**Air Cargo Analysis**

**PROBLEM 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.

TASKS TO 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.
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.
4. Write a query to identify the number of passengers and total revenue in business class from the ticket\_details table.
5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.
6. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket\_details tables.
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.
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.
9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket\_details table.
10. Write a query to create and grant access to a new user to perform operations on a database.
11. Write a query to find the maximum ticket price for each class using window functions on the ticket\_details table.
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.
13. For the route ID 4, write a query to view the execution plan of the passengers\_on\_flights table.
14. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.
15. Write a query to create a view with only business class customers along with the brand of airlines.
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.
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.
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.
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*

    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.

SOLUTION

CREATE DATABASE Air\_Cargo\_Analysis;

USE Air\_Cargo\_Analysis;

SET sql\_mode=(SELECT REPLACE(@@sql\_mode,'ONLY\_FULL\_GROUP\_BY',''));

CREATE TABLE route\_details (route\_id int NOT NULL,

flight\_num int NOT NULL,

origin\_airport varchar(20),

destination\_airport varchar(20),

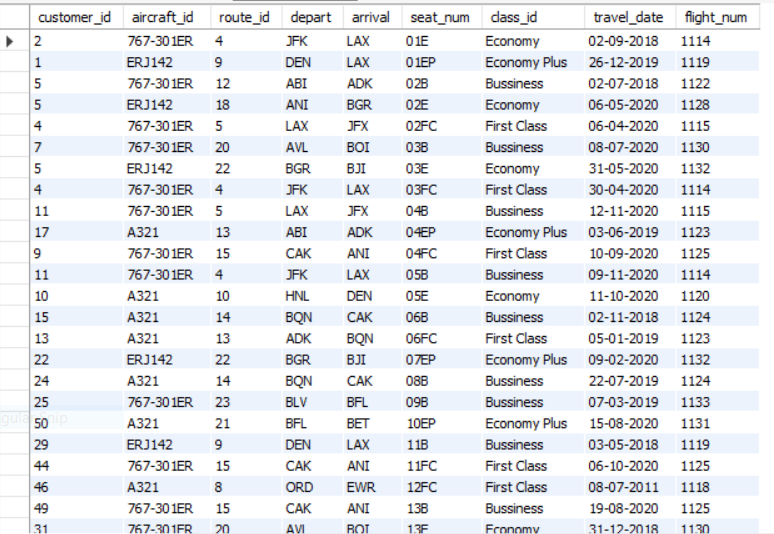
aircraft\_id varchar(10),

distance int NOT NULL,

UNIQUE(route\_id), CHECK (distance>0));

SELECT\*FROM passengers\_on\_flights

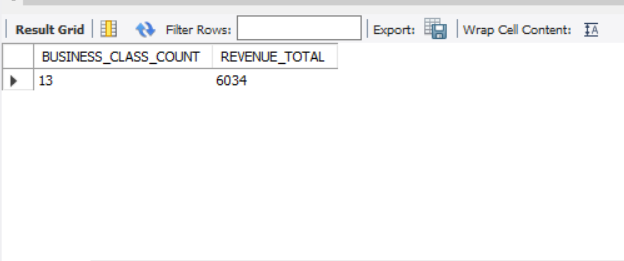
WHERE route\_id BETWEEN 1 AND 25;



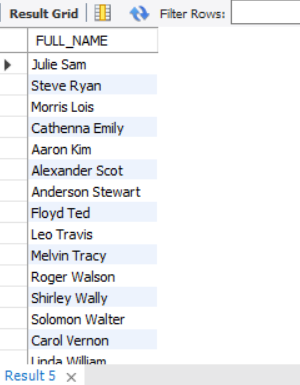
SELECT COUNT(class\_id = 'Business') AS BUSINESS\_CLASS\_COUNT,

SUM(no\_of\_tickets\*price\_per\_ticket) AS REVENUE\_TOTAL FROM ticket\_details

WHERE class\_id = 'Bussiness';



SELECT CONCAT(first\_name, " ", last\_name) AS FULL\_NAME FROM customer;



