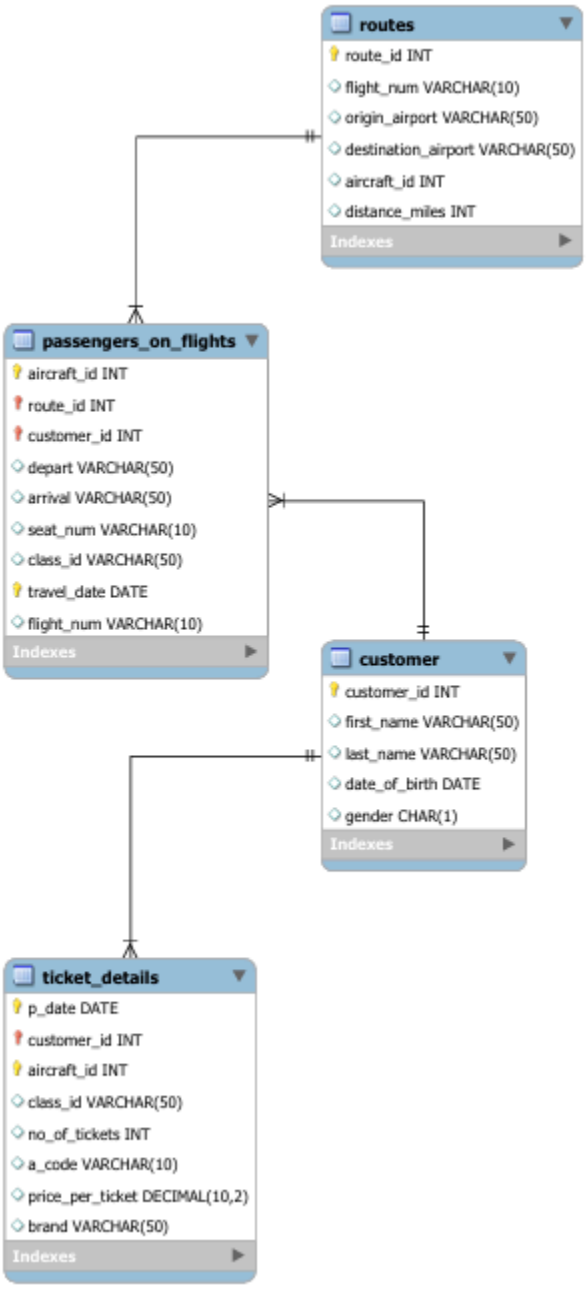
1. Create an ER diagram for the given airlines database.



2. Write a query to create a route_details table using suitable data types for the fields, such as route_id, flight_num, origin_airport, destination_airport, aircraft_id, and distance_miles. Implement the check constraint for the flight number and unique constraint for the route_id fields. Also, make sure that the distance miles field is greater than 0.

```
2  CREATE TABLE route_details (
3      route_id INT PRIMARY KEY UNIQUE,
4      flight_num VARCHAR(10) CHECK (flight_num LIKE '1%'),
5      origin_airport VARCHAR(50),
6      destination_airport VARCHAR(50),
7      aircraft_id INT,
8      distance_miles INT CHECK (distance_miles > 0)
9  );
```

3. Write a query to display all the passengers (customers) who have travelled in routes 01 to 25. Take data from the passengers_on_flights table.

```
12  select * from passengers_on_flights
13  where route_id between 1 and 25;
14
```

4. Write a query to identify the number of passengers and total revenue in business class from the ticket_details table.

```
SELECT                                                    -- ident
    COUNT(no_of_tickets) AS num_passengers,
    SUM(price_per_ticket * no_of_tickets) AS total_revenue
FROM
    ticket_details
WHERE
    class_id = 'Bussiness';
```

5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM customer;
```

6. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket_details tables.

