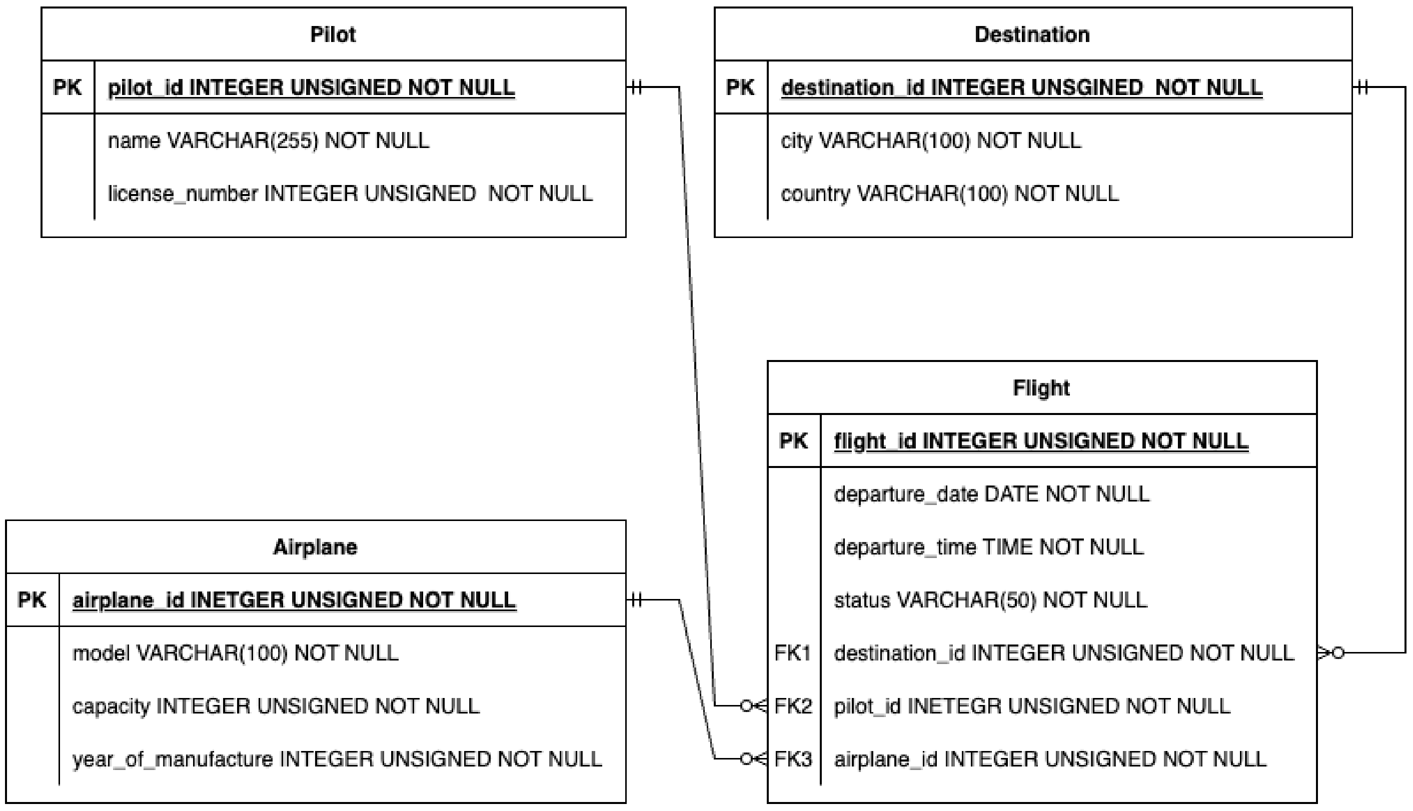
Flight Management System Coursework

ER Diagram:


Pilot
PK
pilot_id INTEGER UNSIGNED NOT NULL
name VARCHAR(255) NOT NULL
license_number INTEGER UNSIGNED NOT NULL
Destination
PK
destination_id INTEGER UNSGINED NOT NULL
city VARCHAR(100) NOT NULL
country VARCHAR(100) NOT NULL
Flight
PK
flight_id INTEGER UNSIGNED NOT NULL
departure_date DATE NOT NULL
departure_time TIME NOT NULL
status VARCHAR(50) NOT NULL
FK1
destination_id INTEGER UNSIGNED NOT NULL
FK2
pilot_id INETEGR UNSIGNED NOT NULL
FK3
airplane_id INTEGER UNSIGNED NOT NULL
Airplane
PK
airplane_id INETGER UNSIGNED NOT NULL
model VARCHAR(100) NOT NULL
capacity INTEGER UNSIGNED NOT NULL
year_of_manufacture INTEGER UNSIGNED NOT NULL

Relational Schema:

Pilot table:

```
CREATE TABLE Pilot (
   pilot_id INTEGER UNSIGNED PRIMARY KEY NOT NULL,
   name VARCHAR(255) NOT NULL,
   license_number INTEGER UNSIGNED NOT NULL);
```

The Pilot table stores information about the pilots, each pilot has a unique identifier (their pilot id) along with other specific attributes like their name and license number. The Primary Key is pilot_id as it ensures that each record in the pilot table is unique and identifiable, it must be provided (cannot be null) and its data type is an integer that cannot be negative. The name of the pilot must be provided (cannot be null) and the data type is a variable length string (max 255 characters). The pilot's licence number must be provided (cannot be null) and the data type is an integer that cannot be negative.

Destination table:

```
CREATE TABLE Destination (
   destination_id INTEGER UNSIGNED PRIMARY KEY NOT NULL,
   city VARCHAR(100) NOT NULL,
   country VARCHAR(100) NOT NULL);
```

The Destination table contains information about the destinations available in the flight management system. Each record represents one destination and includes the city and country details. The Primary Key is destination_id as it uniquely identifies each destination, it must be provided (cannot be null) and the data type is an integer that cannot be negative. The next attribute is city, it must be provided (cannot be null) and the data type for this is a variable length string (max 100 characters). The next attribute is country, it must be provided (cannot be null) and the data type is a variable length string (max 100 characters).