SELECT customer\_id, CONCAT(first\_name, " " , last\_name) AS NAME,

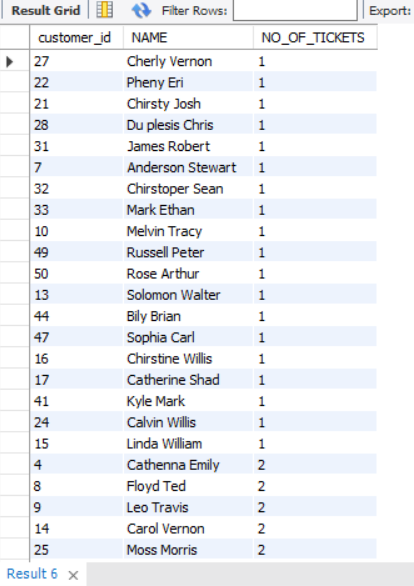
COUNT(no\_of\_tickets) AS NO\_OF\_TICKETS

FROM customer

JOIN ticket\_details USING (customer\_id)

GROUP BY customer\_id, NAME

ORDER BY NO\_OF\_TICKETS;



SELECT customer\_id, first\_name, last\_name FROM customer

JOIN ticket\_details USING(customer\_id)

WHERE brand = 'Emirates';

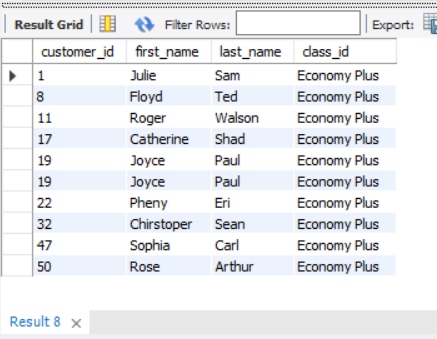


SELECT customer.customer\_id, customer.first\_name, customer.last\_name, passengers\_on\_flights.class\_id

FROM customer

JOIN passengers\_on\_flights ON customer.customer\_id = passengers\_on\_flights.customer\_id

WHERE passengers\_on\_flights.class\_id = 'Economy Plus';



SELECT customer.customer\_id, customer.first\_name, customer.last\_name, passengers\_on\_flights.class\_id

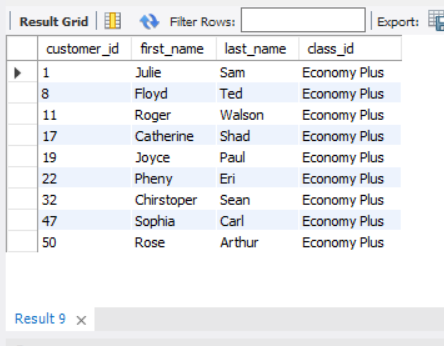
FROM customer

JOIN passengers\_on\_flights ON customer.customer\_id = passengers\_on\_flights.customer\_id

GROUP BY customer\_id

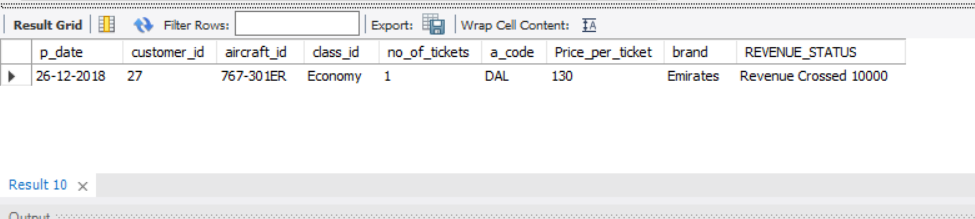
HAVING class\_id = 'Economy Plus'

ORDER BY customer\_id;



SELECT \* , IF(SUM(no\_of\_tickets \* Price\_per\_ticket)>10000,'Revenue Crossed 10000','Revenue Less Than 10000')

AS REVENUE\_STATUS FROM ticket\_details;



CREATE USER 'new\_user'@'localhost' IDENTIFIED BY 'new\_password';

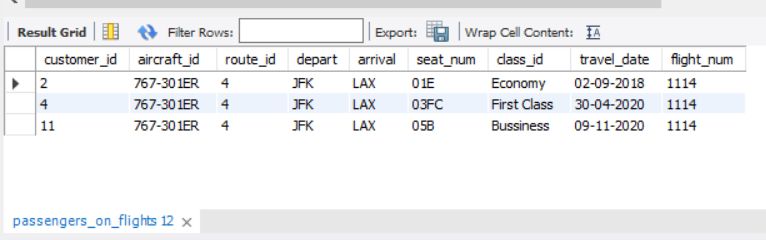
GRANT ALL ON Air\_Cargo\_Analysis.\* TO 'new\_user'@'localhost';

