**SQL PROJECT-2**

**Air Cargo Analysis**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**.sql file: **

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUBMITTED BY:**

**SAUMYA GUREJA**

**Air Cargo Analysis**

Course-end Project 2

**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

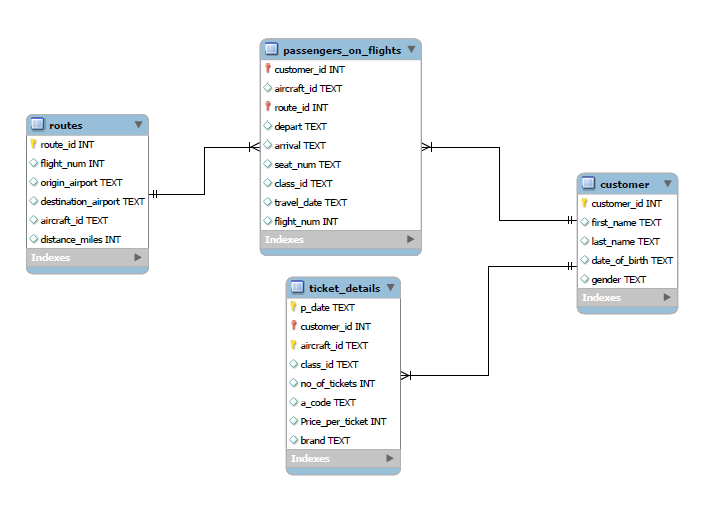
**routes:** Contains information about the route details

* 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

Datasets: <air_cargo_datasets.zip>



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



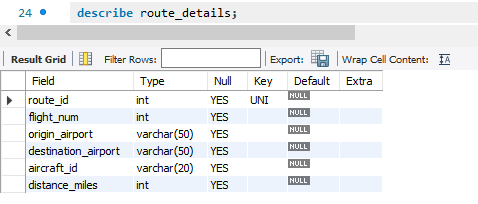
Pdf link: [er diagram air cargo\_pdf\_.pdf](er%20diagram%20air%20cargo_pdf_.pdf) 

---------------------------------------------------------------------------------------------------------------------------------

**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 TABLE route\_details ( route\_id int UNIQUE, flight\_num int CHECK(flight\_num<>0), origin\_airport varchar(200), destination\_airport varchar(200), aircraft\_id varchar(200), distance\_miles int CHECK(distance\_miles>0) );

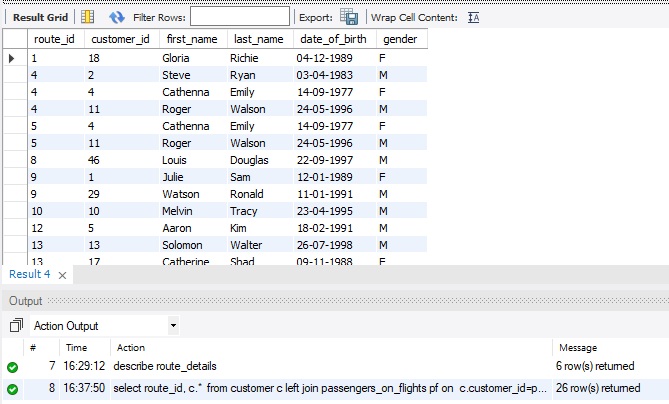
DESCRIBE route\_details;



---------------------------------------------------------------------------------------------------------------------------------

**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.**

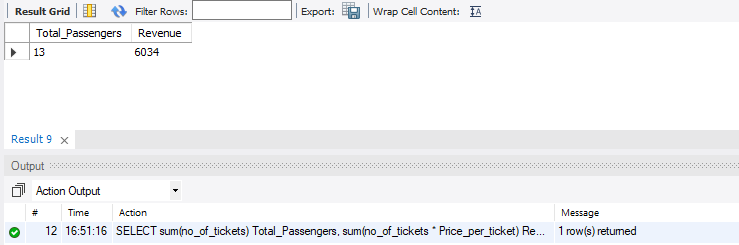
SELECT route\_id, c.\* FROM customer c LEFT JOIN passengers\_on\_flights pf ON c.customer\_id=pf.customer\_id WHERE route\_id BETWEEN 01 AND 25 ORDER BY route\_id asc;



