Air Cargo Analysis

DESCRIPTION

Air Cargo is an aviation company that provides air transportation services for passengers and freight. Air Cargo uses its aircraft to provide different services with the help of partnerships or alliances with other airlines. The company wants to prepare reports on regular passengers, busiest routes, ticket sales details, and other scenarios to improve the ease of travel and booking for customers.

Project Objective:

You, as a DBA expert, need to focus on identifying the regular customers to provide offers, analyze the busiest route which helps to increase the number of aircraft required and prepare an analysis to determine the ticket sales details. This will ensure that the company improves its operability and becomes more customer-centric and a favorable choice for air travel.

Note: You must download the dataset from the course resource section in the LMS and create the tables to perform the above objective

Dataset description:

Customer: Contains the information of customers

- customer_id – ID of the customer
- first_name – First name of the customer
- last_name – Last name of the customer
- date_of_birth – Date of birth of the customer
- gender – Gender of the customer

passengers_on_flights: Contains information about the travel details

- aircraft_id – ID of each aircraft in a brand
- route_id – Route ID of from and to location
- customer_id – ID of the customer
- depart – Departure place from the airport
- arrival – Arrival place in the airport
- seat_num – Unique seat number for each passenger
- class_id – ID of travel class
- travel_date – Travel date of each passenger
- flight_num – Specific flight number for each route

ticket_details: Contains information about the ticket details

- p_date – Ticket purchase date
- customer_id – ID of the customer
- aircraft_id – ID of each aircraft in a brand

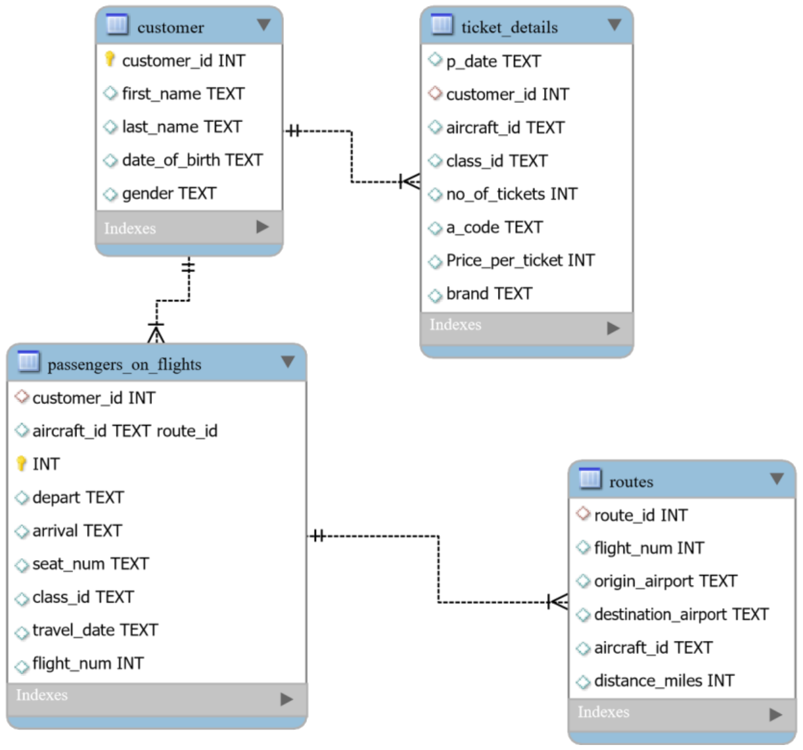
- class_id – ID of travel class
- no_of_tickets – Number of tickets purchased
- a_code – Code of each airport
- price_per_ticket – Price of a ticket
- brand – Aviation service provider for each aircraft

routes: Contains information about the route details

- Route_id – Route ID of from and to location
- Flight_num – Specific fight number for each route
- Origin_airport – Departure location
- Destination_airport – Arrival location
- Aircraft_id – ID of each aircraft in a brand
- Distance_miles – Distance between departure and arrival location

Following operations should be performed:

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



2. Write a query to create 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 TEBLE route_details (route_id INT PRIMARY KEY UNIQUE, flight_num VARCHAR(30) CHECK (flight_num like '[A-Z]{2}[0-9]{4}'), origin_airport VARCHAR(30) NOT NULL, destination_airport VARCHAR(30) NOT NULL, aircraft_id INT NOT NULL, distance_mile FLOAT CHECK (distance_mile > 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 customer_id from passengers_on_flights where route_id BETWEEN '01' and '25';

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

   - SELECT COUNT (DISTINCT customer_id) as 'Number of Passengers',
     SUM(Price_per_ticket) 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 customer.customer_id, customer.first_name, customer.last_name from customer
     INNER JOIN ticket_details
     on customer.customer_id= ticket_details.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 customer.first_name, customer.last_name from customer
     INNER JOIN ticket_details
     on customer.customer_id= ticket_details.customer_id where
     ticket_details.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 COUNT(DISTINCT customer_id) as 'Number of Passengers'
     FROM passengers_on_flights
     WHERE class_id= 'Economy Plus'
     GROUP BY flight_num
     HAVING COUNT(DISTINCT customer_id)>0;

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, 'Revenue has crossed 10000',
     'Revenue is below 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 'new_user'@'localhost' IDENTIFIED BY 'password'; GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON database_name.* TO 'new_user'@'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
  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 * FROM passengers_on_flights WHERE route_id = 4
      ORDER BY customer_id
      LIMIT 100;
      ```

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

    -
      ```
      EXPLAIN SELECT * FROM passengers_on_flights WHERE 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.customer_id, customer.first_name, customer.last_name,
      ticket_details.brand FROM customer
      inner join ticket_details
      on customer.customer_id= ticket_details.customer_id WHERE
      ticket_details.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 getPassengerDetails(IN start_route INT, IN end_route INT)
      ```

```
BEGIN
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
   BEGIN
   SELECT 'Table does not exist' AS 'Error';
   END;
   SELECT * FROM passengers_on_flights WHERE route_id BETWEEN start_route
AND end_route;
END $$
DELIMITER ;
```

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 $$
     CREATE PROCEDURE getLongDistanceRoutes()
     BEGIN
        SELECT * FROM routes WHERE distance_miles > 2000;
     END $$
     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 longdistance travel (LDT) for >6500.

   -
     ```
     DELIMITER $$
     CREATE PROCEDURE groupDistance()
     BEGIN
        SELECT flight_num,
        CASE
           WHEN distance_miles >= 0 AND distance_miles <= 2000 THEN 'Short
     Distance Travel (SDT)'
           WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN
     'Intermediate Distance Travel (IDT)'
           WHEN distance_miles > 6500 THEN 'Long Distance Travel (LDT)'
        END as 'Travel Distance'
        FROM routes;
     END $$
     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*

-
  ```
  DELIMITER $$
  CREATE PROCEDURE extractTicketDetails()
  BEGIN
    SELECT purchase_date, customer_id, class_id,
    (CASE
      WHEN class_id IN ('Business', 'Economy Plus') THEN 'Yes'
      ELSE 'No'
    END) as 'Complimentary Services'
    FROM ticket_details;
  END $$
  DELIMITER ;
  ```