Airplane table:

```
CREATE TABLE Airplane (
   airplane_id INTEGER UNSIGNED PRIMARY KEY NOT NULL,
   model VARCHAR(100) NOT NULL,
   capacity INTEGER UNSIGNED NOT NULL,
   year_of_manufacture INTEGER UNSIGNED NOT NULL);
```

The Airplane table stores information about airplanes and each record represents an airplane, including its unique identifier (airplane id), model, capacity, and year of manufacture. The Primary Key is airplane_id as it uniquely identifies each airplane in the table, it must be provided (cannot be null) and its data type is an integer that cannot be negative. The next attribute is model, it must be provided (cannot be null) and its data type is a variable length string (max 100 characters). The next attribute is capacity meaning the maximum number of passengers the plane can hold, it must be provided (cannot be null) and the data type is an integer that cannot be negative. The next attribute is year_of_manufacture, it must be provided (cannot be null) and the data type is an integer that cannot be negative.

Flight table:

```
CREATE TABLE Flight (
   flight_id INTEGER UNSIGNED PRIMARY KEY NOT NULL,
   departure_date DATE NOT NULL,
   departure_time TIME NOT NULL,
   status VARCHAR(50) NOT NULL,
   destination_id INTEGER UNSIGNED NOT NULL,
   pilot_id INTEGER UNSIGNED NOT NULL,
   airplane_id INTEGER UNSIGNED NOT NULL,
   FOREIGN KEY (destination_id) REFERENCES Destination(destination_id),
   FOREIGN KEY (pilot_id) REFERENCES Pilot(pilot_id),
   FOREIGN KEY (airplane_id) REFERENCES Airplane(airplane_id));
```

The Flight table stores information about individual flights. The Primary Key is flight_id, as this uniquely identifies each flight. The table also contains a flight number, departure date, departure time, status, and references to other related entities. The destination_id, pilot_id, and airplane_id columns are foreign keys that reference the Destination, Pilot, and Airplane tables, respectively. These foreign keys ensure that every flight is associated with an existing destination, pilot, and airplane.

- Primary Key: flight_id – uniquely identifies each flight.
- Foreign Key: destination_id – references the destination_id in the Destination table, linking each flight to a specific destination.
- Foreign Key: pilot_id – references the pilot_id in the Pilot table, linking each flight to an assigned pilot.
- Foreign Key: airplane_id – references the airplane_id in the Airplane table, linking each flight to a specific airplane.

Database structure and purpose of each table:

The flight management database is comprised of four main tables; Pilot table, Destination table; Airplane table and the Flight table. Each table serves a unique role and provides specific information and together the database stores information that can be used to manage flights pilots and resources within an airline.

1. Pilot table

The Pilot table stores information about the pilots who are part of the airline. It maintains the records of each pilot, including their personal information and credentials. The pilot_id is used as a reference in the Flight table, therefore each flight is assigned to a specific pilot whose credentials are stored in the Pilot table. This also allows the user to manage pilot schedules and ensure that only licensed pilots are assigned to flights.

2. Destination table

The Destination table keeps records of various destinations served by the airline. It includes information about each city and country that is part of the airline's routes. The destination_id is used as a reference in the Flight table to assign each flight a specific destination from the set destinations that are saved in the system. This table helps in managing and organizing flight routes and determining which cities and countries the airline services. It allows the customer to modify destination information if the airlines flight routes change and they don't go to a particular destination anymore, or they go to a new destination.

3. Airplane table

The Airplane table stores information about the airplanes available for flights, including details like the model of the airplane, its passenger capacity, and the year of manufacture. The airplane_id (primary key of the Airplane table) is used as a reference in the Flight table to assign a specific airplane to a particular flight. This allows the user to track which airplanes are being used for which flights, this could be important when considering which airplanes are assigned to which flight based on their capacity.

4. Flight table

The Flight table is the core of the flight management system as it stores information about each flight, including the flight schedule (departure time and date), status, and the relationships with the other tables. The Flight table integrates information from the Pilot, Destination, and Airplane tables through foreign keys. The destination_id, pilot_id, and airplane_id fields establish relationships between flights and their associated destinations, pilots, and airplanes. This table is essential for managing flight schedules, ensuring that each flight is assigned to a destination, an airplane, and a certified pilot, from the airline system.

Relationships between tables:

The pilot table is connected to the flight table through the pilot_id, ensuring each flight has a specific assigned pilot. A pilot can be assigned to multiple flights. The destination table is connected to the flight table through the destination _id, each flight has a specific destination and many flights can fly to the same destination. The Airplane table is also connected to the flights table through the airplane_id, each flight has one airplane assigned to it and a single airplane can be assigned to multiple flights. The user would have to ensure that the schedule permits for the airplane to be used or pilot to be assigned to a flight.

SQL Queries:

1. **Insert a New Flight**

```
SELECT MAX(flight_id) FROM Flight
```

This query is used to retrieve the highest value of the flight_id column from the Flight table. The MAX() function returns the maximum value in the flight_id column. This query ensures that we can assign the next flight ID as last_flight_id + 1.

```
INSERT INTO Flight (departure_date, departure_time, status, destination_id, pilot_id,
airplane_id)
VALUES (?, ?, ?, ?, ?, ?, ?);
```

This query inserts a new flight into the Flight table. The ? placeholders are used to insert values provided by the user. The user provides the departure date and time, status, and associated IDs for the destination, pilot, and airplane. The placeholders are replaced with these values when the query is executed. The purpose of this query is to add a new flight to the database with the specified details, ensuring that the flight_id is unique and incremented properly, this is needed for scheduling new flights.

2. **Retrieve Flights by Criteria**

```
SELECT * FROM Flight WHERE 1=1
AND (destination_id = ? OR ? IS NULL)
AND (status = ? OR ? IS NULL)
AND (departure_date = ? OR ? IS NULL);
```

This query retrieves the columns from the Flight table based on multiple user-provided criteria, such as destination, status, and departure date. The WHERE is used to specify the conditions for filtering the rows that are returned. The 1=1 makes the query flexible by allowing additional AND conditions to be appended. The user may provide one or more filters (e.g., destination ID, status, departure date). If a filter is left blank, the query ignores that condition. This provides flexibility in retrieving flights based on user-specified criteria, allowing the user to easily search for specific flights or apply filters.