---------------------------------------------------------------------------------------------------------------------------------

**4.Write a query to identify the number of passengers and total revenue in business class from the ticket\_details table.**

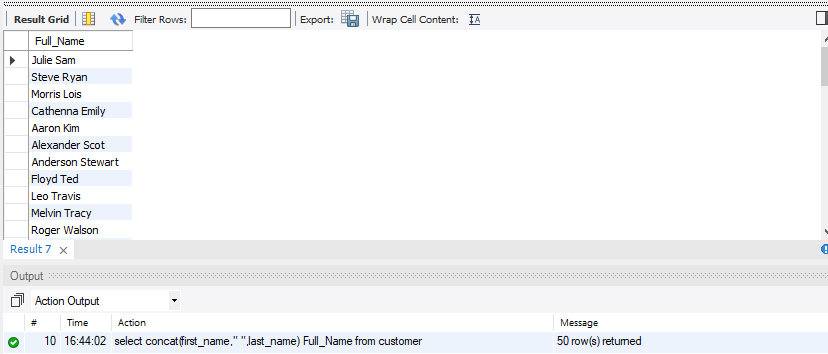
SELECT sum(no\_of\_tickets) Total\_Passengers, sum(no\_of\_tickets \* Price\_per\_ticket) Revenue FROM ticket\_details WHERE class\_id="Bussiness";



---------------------------------------------------------------------------------------------------------------------------------

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

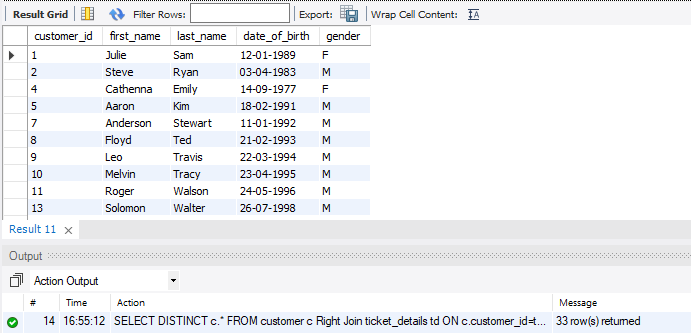
SELECT concat(first\_name," ",last\_name) Full\_Name FROM customer;



---------------------------------------------------------------------------------------------------------------------------------

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

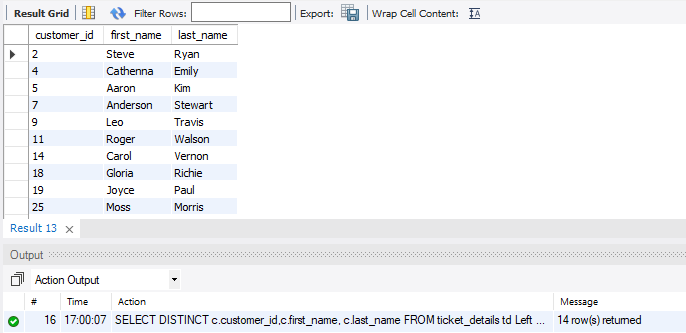
SELECT DISTINCT c.\* FROM customer c Right Join ticket\_details td ON c.customer\_id = td.customer\_id ORDER BY customer\_id asc;



---------------------------------------------------------------------------------------------------------------------------------

**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.**

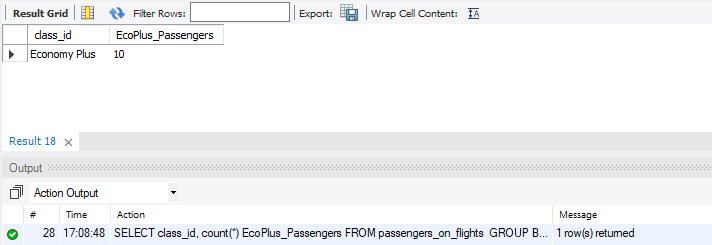
SELECT DISTINCT c.customer\_id, c.first\_name, c.last\_name FROM ticket\_details td LEFT JOIN customer c ON td.customer\_id = c.customer\_id WHERE brand ="Emirates" ORDER BY customer\_id asc;



