coverage tool marks the entire function as missed.
        ii. Registration function, DOB validation:
            1. The first errors['dob_error'] line is being skipped because the current test cases only check for a valid date format (YYYY-MM-DD) when the user is underage. This means the format validation (if not re.match(dob_regex,dob)) is bypassed, and the test moves directly to the age validation in the else statement. Since there is no test case that provides an invalid date format, the format validation line is never executed, resulting in it being marked as missed in the coverage report.
        iii. main function call
            1. The main function call is not tested because it is assumed that this function will only run when it is called as a script within the CLI and is not called by any other file. It is not a part of any logic/functionality in the application that requires testing.
        iv. Index route
            1. The line user_input = request.form['user_input'] and return render_template('index.html', user_input=user_input) is not covered because none of the tests make a post request to the / route to provide data through user_input. The tests in test_app.py mainly focuses on login, registration, and booking routes and leaves the index route untested.
        v. Register route
            1. The except sqlite3.IntegrityError block in the register function is not being tested because the test only checks for an error

message “Email already registered”, rather than checking that the sqlite3.IntegrityError is explicitly raised and handled by the except block. It could not be triggered because there is already a check before the try block:

```
if existing_user:
        errors['email_error'] = "Email already registered."  # check for duplicate email before inserting the user into the db
```

to ensure no duplicate emails reach the database, hence not triggering the except block.

b. test_app.
    i. try-except in TestLogin
        1. The except condition is being skipped in our tests as the function is a part of the testing environment setup and the tests created were intended to cover application functionality. The assumption is made that the initial database addition of test data will always work due to the previous queries to clear the database, making the try-except case a formality rather than an intentional functionality.
    ii. main function call
        1. The main function call is not tested because it is assumed that this function will only run when it is called as a script within the CLI and is not called by any other file. It is not a part of any logic/functionality in the application that requires testing.