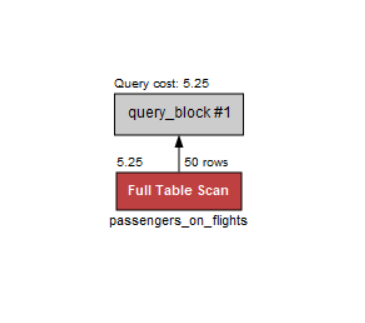
SELECT customer\_id,class\_id,brand , MAX(Price\_per\_ticket) OVER (PARTITION BY class\_id) AS max\_price FROM ticket\_details;



SELECT \* FROM passengers\_on\_flights WHERE route\_id = 4 ;

SELECT \* FROM passengers\_on\_flights HAVING route\_id = 4 ;



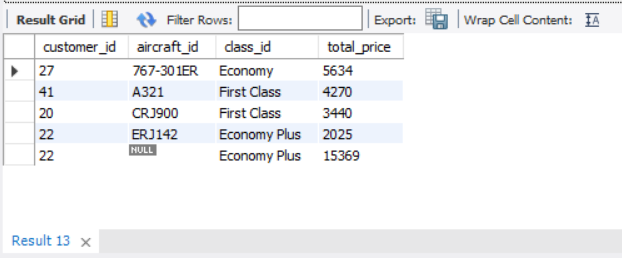


SELECT customer\_id, aircraft\_id , class\_id , sum(no\_of\_tickets \* Price\_per\_ticket)

AS total\_price

FROM ticket\_details

GROUP BY aircraft\_id WITH ROLLUP ;



DROP VIEW business\_class ;

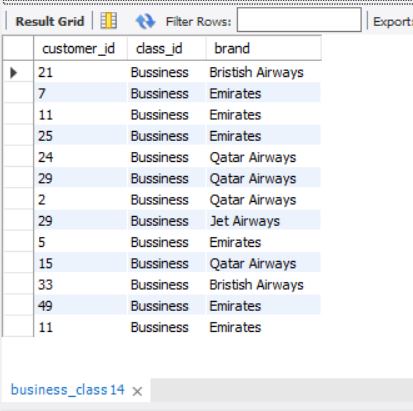
CREATE VIEW business\_class AS

SELECT customer\_id, class\_id, brand

FROM ticket\_details

WHERE class\_id = 'bussiness' ;

SELECT \* FROM business\_class ;



DROP PROCEDURE passenger\_details;

DELIMITER &&

CREATE PROCEDURE passenger\_details(route\_id INT)

BEGIN

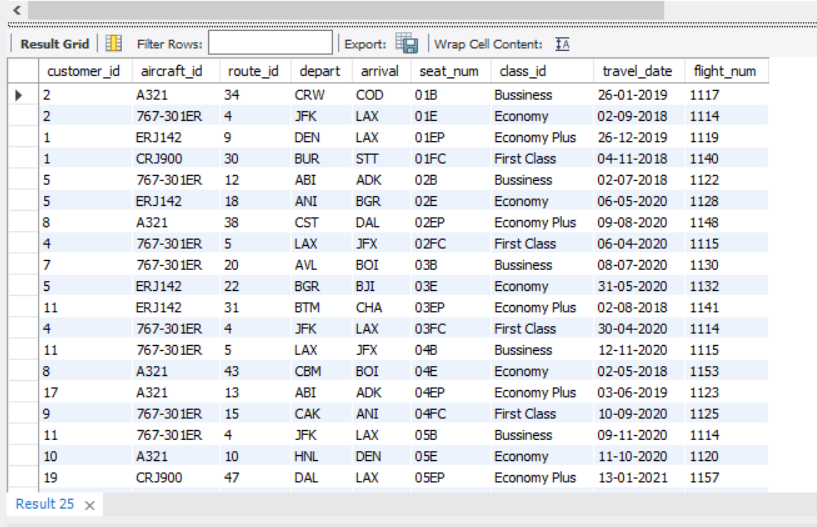
SELECT \* FROM passengers\_on\_flights

WHERE route\_id BETWEEN

1 AND 50 ORDER BY route\_id ;

END &&

CALL passenger\_details() ;



DROP PROCEDURE travelled\_distance() IF EXIST ;