---------------------------------------------------------------------------------------------------------------------------------

**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.**

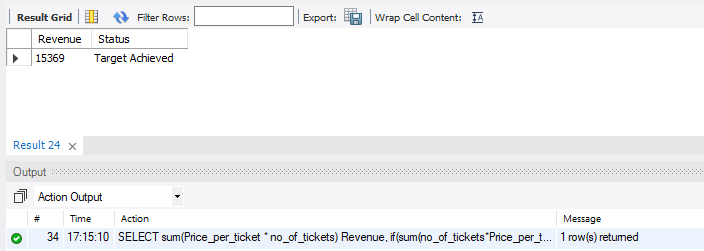
SELECT class\_id, count(\*) EcoPlus\_Passengers FROM passengers\_on\_flights GROUP BY class\_id HAVING class\_id = "Economy Plus";



---------------------------------------------------------------------------------------------------------------------------------

**9.Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket\_details table.**

SELECT sum(Price\_per\_ticket \* no\_of\_tickets) Revenue, IF(sum(no\_of\_tickets \* Price\_per\_ticket)>10000, "Target Achieved","Target Not Achieved") Status FROM ticket\_details;

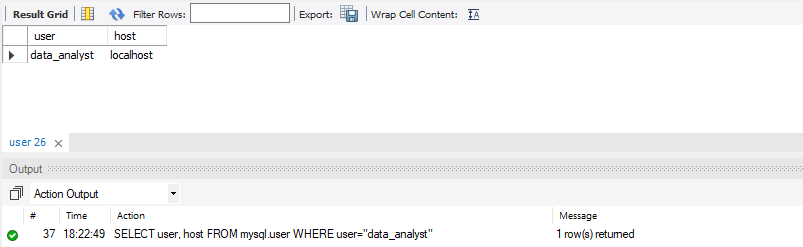


---------------------------------------------------------------------------------------------------------------------------------

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

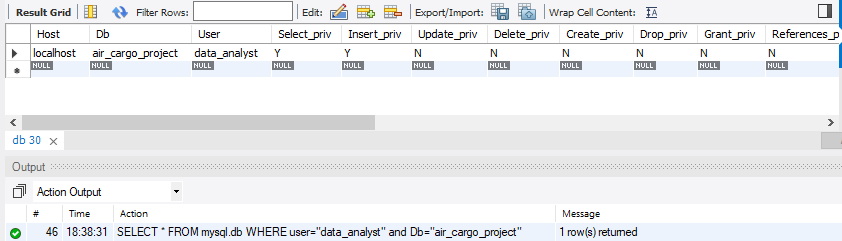
CREATE USER "data\_analyst"@"localhost" IDENTIFIED BY "data\_analyst";

SELECT user, host FROM mysql.user WHERE user="data\_analyst";



GRANT select, insert ON air\_cargo\_project.\* TO "data\_analyst"@"localhost";

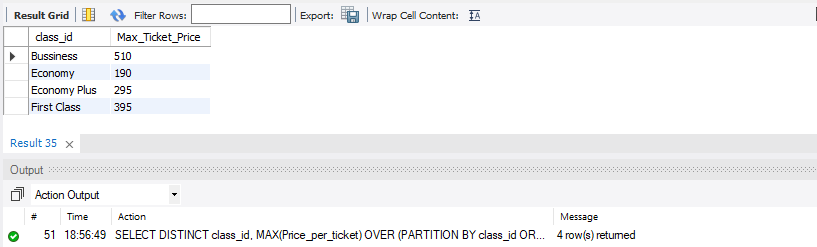
SELECT \* FROM mysql.db WHERE user="data\_analyst" and Db="air\_cargo\_project";



---------------------------------------------------------------------------------------------------------------------------------

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

SELECT DISTINCT class\_id, MAX(Price\_per\_ticket) OVER (PARTITION BY class\_id ORDER BY class\_id asc) Max\_Ticket\_Price FROM ticket\_details;