```
SELECT c.customer_id, c.first_name, c.last_name
FROM customer c
JOIN ticket_details t ON c.customer_id = t.customer_id;
```

7. Write a query to identify the customer's first name and last name based on their customer ID and brand (Emirates) from the ticket_details table.

```
--
34 •    SELECT c.first_name, c.last_name
35      FROM customer c
36      JOIN ticket_details t ON c.customer_id = t.customer_id
37      WHERE t.brand = 'Emirates';
```

8. Write a query to identify the customers who have travelled by *Economy Plus* class using Group By and Having clause on the passengers_on_flights table.

```
40 •    SELECT class_id, COUNT(*) AS num_passengers
41      FROM passengers_on_flights
42      WHERE class_id = 'Economy Plus'
43      GROUP BY class_id
44      HAVING COUNT(*) > 0;
```

9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket_details table.

```
47 •    SELECT IF(SUM(price_per_ticket * no_of_tickets) > 10000, 'Revenue Crossed 10000', 'Revenue Below 10000')
48       AS revenue_status
49      FROM ticket_details;
50
```

10. Write a query to create and grant access to a new user to perform operations on a database.

```
52 •    CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password123';   -- create and gran
53 •    GRANT SELECT, INSERT, UPDATE ON air_cargo_analysis.* TO 'newuser'@'localhost';
54 •    FLUSH PRIVILEGES;
```

11. Write a query to find the maximum ticket price for each class using window functions on the ticket_details table.

```sql
SELECT             -- find the maximum ticket price for each class using window functions
    class_id,
    price_per_ticket,
    MAX(price_per_ticket) OVER (PARTITION BY class_id) AS max_ticket_price
FROM
    ticket_details;
```

12. Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers_on_flights table.

```sql
CREATE INDEX idx_route_id ON passengers_on_flights(route_id);
Select
route_id,depart,arrival,seat_num,class_id,travel_date,flight_num,aircraft_id
from passengers_on_flights
where route_id = 4;
```

13. For the route ID 4, write a query to view the execution plan of the passengers_on_flights table.

```sql
EXPLAIN                 -- extract the passengers whose route ID is 4 by im
SELECT
    aircraft_id,                -- Specify the columns you want to retrieve
    customer_id,
    depart,
    arrival,
    seat_num,
    class_id,
    travel_date,
    flight_num
FROM
    passengers_on_flights
WHERE
    route_id = 4;               -- Filter for route ID 4idx_route_id
```

14. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.

```sql
SELECT
    customer_id,                                    -- cal
    aircraft_id,
    SUM(price_per_ticket * no_of_tickets) AS total_price
FROM
    ticket_details
GROUP BY
    customer_id, aircraft_id WITH ROLLUP;
```

15. Write a query to create a view with only business class customers along with the brand of airlines.

```sql
create view business_class_brand As   -- cre
select
class_id,
brand
from
ticket_details
where
class_id = "bussiness";

select * from business_class_brand
```

16. Write a query to create a stored procedure to get the details of all passengers flying between a range of routes defined in run time. Also, return an error message if the table doesn't exist.

```
110     DELIMITER //                                        -- create a stored procedure
111
112     CREATE PROCEDURE get_passengers_by_route_range(
113         IN start_route INT,
114         IN end_route INT
115     )
116     BEGIN
117
118         IF (SELECT COUNT(*)
119             FROM information_schema.tables
120             WHERE table_name = 'passengers_on_flights') = 0 THEN
121             SIGNAL SQLSTATE '45000'
122             SET MESSAGE_TEXT = 'Error: Table passengers_on_flights does not exist.';
123         ELSE
124
125             SELECT *
126             FROM passengers_on_flights
127             WHERE route_id BETWEEN start_route AND end_route;
128         END IF;
129     END //
130
131     DELIMITER ;
132
133     call get_passengers_by_route_range(1,10);
```

17. Write a query to create a stored procedure that extracts all the details from the routes table where the travelled distance is more than 2000 miles.