3. Update Flight Information

```
UPDATE Flight
SET departure_date = ?, departure_time = ?, status = ?
WHERE flight_id = ?;
```

This query updates information for an existing flight, such as changing the departure date, time, or status. The user specifies the flight ID of the flight they want to update and provides new values for the departure date, departure time, or status. The query then updates the Flight table by setting the new values for the columns departure_date, departure_time, and status for a particular flight. This is needed to manage schedule changes and flight adjustments. Airlines often need to update flight details due to unexpected delays, cancellations, or rescheduling.

4. Assign a Pilot to a Flight

```
UPDATE Flight
SET pilot_id = ?
WHERE flight_id = ?;
```

This query assigns a new pilot to an existing flight by updating the pilot_id. The user provides the flight ID and the new pilot ID. The Flight table is updated to reflect the change. This is used to manage pilot assignments and ensure that every flight has a registered pilot assigned.

5. View Pilot Schedule

```
SELECT * FROM Flight
WHERE pilot_id = ?;
```

This query retrieves all flights that are assigned to a specific pilot.The user provides the pilot ID, and the query retrieves all flights where the pilot_id matches the provided value. It is used to view the schedule for a particular pilot. This is necessary for tracking pilot availability.

6. View Destination Information

```
SELECT * FROM Destination
WHERE destination_id = ?;
```

This query retrieves detailed information about a specific destination. The user provides the destination ID, and the query returns the corresponding city and country information from the Destination table. It’s useful for providing details about a destination, helping users understand where flights are going.

7. **Update Destination Information**

```
UPDATE Destination
SET city = ?, country = ?
WHERE destination_id = ?;
```

This query updates information for an existing destination. The user specifies the destination ID and provides new values for city and country. The Destination table is updated to reflect setting the city and country values to the new values inputted for the given destination_id. This is used to update details about destinations, such as renaming cities or amending flight route information.

8. **View All Flights**

```
SELECT * FROM Flight;
```

This query retrieves all records from the Flight table. The query selects all columns and rows, displaying detailed information about every flight in the system. This can be helpful for the user to view the current schedule.

9. **Number of Flights to Each Destination**

```
SELECT d.destination_id, d.city, d.country, COUNT(f.flight_id) AS number_of_flights
FROM Destination d
LEFT JOIN Flight f ON d.destination_id = f.destination_id
GROUP BY d.destination_id, d.city, d.country;
```

This query provides a count of flights to each destination, which helps understand how frequently each city or country is being serviced.

- SELECT d.destination_id, d.city, d.country, COUNT(f.flight_id) AS number_of_flights: Selects the destination_id, city, and country from the Destination table and counts the number of flights for each destination.
- LEFT JOIN Flight f ON d.destination_id = f.destination_id: Performs a LEFT JOIN to connect the Destination table with the Flight table, ensuring that all destinations are included even if there are no flights to that destination.
- GROUP BY d.destination_id, d.city, d.country: Groups the results by destination to calculate the count of flights for each.

This query is helpful in allowing the user to understand which destinations are most popular which could influence flight routes.

10. **Number of Flights Assigned to Each Pilot**

```
SELECT p.pilot_id, p.name, COUNT(f.flight_id) AS number_of_flights
FROM Pilot p
LEFT JOIN Flight f ON p.pilot_id = f.pilot_id
GROUP BY p.pilot_id, p.name;
```

This query provides a count of flights assigned to each pilot, which is useful for balancing workloads and ensuring pilots are not overworked.

- SELECT p.pilot_id, p.name, COUNT(f.flight_id) AS number_of_flights: Selects the pilot_id and name from the Pilot table and counts the number of flights assigned to each pilot.
- LEFT JOIN Flight f ON p.pilot_id = f.pilot_id: Performs a LEFT JOIN between the Pilot and Flight tables, ensuring all pilots are included even if they are not assigned to any flights.
- GROUP BY p.pilot_id, p.name: Groups the results by pilot to calculate the count of flights for each pilot.

This is useful for the user to manage pilot schedules effectively, ensuring that the workload is distributed evenly among all pilots.

11. Number of Flights for Each Airplane

```
SELECT a.airplane_id, a.model, COUNT(f.flight_id) AS number_of_flights
FROM Airplane a
LEFT JOIN Flight f ON a.airplane_id = f.airplane_id
GROUP BY a.airplane_id, a.model;
```

This query shows how often each airplane is used, helping in the management of airplane maintenance and scheduling.

- SELECT a.airplane_id, a.model, COUNT(f.flight_id) AS number_of_flights: Selects the airplane_id and model from the Airplane table and counts the number of flights assigned to each airplane.
- LEFT JOIN Flight f ON a.airplane_id = f.airplane_id: Connects the Airplane table with the Flight table to count how often each airplane is used.
- GROUP BY a.airplane_id, a.model: Groups the results by airplane to calculate the count of flights for each.

Tracking airplane usage is essential, ensuring proper scheduling and maintenance management. To extend this database further one could include further attribute regarding airplane maintenance information.

12. Delete flight

```
DELETE FROM Flight WHERE flight_id = ?;
```

This query deletes a flight entry from the Flight table based on the specified flight ID, this can be used to keep the database up to date and remove any flights that are no longer scheduled.

13. View complete flight details

```
SELECT
    f.flight_id,
    f.departure_date,
    f.departure_time,
    f.status,
    d.city AS destination_city,
    d.country AS destination_country,
    p.name AS pilot_name,
    a.model AS airplane_model,
    a.capacity AS airplane_capacity
FROM
    Flight f, Destination d, Pilot p, Airplane a
WHERE
    f.destination_id = d.destination_id
    AND f.pilot_id = p.pilot_id
    AND f.airplane_id = a.airplane_id
```

This query involves all four tables using an EQUI JOIN to combine the information from all four. The WHERE clause specifies conditions to match columns across different tables. The conditions are:

- f.destination_id = d.destination_id: Matches each flight to its destination using the destination_id.
- f.pilot_id = p.pilot_id: Matches each flight to the pilot using the pilot_id.
- f.airplane_id = a.airplane_id: Matches each flight to its airplane using the airplane_id.

As a result the EQUI JOIN includes only the rows that meet the specified WHERE conditions, providing an overview of the flight, including its departure information, destination details, pilot name, and airplane model and capacity.