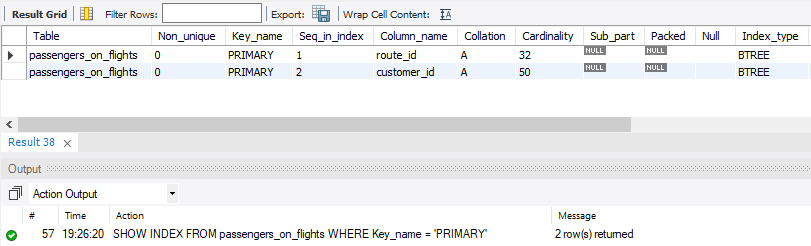


---------------------------------------------------------------------------------------------------------------------------------

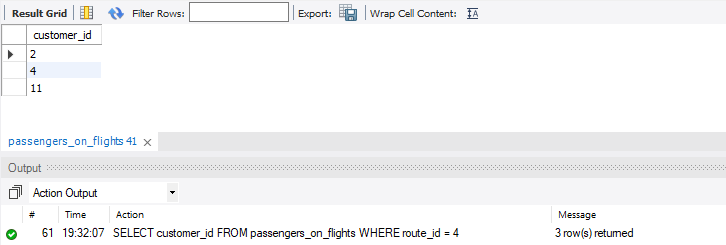
**12.Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers\_on\_flights table.**

ALTER TABLE passengers\_on\_flights ADD PRIMARY KEY (route\_id, customer\_id);

SHOW INDEX FROM passengers\_on\_flights WHERE Key\_name = 'PRIMARY';



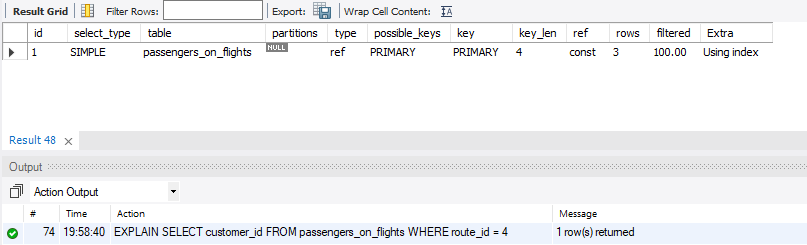
SELECT customer\_id FROM passengers\_on\_flights WHERE route\_id=4;



---------------------------------------------------------------------------------------------------------------------------------

**13.For the route ID 4, write a query to view the execution plan of the passengers\_on\_flights table.**

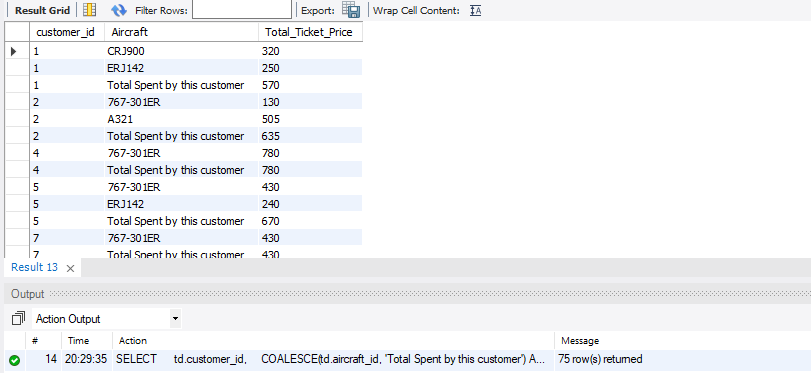
EXPLAIN SELECT customer\_id FROM passengers\_on\_flights WHERE route\_id = 4;



---------------------------------------------------------------------------------------------------------------------------------

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

SELECT td.customer\_id, COALESCE(td.aircraft\_id, 'Total Spent by this customer') AS Aircraft, SUM(td.no\_of\_tickets \* td.Price\_per\_ticket) Total\_Ticket\_Price FROM ticket\_details td GROUP BY td.customer\_id, td.aircraft\_id WITH ROLLUP;



---------------------------------------------------------------------------------------------------------------------------------