```
DELIMITER //  -- create a stored procedure that extracts all the details from the

CREATE PROCEDURE get_long_distance_routes()
BEGIN
    -- Query to extract all routes where distance is more than 2000 miles
    SELECT *
    FROM routes
    WHERE distance_miles > 2000;  -- Filter for distances greater than 2000 miles
END //

DELIMITER ;
call get_long_distance_routes();
```

18. Write a query to create a stored procedure that groups the distance travelled by each flight into three categories. The categories are, short distance travel (SDT) for >=0 AND <= 2000 miles, intermediate distance travel (IDT) for >2000 AND <=6500, and long-distance travel (LDT) for >6500.

```
Delimiter //          -- create a stored procedure that groups the distance t

create procedure get_distance_category()
begin
select flight_num, distance_miles,
        case
            when distance_miles >=0 AND distance_miles <= 2000 then 'SDT'
            when distance_miles >=2000 AND distance_miles <= 6500 then 'IDT'
            when distance_miles >=6500 then 'LDT'
            else 'Unknown'
            END As travel_categories
from routes;

End //

delimiter ;
call get_distance_category()
```

19. Write a query to extract ticket purchase date, customer ID, class ID and specify if the complimentary services are provided for the specific class using a stored function in stored procedure on the ticket_details table.

Condition:

- If the class is *Business* and *Economy Plus,* then complimentary services are given as *Yes,* else it is *No*

```
DELIMITER //

CREATE PROCEDURE get_ticket_details_with_services()
BEGIN
    -- Query to extract ticket purchase date, customer ID, class ID and c
    SELECT
        p_date AS ticket_purchase_date,            -- Ticket purchase dat
        customer_id,                                -- Customer ID
        class_id,                                   -- Class ID
        get_complimentary_services(class_id) AS complimentary_services  -
    FROM
        ticket_details;                             -- Source table
END //

DELIMITER ;

call get_ticket_details_with_services()
```

```
DELIMITER //

CREATE FUNCTION get_complimentary_services(class_id VARCHAR(50))
RETURNS VARCHAR(3)
DETERMINISTIC  -- Add DETERMINISTIC to ensure it complies with binary logging requirements
BEGIN
    DECLARE service_status VARCHAR(3);

    IF class_id IN ('Business', 'Economy Plus') THEN
        SET service_status = 'Yes';
    ELSE
        SET service_status = 'No';
    END IF;

    RETURN service_status;
END //
DELIMITER ;
DELIMITER //
```

20. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.

```
208     DELIMITER //
209
210 ●   CREATE PROCEDURE get_first_customer_scott()
211     BEGIN
212         DECLARE done INT DEFAULT 0;               -- Variable to check if we have fetched the reco
213         DECLARE customer_id INT;                  -- Variable to hold customer ID
214         DECLARE first_name VARCHAR(50);           -- Variable to hold first name
215         DECLARE last_name VARCHAR(50);            -- Variable to hold last name
216
217         -- Declare the cursor
218         DECLARE customer_cursor CURSOR FOR
219         SELECT customer_id, first_name, last_name
220         FROM customer
221         WHERE last_name LIKE '%Scott';             -- Filter for last names ending with 'Scott'
222
223         -- Declare a CONTINUE HANDLER to handle the end of the cursor
224         DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
225
226         -- Open the cursor
227         OPEN customer_cursor;
228
229         -- Fetch the first record
230         FETCH customer_cursor INTO customer_id, first_name, last_name;
231
232         -- Check if the record was fetched
233         IF NOT done THEN
234             SELECT customer_id, first_name, last_name;  -- Display the fetched record
235         ELSE
236             SELECT 'No customer found whose last name ends with Scott' AS message;  -- Handle no result
237         END IF;
238
239         -- Close the cursor
240         CLOSE customer_cursor;
241     END //
242
243     DELIMITER ;
244
245 ●   call get_first_customer_scott()
```