Reflection:

I began this project by researching key information that is important for managing flights to inform me of what should be included in a flight management database. This helped me understand the different attributes and entities needed, such as flight schedules, pilot details, destinations and airplane information. Then, I started by creating my ER diagram using draw.io, and after becoming familiar with the software, I found it relatively simple to create the diagrams.

Next, I set up my database in SQL using the necessary commands. To do this, I reviewed lab material that provided a solid foundation for me to construct my SQL tables. I then used AI to generate sample data for all four of the tables and inputted this data into my SQL database.

After setting up the database, I moved on to creating a Python file for the command-line interface (CLI). This required me to revisit some Python concepts and understand the commands and functions that allow the Python file to connect and interact with the SQL database. It was a valuable refresher, and I appreciated the challenge of integrating Python with SQL for a more interactive experience.

One of the significant challenges I faced was with the view_flights_by_criteria function. Initially, I wasn't sure how to program the software to retrieve flights based on multiple criteria simultaneously. At first, I coded the software to let the user filter flights by selecting one criterion at a time, which was limiting. To solve this, I used the append function to allow the user to give inputs for multiple criteria, enabling them to retrieve flights based on several filters at once. To achieve this, I created a base query that starts with a simple WHERE condition, and as the user provided additional inputs, I dynamically added AND conditions and stored each input in a list. This list was then used with cursor.execute(query, params) to execute the query with multiple parameters.

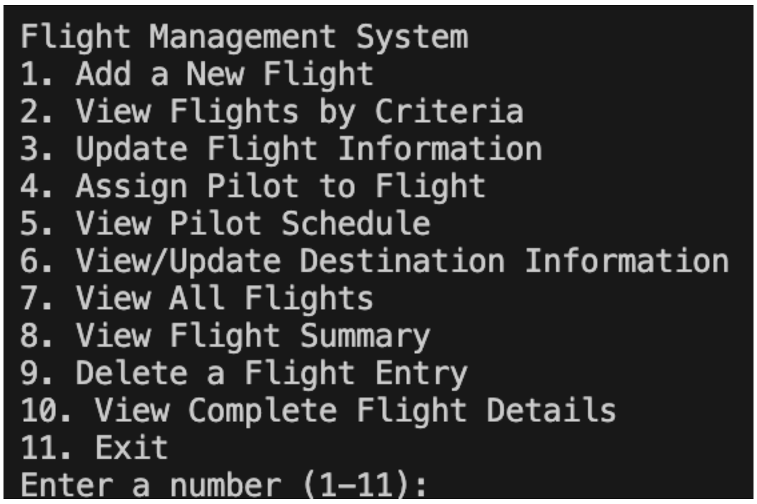
I encountered a similar challenge with the update_flight_info function, as I wasn't initially sure how to allow the user to make multiple updates simultaneously. Eventually, I applied the same strategy of building a flexible query that adjusts based on the user's inputs, making the function more versatile.

After completing the development, I thoroughly tested the application by trying all of the functions and deliberately inputting various values to try and break the application. During testing, I noticed that entering invalid inputs caused error messages, and the application would often close abruptly. To address this, I worked on making the code more robust by ensuring that invalid responses were adequately handled. I decided to implement try and except blocks to ensure that if an invalid input was provided, the application wouldn't crash but would simply print "Invalid input." This significantly improved the user experience and reliability of the system.

Overall, this project was a helpful learning experience that helped me understand how to design a database, create a user-friendly interface, and handle challenges in making the software both functional and robust. The ability to solve problems, such as allowing multiple criteria for viewing flights and ensuring robust error handling, has greatly enhanced my confidence in developing integrated systems.

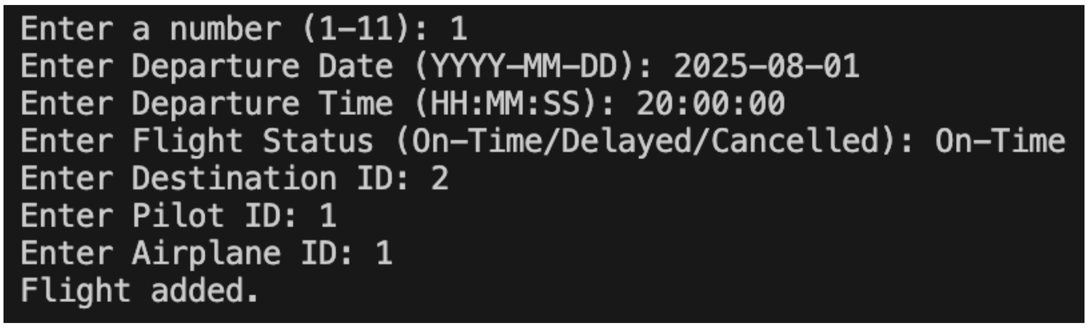
Screenshots:

Initial Menu:



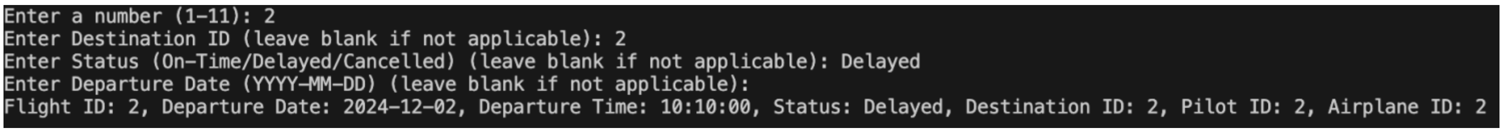
```
Flight Management System
1. Add a New Flight
2. View Flights by Criteria
3. Update Flight Information
4. Assign Pilot to Flight
5. View Pilot Schedule
6. View/Update Destination Information
7. View All Flights
8. View Flight Summary
9. Delete a Flight Entry
10. View Complete Flight Details
11. Exit
Enter a number (1-11):
```