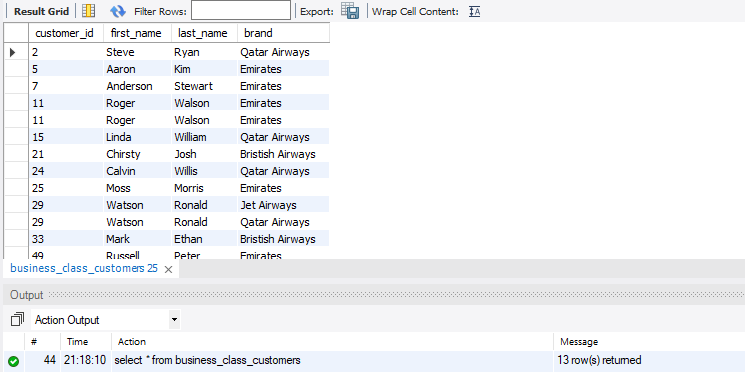
**15.Write a query to create a view with only business class customers along with the brand of airlines.**

CREATE VIEW business\_class\_customers AS

SELECT t.customer\_id, c.first\_name,c.last\_name,t.brand FROM ticket\_details t, customer c

WHERE t.customer\_id = c.customer\_id and t.class\_id = 'Bussiness';

SELECT \* FROM business\_class\_customers;



---------------------------------------------------------------------------------------------------------------------------------

**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 passengers\_between\_route\_range(IN start\_route\_id INT, IN end\_route\_id INT)

BEGIN

IF NOT EXISTS (SELECT \* FROM information\_schema.tables

WHERE table\_name = 'passengers\_on\_flights' AND table\_schema = DATABASE()) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error: Table passengers\_on\_flights does not exist.';

ELSE

SELECT pf.customer\_id, c.first\_name, c.last\_name, pf.route\_id, pf.depart, pf.arrival

FROM passengers\_on\_flights pf

JOIN customer c ON pf.customer\_id = c.customer\_id

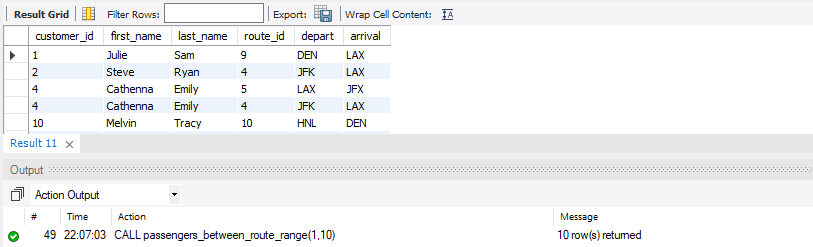
WHERE pf.route\_id BETWEEN start\_route\_id AND end\_route\_id;

END IF;

END$$

DELIMITER ;

CALL passengers\_between\_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 PROCEDURE long\_distance\_route()

BEGIN

IF NOT EXISTS (SELECT \* FROM information\_schema.tables

WHERE table\_name = 'routes'

AND table\_schema = DATABASE()) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error: Table routes does not exist.';

ELSE

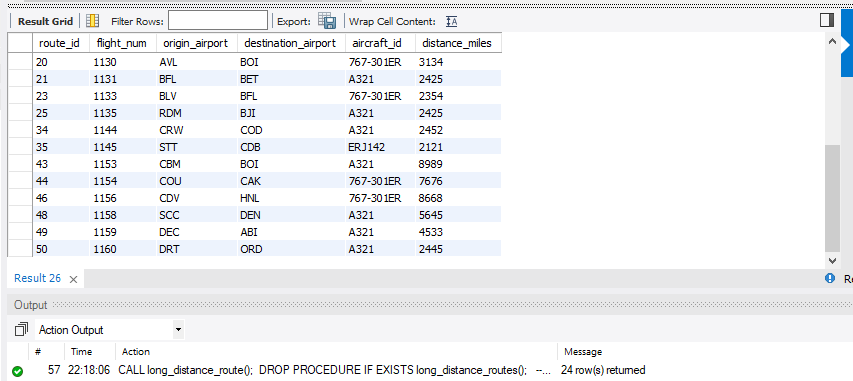
SELECT \* FROM routes

WHERE distance\_miles > 2000;

END IF;

END //

CALL long\_distance\_route();



---------------------------------------------------------------------------------------------------------------------------------