DELIMITER //

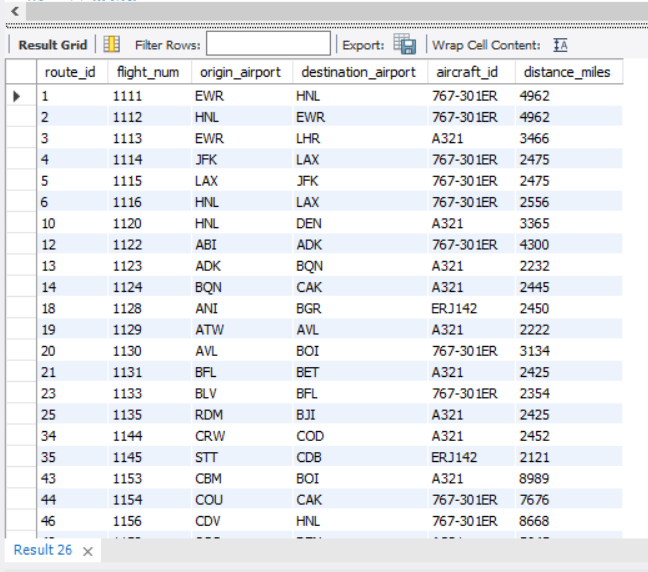
CREATE PROCEDURE travelled\_distance()

BEGIN

SELECT \* FROM routes WHERE distance\_miles > 2000 ;

END //

CALL travelled\_distance() ;



DROP PROCEDURE travel\_category;

DELIMITER //

CREATE PROCEDURE travel\_category(IN distance int , OUT category VARCHAR(40))

BEGIN

SELECT distance\_miles INTO distance

FROM routes

WHERE routes.distance\_miles = distance;

IF distance >= 0 AND distance <=2000 THEN

SET category = 'SHORT DISTANCE TRAVEL';

ELSEIF distance >= 2000 AND distance <=6500 THEN

SET category = 'INTERMEDIATE DISTANCE TRAVEL';

ELSEIF distance > 6000 THEN

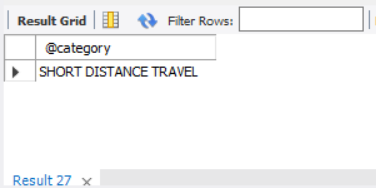
SET category = 'LONG DISTANCE TRAVEL';

END IF ;

END //

CALL travel\_category(1523 , @category) ;

SELECT @category;



DROP FUNCTION IF EXISTS Complementary\_Services

DELIMITER //

CREATE FUNCTION Complementary\_Services(class\_id VARCHAR(40))

RETURNS VARCHAR(10) DETERMINISTIC

BEGIN

DECLARE SERVICE VARCHAR(20);

IF class\_id = 'Economy Plus' OR 'Business' THEN

SET SERVICE = 'YES';

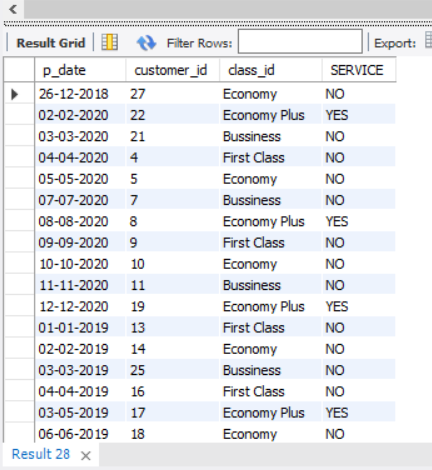
ELSE SET SERVICE = 'NO';

END IF;

RETURN SERVICE;

END //

SELECT p\_date,customer\_id,class\_id, Complementary\_Services(class\_id) AS SERVICE FROM ticket\_details;



DROP PROCEDURE IF EXISTS First\_Scott

DELIMITER //

CREATE PROCEDURE First\_Scott()

BEGIN

DECLARE a VARCHAR(50);

DECLARE b VARCHAR(50);

DECLARE cursor\_1 CURSOR FOR SELECT first\_name, last\_name FROM customer

WHERE last\_name = 'Scott';

OPEN cursor\_1;

REPEAT FETCH cursor\_1 INTO a,b;

UNTIL b=0

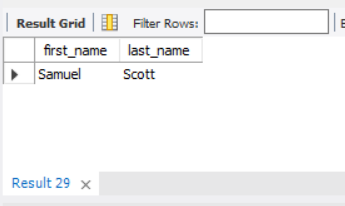
END REPEAT;

SELECT a AS first\_name , b as last\_name;

CLOSE cursor\_1;

END //

CALL First\_Scott()



\*\*\*