1. Add a New Flight



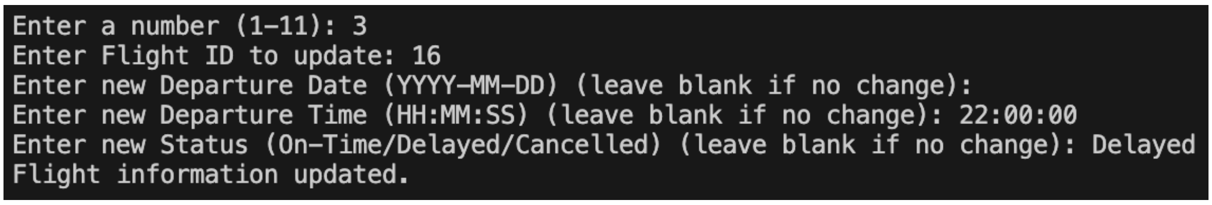
```
Enter a number (1-11): 1
Enter Departure Date (YYYY-MM-DD): 2025-08-01
Enter Departure Time (HH:MM:SS): 20:00:00
Enter Flight Status (On-Time/Delayed/Cancelled): On-Time
Enter Destination ID: 2
Enter Pilot ID: 1
Enter Airplane ID: 1
Flight added.
```

2. View Flights by Criteria



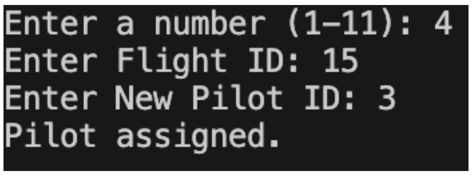
```
Enter a number (1-11): 2
Enter Destination ID (leave blank if not applicable): 2
Enter Status (On-Time/Delayed/Cancelled) (leave blank if not applicable): Delayed
Enter Departure Date (YYYY-MM-DD) (leave blank if not applicable):
Flight ID: 2, Departure Date: 2024-12-02, Departure Time: 10:10:00, Status: Delayed, Destination ID: 2, Pilot ID: 2, Airplane ID: 2
```

3. Update Flight Information



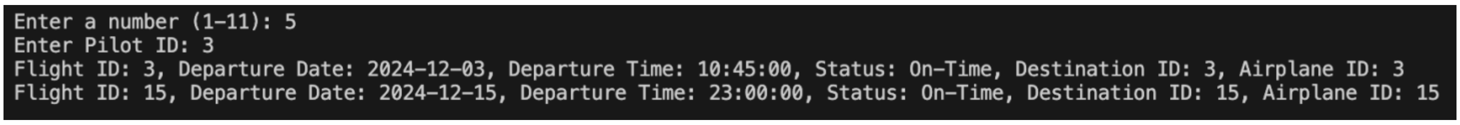
```
Enter a number (1-11): 3
Enter Flight ID to update: 16
Enter new Departure Date (YYYY-MM-DD) (leave blank if no change):
Enter new Departure Time (HH:MM:SS) (leave blank if no change): 22:00:00
Enter new Status (On-Time/Delayed/Cancelled) (leave blank if no change): Delayed
Flight information updated.
```

4. Assign Pilot to Flight



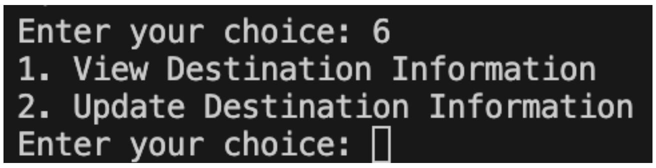
```
Enter a number (1-11): 4
Enter Flight ID: 15
Enter New Pilot ID: 3
Pilot assigned.
```

5. View Pilot Schedule



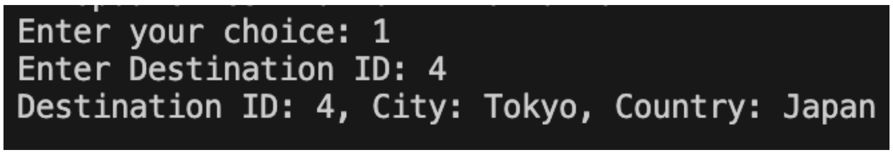
```
Enter a number (1-11): 5
Enter Pilot ID: 3
Flight ID: 3, Departure Date: 2024-12-03, Departure Time: 10:45:00, Status: On-Time, Destination ID: 3, Airplane ID: 3
Flight ID: 15, Departure Date: 2024-12-15, Departure Time: 23:00:00, Status: On-Time, Destination ID: 15, Airplane ID: 15
```

6. View/Update Destination Information



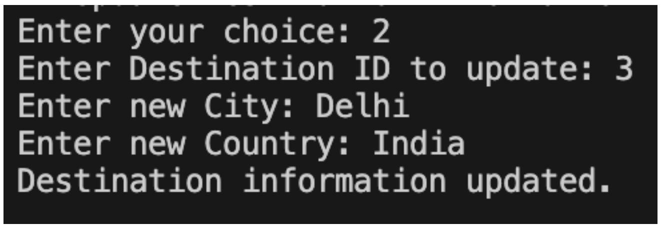
```
Enter your choice: 6
1. View Destination Information
2. Update Destination Information
Enter your choice: 
```

   a. View



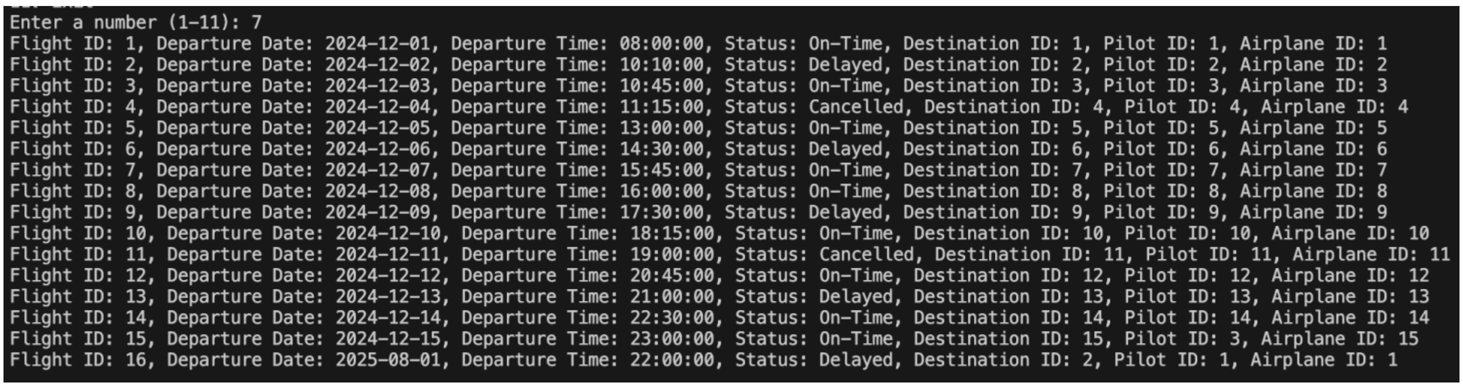
```
Enter your choice: 1
Enter Destination ID: 4
Destination ID: 4, City: Tokyo, Country: Japan
```