**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 flight\_distance\_category()

BEGIN

IF NOT EXISTS (SELECT \* FROM information\_schema.tables

WHERE table\_name = 'routes'

AND table\_schema = DATABASE()) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error: Table routes does not exist.';

ELSE

SELECT

flight\_num,

CASE

WHEN distance\_miles >= 0 AND distance\_miles <= 2000 THEN 'Short Distance Travel'

WHEN distance\_miles > 2000 AND distance\_miles <= 6500 THEN 'Intermediate Distance Travel'

WHEN distance\_miles > 6500 THEN 'Long Distance Travel'

END AS distance\_category,

distance\_miles

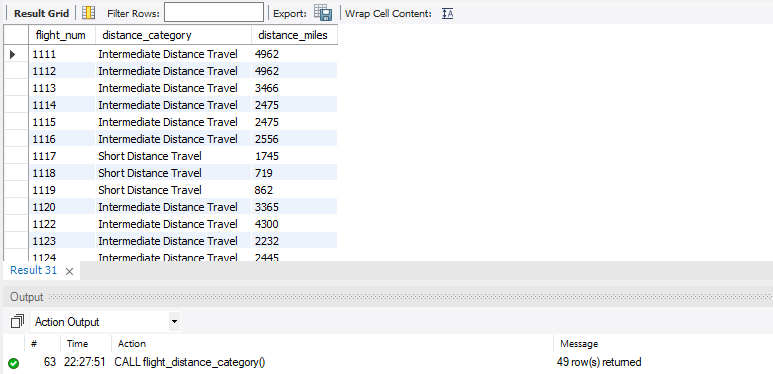
FROM routes;

END IF;

END //

DELIMITER ;

CALL flight\_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 FUNCTION complimentary\_services(class\_id VARCHAR(50))

RETURNS VARCHAR(3)

DETERMINISTIC

BEGIN

DECLARE service VARCHAR(3);

IF class\_id IN ('Business', 'Economy Plus') THEN

SET service = 'Yes';

ELSE

SET service = 'No';

END IF;

RETURN service;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE ticket\_details\_with\_services()

BEGIN

IF NOT EXISTS (SELECT \* FROM information\_schema.tables

WHERE table\_name = 'ticket\_details'

AND table\_schema = DATABASE()) THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error: Table ticket\_details does not exist.';

ELSE

SELECT

p\_date AS ticket\_purchase\_date,

customer\_id,

class\_id,

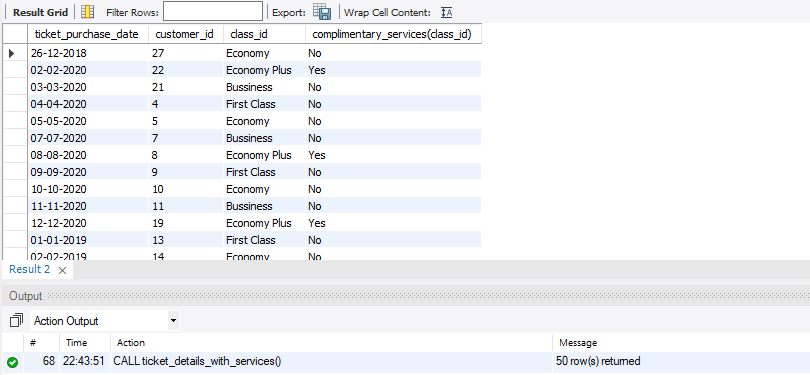
complimentary\_services(class\_id)

FROM ticket\_details;

END IF;

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 last\_name\_scott()

BEGIN

DECLARE cust\_id INT;

DECLARE fst\_name VARCHAR(128);

DECLARE lst\_name VARCHAR(128);

DECLARE done INT DEFAULT 0;

DECLARE find\_scott 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 find\_scott;

read\_loop: LOOP

FETCH find\_scott INTO cust\_id, fst\_name, lst\_name;

IF done = 1 THEN

LEAVE read\_loop;

END IF;

SELECT cust\_id AS CustomerID,

fst\_name AS FirstName,

lst\_name AS LastName;

END LOOP;

CLOSE find\_scott;

END //

DELIMITER ;

