-- Air Cargo Analysis --

create database airlines;

use airlines;

select \* from customer;

select \* from passengers\_on\_flights;

select \* from routes;

select \* from ticket\_details;

ALTER TABLE `airlines`.`customer`

MODIFY COLUMN `customer\_id` VARCHAR(10) NOT NULL,

ADD PRIMARY KEY (`customer\_id`);

ALTER TABLE `airlines`.`routes`

MODIFY COLUMN `route\_id` VARCHAR(10) NOT NULL,

ADD PRIMARY KEY (`route\_id`);

-- 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 PRIMARY KEY,

flight\_num VARCHAR(4) CHECK (flight\_num >= '1111' AND flight\_num <= '1157'),

origin\_airport VARCHAR(255),

destination\_airport VARCHAR(255),

aircraft\_id INT,

distance\_miles DECIMAL(10, 2) CHECK (distance\_miles > 0),

UNIQUE (route\_id)

);

select \* from route\_details;

-- 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 DISTINCT c.customer\_id, c.first\_name, c.last\_name

FROM passengers\_on\_flights p

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

WHERE p.route\_id BETWEEN 1 AND 25;

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

SELECT class\_id, COUNT(\*) AS number\_of\_passengers, SUM(price\_per\_ticket \* no\_of\_tickets) AS total\_revenue

FROM ticket\_details

WHERE class\_id = 'Business'

GROUP BY class\_id;

-- 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;

-- 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;

-- 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

FROM ticket\_details t

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

WHERE t.brand = 'Emirates';

-- 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.customer\_id, c.first\_name, c.last\_name

FROM passengers\_on\_flights p

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

WHERE p.class\_id = 'Economy Plus'

GROUP BY c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(\*) > 0;

-- 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 \* no\_of\_tickets) > 10000, 'Yes', 'No') AS revenue\_crossed\_10000

FROM ticket\_details;

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

CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';

GRANT SELECT, INSERT, UPDATE, DELETE ON dbname.\* TO 'newuser'@'localhost';

-- 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) AS max\_ticket\_price

FROM ticket\_details;

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

CREATE INDEX idx\_route\_id ON passengers\_on\_flights (route\_id);

-- 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;

-- 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 \* no\_of\_tickets) AS total\_price

FROM ticket\_details

GROUP BY customer\_id, aircraft\_id WITH ROLLUP;

-- 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 c.customer\_id, c.first\_name, c.last\_name, t.brand AS airline\_brand

FROM customer c

JOIN ticket\_details t ON c.customer\_id = t.customer\_id

WHERE t.class\_id = 'Business';

select \* from business\_class\_customers;

-- 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\_id INT, IN end\_route\_id INT)

BEGIN

DECLARE table\_exists INT;

-- Check if the table exists

SELECT COUNT(\*) INTO table\_exists

FROM information\_schema.tables

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

IF table\_exists = 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Table passengers\_on\_flights does not exist';

ELSE

SELECT \*

FROM passengers\_on\_flights

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

END IF;

END //

DELIMITER ;

CALL GetPassengersByRouteRange(1, 25);

-- 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 ;

CALL GetLongDistanceRoutes();

-- 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 GroupDistanceCategories()

BEGIN

SELECT

\*,

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 Distance\_Category

FROM routes;

END //

DELIMITER ;

CALL GroupDistanceCategories();

-- 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 GetComplimentaryServices(class\_id VARCHAR(20)) RETURNS VARCHAR(3)

DETERMINISTIC

NO SQL

BEGIN

DECLARE complimentary VARCHAR(3);

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

SET complimentary = 'Yes';

ELSE

SET complimentary = 'No';

END IF;

RETURN complimentary;

END //

CREATE PROCEDURE ExtractTicketDetailsWithComplimentaryServices()

BEGIN

SELECT p\_date, customer\_id, class\_id, GetComplimentaryServices(class\_id) AS ComplimentaryServices

FROM ticket\_details;

END //

DELIMITER ;

CALL ExtractTicketDetailsWithComplimentaryServices();

-- 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 GetCustomerByLastName()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE last\_name VARCHAR(50);

DECLARE cur CURSOR FOR

SELECT last\_name FROM customer WHERE last\_name LIKE '%Scott';

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

FETCH cur INTO last\_name;

IF NOT done THEN

SELECT \* FROM customer WHERE last\_name = last\_name LIMIT 1;

END IF;

CLOSE cur;

END //

DELIMITER ;

CALL GetCustomerByLastName();