   b. Update



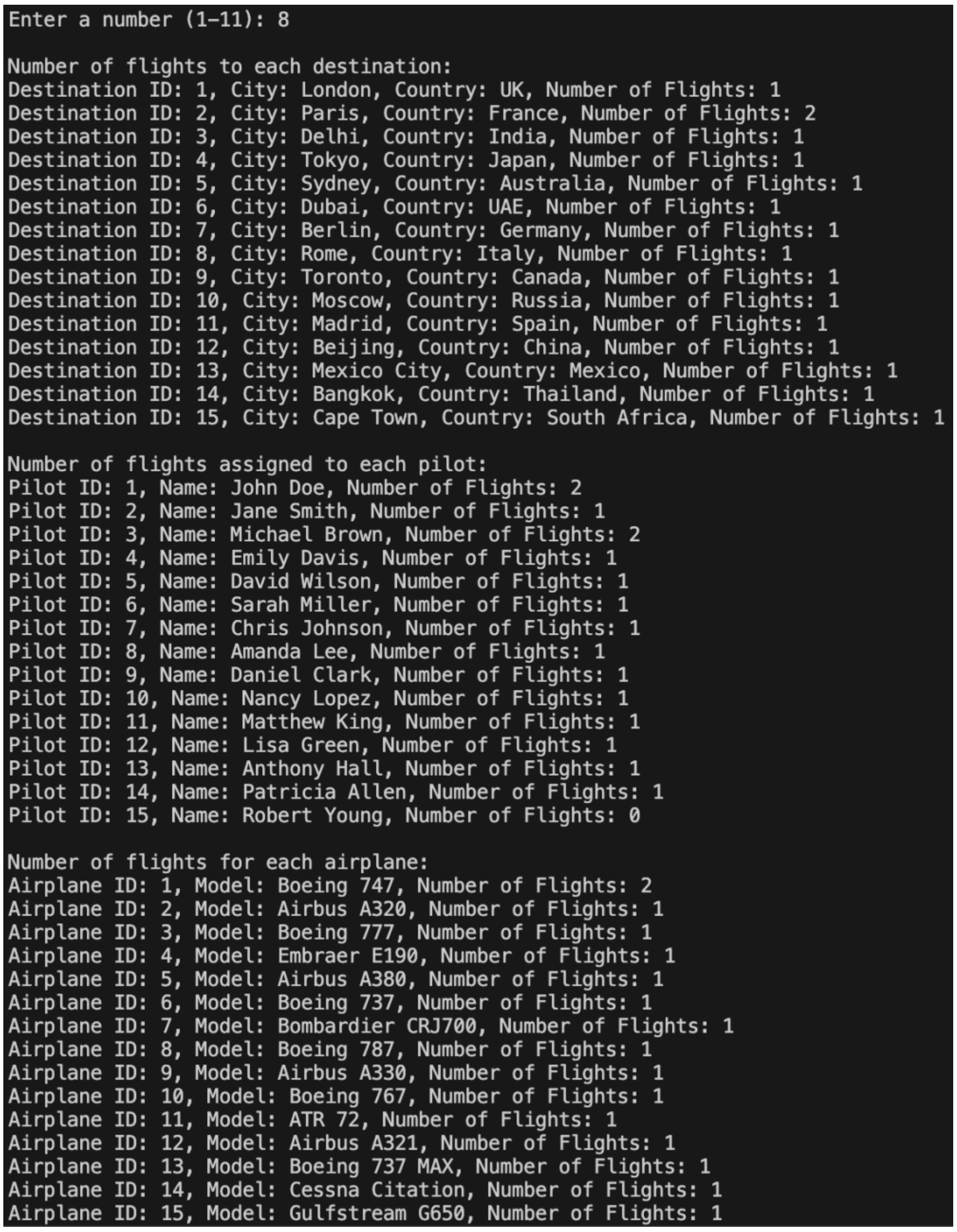
```
Enter your choice: 2
Enter Destination ID to update: 3
Enter new City: Delhi
Enter new Country: India
Destination information updated.
```

7. View All Flights



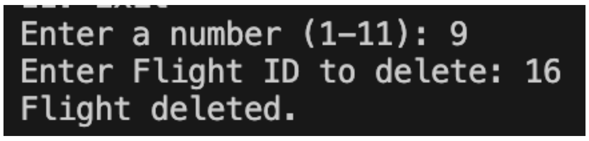
```
Enter a number (1–11): 7
Flight ID: 1, Departure Date: 2024–12–01, Departure Time: 08:00:00, Status: On–Time, Destination ID: 1, Pilot ID: 1, Airplane ID: 1
Flight ID: 2, Departure Date: 2024–12–02, Departure Time: 10:10:00, Status: Delayed, Destination ID: 2, Pilot ID: 2, Airplane ID: 2
Flight ID: 3, Departure Date: 2024–12–03, Departure Time: 10:45:00, Status: On–Time, Destination ID: 3, Pilot ID: 3, Airplane ID: 3
Flight ID: 4, Departure Date: 2024–12–04, Departure Time: 11:15:00, Status: Cancelled, Destination ID: 4, Pilot ID: 4, Airplane ID: 4
Flight ID: 5, Departure Date: 2024–12–05, Departure Time: 13:00:00, Status: On–Time, Destination ID: 5, Pilot ID: 5, Airplane ID: 5
Flight ID: 6, Departure Date: 2024–12–06, Departure Time: 14:30:00, Status: Delayed, Destination ID: 6, Pilot ID: 6, Airplane ID: 6
Flight ID: 7, Departure Date: 2024–12–07, Departure Time: 15:45:00, Status: On–Time, Destination ID: 7, Pilot ID: 7, Airplane ID: 7
Flight ID: 8, Departure Date: 2024–12–08, Departure Time: 16:00:00, Status: On–Time, Destination ID: 8, Pilot ID: 8, Airplane ID: 8
Flight ID: 9, Departure Date: 2024–12–09, Departure Time: 17:30:00, Status: Delayed, Destination ID: 9, Pilot ID: 9, Airplane ID: 9
Flight ID: 10, Departure Date: 2024–12–10, Departure Time: 18:15:00, Status: On–Time, Destination ID: 10, Pilot ID: 10, Airplane ID: 10
Flight ID: 11, Departure Date: 2024–12–11, Departure Time: 19:00:00, Status: Cancelled, Destination ID: 11, Pilot ID: 11, Airplane ID: 11
Flight ID: 12, Departure Date: 2024–12–12, Departure Time: 20:45:00, Status: On–Time, Destination ID: 12, Pilot ID: 12, Airplane ID: 12
Flight ID: 13, Departure Date: 2024–12–13, Departure Time: 21:00:00, Status: Delayed, Destination ID: 13, Pilot ID: 13, Airplane ID: 13
Flight ID: 14, Departure Date: 2024–12–14, Departure Time: 22:30:00, Status: On–Time, Destination ID: 14, Pilot ID: 14, Airplane ID: 14
Flight ID: 15, Departure Date: 2024–12–15, Departure Time: 23:00:00, Status: On–Time, Destination ID: 15, Pilot ID: 3, Airplane ID: 15
Flight ID: 16, Departure Date: 2025–08–01, Departure Time: 22:00:00, Status: Delayed, Destination ID: 2, Pilot ID: 1, Airplane ID: 1
```

