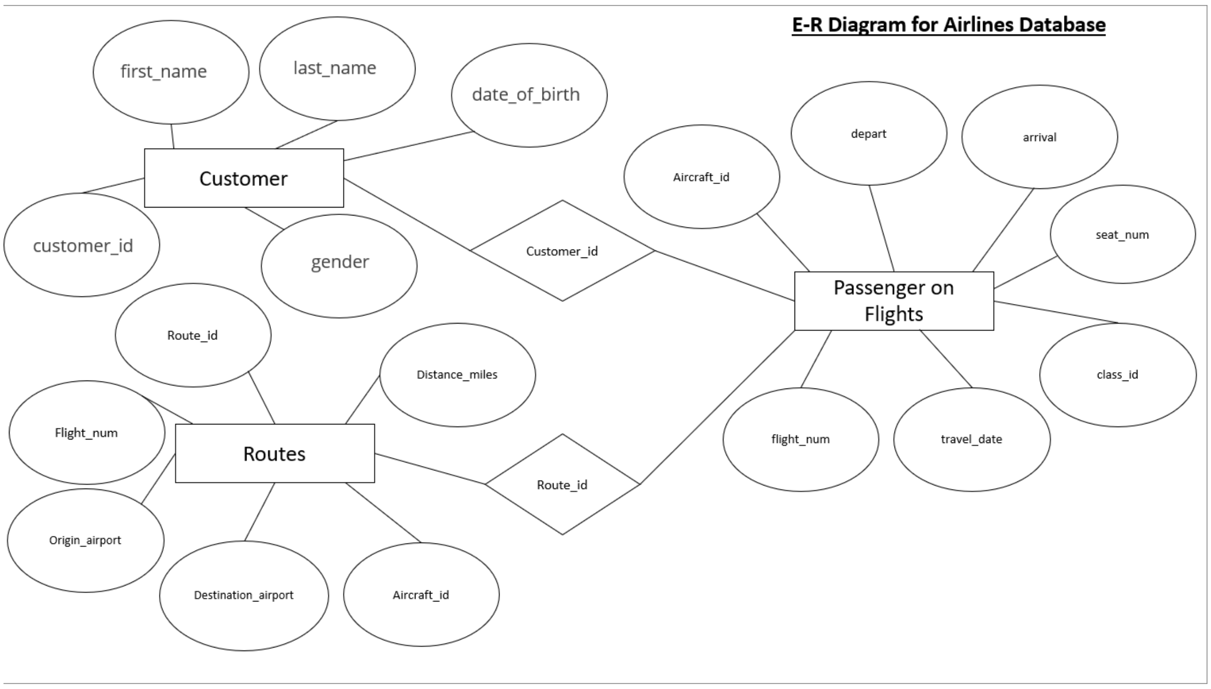
Course-End Project: Air Cargo Analysis

Operations to be performed:

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.

```
CREATE TABLE ROUTE_DETAILS
( ROUTE_ID INT PRIMARY KEY,
FLIGHT_NUM INT CHECK(FLIGHT_NUM > 0),
ORIGIN_AIRPORT VARCHAR(10),
DESTINATION_AIRPORT VARCHAR(10),
AIRCRAFT_ID VARCHAR (10),
DISTANCE_MILES INT CHECK(DISTANCE_MILES>0));
```

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.

```
SELECT C.FIRST_NAME, C.LAST_NAME,
P.ROUTE_ID, P.TRAVEL_DATE, P.FLIGHT_NUM,
R.ORIGIN_AIRPORT, R.DESTINATION_AIRPORT, R.AIRCRAFT_ID
FROM CUSTOMER AS C INNER JOIN  PASSENGERS_ON_FLIGHTS AS P
ON C.CUSTOMER_ID = P.CUSTOMER_ID
INNER JOIN ROUTES AS R
ON P.ROUTE_ID = R.ROUTE_ID
WHERE P.ROUTE_ID BETWEEN 1 AND 25;
```

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

```
SELECT CLASS_ID, COUNT(CUSTOMER_ID) AS NO_OF_PASSENGERS,
SUM(Price_per_ticket) AS TOTAL_REVENVE FROM ticket_details
WHERE CLASS_ID='Bussiness'
GROUP BY CLASS_ID;
```

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 CONCAT(C.FIRST_NAME, ' ', C.LAST_NAME) AS FULL_NAME,
T.AIRCRAFT_ID, T.CLASS_ID
FROM CUSTOMER AS C INNER JOIN ticket_details AS 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.

```
SELECT C.FIRST_NAME, C.LAST_NAME,
T.AIRCRAFT_ID, T.CLASS_ID, T.BRAND
FROM CUSTOMER AS C INNER JOIN ticket_details AS T
ON C.CUSTOMER_ID = T.CUSTOMER_ID
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.

