CREATE TABLE Passengers (

passenger\_id INT AUTO\_INCREMENT PRIMARY KEY,

record\_locator VARCHAR(10),

name VARCHAR(255)

);

CREATE TABLE Flights (

flight\_id INT AUTO\_INCREMENT PRIMARY KEY,

flight\_number VARCHAR(10),

date\_of\_flight DATE,

departure\_time TIME,

arrival\_time TIME,

class\_of\_service VARCHAR(50),

origin\_city VARCHAR(255),

destination\_city VARCHAR(255)

);

CREATE TABLE Reservations (

reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,

passenger\_id INT,

flight\_id INT,

ticket\_status ENUM('booked', 'reissued', 'canceled'),

FOREIGN KEY (passenger\_id) REFERENCES Passengers(passenger\_id),

FOREIGN KEY (flight\_id) REFERENCES Flights(flight\_id)

);

The error 1050 Table 'reservations' already exists means that the reservations table has already been created in your MySQL database. You don't need to create it again. Instead, you can verify the table's structure or modify it if necessary.

**Steps to Resolve the Issue**

**1. Verify the Existing Table**

Check if the reservations table already exists and view its structure:

DESCRIBE reservations;

This will show you the columns and their data types in the reservations table. Verify if the structure matches what you need.

**2. Modify the Existing Table (If Needed)**

If the reservations table exists but is missing some columns or has incorrect data types, you can use the ALTER TABLE statement to modify it.

For example:

* **Add a New Column**:

**Add a New Column**:

ALTER TABLE reservations ADD COLUMN arrival\_time TIME NOT NULL;

**Modify an Existing Column**:

ALTER TABLE reservations MODIFY COLUMN ticket\_status VARCHAR(30) DEFAULT 'booked';

**Rename a Column**:

ALTER TABLE reservations CHANGE COLUMN origin origin\_city VARCHAR(50);

**3. Drop and Recreate the Table (If Necessary)**

If the existing reservations table is incorrect and you want to start fresh, you can drop the table and recreate it.

**Warning**: Dropping a table will delete all data in it. Only do this if you're sure you don't need the existing data.

DROP TABLE reservations;

DROP TABLE reservations;

Then, recreate the table with the correct structure:

CREATE TABLE reservations (

reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,

passenger\_id INT NOT NULL,

flight\_id VARCHAR(20) NOT NULL,

date\_of\_flight DATE NOT NULL,

departure\_time TIME NOT NULL,

arrival\_time TIME NOT NULL,

origin\_city VARCHAR(50) NOT NULL,

destination\_city VARCHAR(50) NOT NULL,

ticket\_status VARCHAR(20) DEFAULT 'booked',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**4. Check for Duplicate Tables**

If you're still encountering issues, ensure there are no duplicate tables with similar names (e.g., Reservations, RESERVATIONS, etc.). MySQL table names are case-sensitive on some systems.

Run the following command to list all tables in your database:

SHOW TABLES;

**Next Steps**

1. If the reservations table already exists and is correct, proceed with inserting data or updating your Flask application to interact with it.
2. If you need to modify the table, use ALTER TABLE as shown above.
3. If you want to start fresh, drop and recreate the table.

Let me know if you need further assistance!

ALTER TABLE reservations

ADD COLUMN flight\_id VARCHAR(20) NOT NULL,

ADD COLUMN date\_of\_flight DATE NOT NULL,

ADD COLUMN departure\_time TIME NOT NULL,

ADD COLUMN arrival\_time TIME NOT NULL,

ADD COLUMN origin\_city VARCHAR(50) NOT NULL,

ADD COLUMN destination\_city VARCHAR(50) NOT NULL,

ADD COLUMN ticket\_status VARCHAR(20) DEFAULT 'booked';

**TO CREATE A NEW TABLE ALL TOGETHER:**

CREATE TABLE reservations (

reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,

passenger\_id INT NOT NULL,

flight\_id VARCHAR(20) NOT NULL,

date\_of\_flight DATE NOT NULL,

departure\_time TIME NOT NULL,

arrival\_time TIME NOT NULL,

origin\_city VARCHAR(50) NOT NULL,

destination\_city VARCHAR(50) NOT NULL,

ticket\_status VARCHAR(20) DEFAULT 'booked',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Best Practice: Normalize Your Database**

If your database is growing in complexity, consider normalizing it by creating separate tables for related data. For example:

1. **Passengers Table**:
   * Store passenger details (e.g., passenger\_id, name, email).
2. **Flights Table**:
   * Store flight details (e.g., flight\_id, flight\_number, origin\_city, destination\_city, departure\_time, arrival\_time).
3. **Reservations Table**:
   * Link passengers and flights (e.g., reservation\_id, passenger\_id, flight\_id, ticket\_status).

This approach avoids redundancy and makes your database easier to manage.

**Steps to Decide**

1. **Review the Purpose of the Existing Table**:
   * If the table is already tracking reservations, alter it.
   * If the table is for something else (e.g., ticket reissues), create a new table.
2. **Check the Existing Data**:
   * If the existing table has data that might conflict with the new fields, creating a new table is safer.
3. **Plan for Future Expansion**:
   * If you anticipate adding more fields or relationships, consider creating a new table to keep things organized.

**Recommendation**

* If the existing table is already for reservations, **alter the table** to add the new fields.
* If the existing table is for something else (e.g., ticket reissues), **create a new table** for reservations.

Let me know if you need help with the SQL commands or updating your Flask application to work with the new table structure!

**Use unittest (Built-in Python Testing Framework)**

The unittest module is included in Python's standard library and is a great starting point for testing your application.

I will be using "unittest" which is built-in Python Testing Framework.

I found the "unittest" module was included in Python's standard library and I decided to use it for testing my application.