8. View Flight Summary



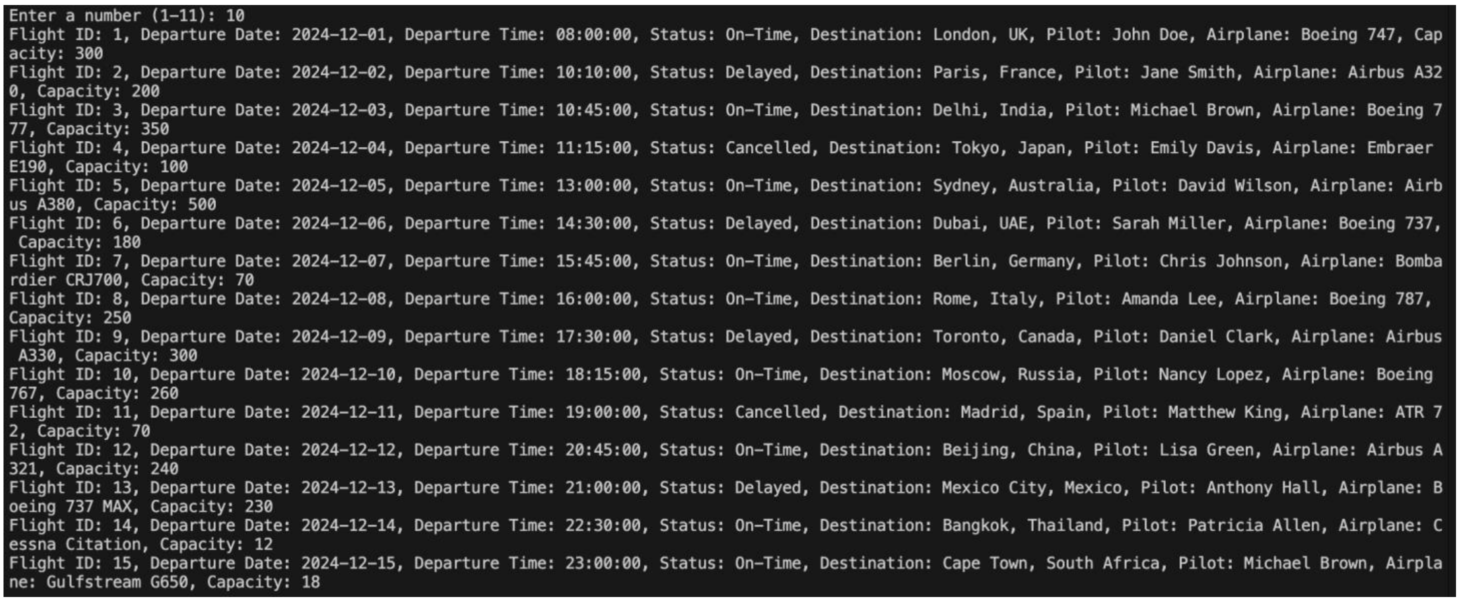
```
Enter a number (1–11): 8

Number of flights to each destination:
Destination ID: 1, City: London, Country: UK, Number of Flights: 1
Destination ID: 2, City: Paris, Country: France, Number of Flights: 2
Destination ID: 3, City: Delhi, Country: India, Number of Flights: 1
Destination ID: 4, City: Tokyo, Country: Japan, Number of Flights: 1
Destination ID: 5, City: Sydney, Country: Australia, Number of Flights: 1
Destination ID: 6, City: Dubai, Country: UAE, Number of Flights: 1
Destination ID: 7, City: Berlin, Country: Germany, Number of Flights: 1
Destination ID: 8, City: Rome, Country: Italy, Number of Flights: 1
Destination ID: 9, City: Toronto, Country: Canada, Number of Flights: 1
Destination ID: 10, City: Moscow, Country: Russia, Number of Flights: 1
Destination ID: 11, City: Madrid, Country: Spain, Number of Flights: 1
Destination ID: 12, City: Beijing, Country: China, Number of Flights: 1
Destination ID: 13, City: Mexico City, Country: Mexico, Number of Flights: 1
Destination ID: 14, City: Bangkok, Country: Thailand, Number of Flights: 1
Destination ID: 15, City: Cape Town, Country: South Africa, Number of Flights: 1

Number of flights assigned to each pilot:
Pilot ID: 1, Name: John Doe, Number of Flights: 2
Pilot ID: 2, Name: Jane Smith, Number of Flights: 1
Pilot ID: 3, Name: Michael Brown, Number of Flights: 2
Pilot ID: 4, Name: Emily Davis, Number of Flights: 1
Pilot ID: 5, Name: David Wilson, Number of Flights: 1
Pilot ID: 6, Name: Sarah Miller, Number of Flights: 1
Pilot ID: 7, Name: Chris Johnson, Number of Flights: 1
Pilot ID: 8, Name: Amanda Lee, Number of Flights: 1
Pilot ID: 9, Name: Daniel Clark, Number of Flights: 1
Pilot ID: 10, Name: Nancy Lopez, Number of Flights: 1
Pilot ID: 11, Name: Matthew King, Number of Flights: 1
Pilot ID: 12, Name: Lisa Green, Number of Flights: 1
Pilot ID: 13, Name: Anthony Hall, Number of Flights: 1
Pilot ID: 14, Name: Patricia Allen, Number of Flights: 1
Pilot ID: 15, Name: Robert Young, Number of Flights: 0

Number of flights for each airplane:
Airplane ID: 1, Model: Boeing 747, Number of Flights: 2
Airplane ID: 2, Model: Airbus A320, Number of Flights: 1
Airplane ID: 3, Model: Boeing 777, Number of Flights: 1
Airplane ID: 4, Model: Embraer E190, Number of Flights: 1
Airplane ID: 5, Model: Airbus A380, Number of Flights: 1
Airplane ID: 6, Model: Boeing 737, Number of Flights: 1
Airplane ID: 7, Model: Bombardier CRJ700, Number of Flights: 1
Airplane ID: 8, Model: Boeing 787, Number of Flights: 1
Airplane ID: 9, Model: Airbus A330, Number of Flights: 1
Airplane ID: 10, Model: Boeing 767, Number of Flights: 1
Airplane ID: 11, Model: ATR 72, Number of Flights: 1
Airplane ID: 12, Model: Airbus A321, Number of Flights: 1
Airplane ID: 13, Model: Boeing 737 MAX, Number of Flights: 1
Airplane ID: 14, Model: Cessna Citation, Number of Flights: 1
Airplane ID: 15, Model: Gulfstream G650, Number of Flights: 1
```