```
SELECT C.FIRST_NAME, C.LAST_NAME,
P.CLASS_ID FROM CUSTOMER AS C INNER JOIN passengers_on_flights
AS P
ON C.CUSTOMER_ID = P.CUSTOMER_ID
WHERE CLASS_ID = (SELECT CLASS_ID FROM passengers_on_flights
GROUP BY CLASS_ID HAVING MAX(CLASS_ID)='Economy Plus');
```

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

```
SELECT IF(SUM(price_per_ticket) > 10000,
       'Crossed 10000', 'Not Crossed 10000') AS revenue_status
FROM ticket_details;
```

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

```
CREATE USER 'vennela'@'localhost' IDENTIFIED BY '123456';
GRANT ALL PRIVILEGES ON airlines.* TO 'vennela'@'localhost';
FLUSH PRIVILEGES;
```

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

```
SELECT
class_id,
Price_per_ticket,
MAX(Price_per_ticket) OVER (PARTITION BY class_id) AS
max_price_per_class
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.

```
SELECT
   C.customer_id,
   C.first_name,
   C.last_name,
   P.route_id,
   P.aircraft_id,
   P.class_id,
   P.travel_date,
   P.flight_num
FROM passengers_on_flights AS P
JOIN customer AS C ON C.customer_id = P.customer_id
WHERE P.route_id = 4;

CREATE INDEX idx_route_id ON passengers_on_flights(route_id);
```

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

```
SELECT
   C.customer_id,
   C.first_name,
   C.last_name,
   P.route_id,
   P.aircraft_id,
   P.class_id,
   P.travel_date,
   P.flight_num
FROM passengers_on_flights AS P
JOIN customer AS C ON C.customer_id = P.customer_id
WHERE P.route_id = 4;
```

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

```
SELECT
   customer_id,
   aircraft_id,
   SUM(Price_per_ticket) 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.

```
CREATE VIEW business_class_customers AS
SELECT
   customer_id,
   brand,
   class_id,
   aircraft_id
FROM ticket_details
WHERE LOWER(class_id) = 'bussiness';
```

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.

```
DELIMITER $$
CREATE PROCEDURE GetPassengersByRouteRange(IN start_route INT,
IN end_route INT)
BEGIN
  -- check if table exists
  IF (SELECT COUNT(*)
    FROM information_schema.tables
    WHERE table_schema = DATABASE()
     AND table_name = 'passengers_on_flights') = 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Table passengers_on_flights does not exist';
  ELSE
    SELECT C.first_name, C.last_name, P.*
    FROM passengers_on_flights AS P
    JOIN customer AS C ON C.customer_id = P.customer_id
    WHERE P.route_id BETWEEN start_route AND end_route;
  END IF;
END$$
DELIMITER ;
```

FOR EXECUTION:

```
CALL GetPassengersByRouteRange(1,25);
```

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 $$
CREATE PROCEDURE GetLongRoutes()
BEGIN
  SELECT *
  FROM routes
  WHERE distance_miles > 2000;
END$$
DELIMITER ;
```

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 PROCEDURE CategorizeFlightDistance()
BEGIN
  SELECT
    route_id,
    flight_num,
    distance_miles,
    CASE
      WHEN distance_miles BETWEEN 0 AND 2000 THEN 'SDT'
      WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN
'IDT'

      WHEN distance_miles > 6500 THEN 'LDT'
    END AS distance_category
  FROM routes;
END$$
DELIMITER ;
```

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*

FUNCTION:

```
DELIMITER $$
CREATE FUNCTION ComplimentaryServices(class VARCHAR(30))
RETURNS VARCHAR(3)
DETERMINISTIC
BEGIN
  RETURN CASE
    WHEN class IN ('Bussiness','Economy Plus') THEN 'Yes'
    ELSE 'No'
  END;
END$$
DELIMITER ;
```

PROCEDURE:

```
DELIMITER $$
CREATE PROCEDURE GetTicketComplimentaryInfo()
BEGIN
  SELECT
    p_date AS ticket_purchase_date,
    customer_id,
    class_id,
    ComplimentaryServices(class_id) AS complimentary_services
  FROM ticket_details;
END$$
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.

```
DELIMITER $$
CREATE PROCEDURE GetFirstScottCustomer()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE v_customer_id INT;
  DECLARE v_first_name VARCHAR(50);
  DECLARE v_last_name VARCHAR(50);
  DECLARE cur CURSOR FOR
    SELECT customer_id, first_name, last_name
    FROM customer
    WHERE last_name LIKE '%Scott';
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cur;
  FETCH cur INTO v_customer_id, v_first_name, v_last_name;
  IF done = 0 THEN
    SELECT v_customer_id AS customer_id,
        v_first_name AS first_name,
        v_last_name AS last_name;
  END IF;
  CLOSE cur;
END$$
DELIMITER ;
```

FOR EXECUTION:

```
CALL GetFirstScottCustomer();
```