9. Delete a Flight Entry



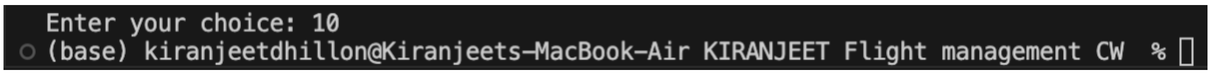
```
Enter a number (1-11): 9
Enter Flight ID to delete: 16
Flight deleted.
```

10. View complete flight details



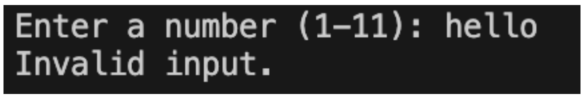
```
Enter a number (1-11): 10
Flight ID: 1, Departure Date: 2024-12-01, Departure Time: 08:00:00, Status: On-Time, Destination: London, UK, Pilot: John Doe, Airplane: Boeing 747, Cap
acity: 300
Flight ID: 2, Departure Date: 2024-12-02, Departure Time: 10:10:00, Status: Delayed, Destination: Paris, France, Pilot: Jane Smith, Airplane: Airbus A32
0, Capacity: 200
Flight ID: 3, Departure Date: 2024-12-03, Departure Time: 10:45:00, Status: On-Time, Destination: Delhi, India, Pilot: Michael Brown, Airplane: Boeing 7
77, Capacity: 350
Flight ID: 4, Departure Date: 2024-12-04, Departure Time: 11:15:00, Status: Cancelled, Destination: Tokyo, Japan, Pilot: Emily Davis, Airplane: Embraer
E190, Capacity: 100
Flight ID: 5, Departure Date: 2024-12-05, Departure Time: 13:00:00, Status: On-Time, Destination: Sydney, Australia, Pilot: David Wilson, Airplane: Airb
us A380, Capacity: 500
Flight ID: 6, Departure Date: 2024-12-06, Departure Time: 14:30:00, Status: Delayed, Destination: Dubai, UAE, Pilot: Sarah Miller, Airplane: Boeing 737,
 Capacity: 180
Flight ID: 7, Departure Date: 2024-12-07, Departure Time: 15:45:00, Status: On-Time, Destination: Berlin, Germany, Pilot: Chris Johnson, Airplane: Bomba
rdier CRJ700, Capacity: 70
Flight ID: 8, Departure Date: 2024-12-08, Departure Time: 16:00:00, Status: On-Time, Destination: Rome, Italy, Pilot: Amanda Lee, Airplane: Boeing 787,
Capacity: 250
Flight ID: 9, Departure Date: 2024-12-09, Departure Time: 17:30:00, Status: Delayed, Destination: Toronto, Canada, Pilot: Daniel Clark, Airplane: Airbus
 A330, Capacity: 300
Flight ID: 10, Departure Date: 2024-12-10, Departure Time: 18:15:00, Status: On-Time, Destination: Moscow, Russia, Pilot: Nancy Lopez, Airplane: Boeing
767, Capacity: 260
Flight ID: 11, Departure Date: 2024-12-11, Departure Time: 19:00:00, Status: Cancelled, Destination: Madrid, Spain, Pilot: Matthew King, Airplane: ATR 7
2, Capacity: 70
Flight ID: 12, Departure Date: 2024-12-12, Departure Time: 20:45:00, Status: On-Time, Destination: Beijing, China, Pilot: Lisa Green, Airplane: Airbus A
321, Capacity: 240
Flight ID: 13, Departure Date: 2024-12-13, Departure Time: 21:00:00, Status: Delayed, Destination: Mexico City, Mexico, Pilot: Anthony Hall, Airplane: B
oeing 737 MAX, Capacity: 230
Flight ID: 14, Departure Date: 2024-12-14, Departure Time: 22:30:00, Status: On-Time, Destination: Bangkok, Thailand, Pilot: Patricia Allen, Airplane: C
essna Citation, Capacity: 12
Flight ID: 15, Departure Date: 2024-12-15, Departure Time: 23:00:00, Status: On-Time, Destination: Cape Town, South Africa, Pilot: Michael Brown, Airpla
ne: Gulfstream G650, Capacity: 18
```

11. Exit



```
  Enter your choice: 10
○ (base) kiranjeetdhillon@Kiranjeets-MacBook-Air KIRANJEET Flight management CW  %
```

Invalid input response:



```
Enter a number (1-11): hello
Invalid input.
```