Project SQL Script

*The SQL script for the project -Air Cargo Analysis with the operations to be executed has been attached below respectively:*

CREATE schema Air\_Cargo\_Project;

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

use Air\_Cargo\_Project;

*/\**

*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 PRIMARY KEY,

flight\_num VARCHAR(10) CHECK (flight\_num LIKE 'FL%'),

origin\_airport VARCHAR(50),

destination\_airport VARCHAR(50),

aircraft\_id INT,

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

UNIQUE (route\_id)

);

*/\**

*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 DISTINCT c.first\_name, c.last\_name

FROM Customer c

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

JOIN routes r ON pof.route\_id = r.route\_id

WHERE r.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(\*) AS total\_passengers,

SUM(Price\_per\_ticket) AS total\_revenue

FROM ticket\_details

WHERE class\_id = 'business';

*/\**

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

FROM customer c

INNER JOIN ticket\_details td ON c.customer\_id = td.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

FROM customer c

INNER JOIN ticket\_details td ON c.customer\_id = td.customer\_id

WHERE td.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.customer\_id, c.first\_name, c.last\_name

FROM customer c

inner JOIN passengers\_on\_flights pof ON c.customer\_id = pof.customer\_id

WHERE pof.class\_id = 'Economy Plus'

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

HAVING COUNT(\*) > 0;

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

SELECT

CASE WHEN SUM(Price\_per\_ticket) > 10000 THEN 'Revenue exceeded 10,000'

ELSE 'Revenue is not exceeding 10,000' END 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 ALL PRIVILEGES ON Local\_instance\_3306.\* 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.*

WITH RankedTicketDetails AS (

SELECT

class\_id Class,

Price\_per\_ticket Maximum\_Price,

ROW\_NUMBER() OVER (PARTITION BY class\_id ORDER BY Price\_per\_ticket DESC) AS row\_num

FROM ticket\_details

)

SELECT Class, Maximum\_Price

FROM RankedTicketDetails

WHERE row\_num = 1;

*/\**

*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.*

*\*/*

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

SELECT \*

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 \*

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

IFNULL(customer\_id, 'Grand Total') AS customer\_id,

IFNULL(aircraft\_id, 'Total per Customer') AS 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.*

SELECT

td.customer\_id,

td.class\_id,

brand AS airline\_brand

FROM

ticket\_details td

WHERE

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 //

drop procedure if exists GetPassengerDetailsByRouteRange;

CREATE PROCEDURE GetPassengerDetailsByRouteRange(

IN min\_route\_id INT,

IN max\_route\_id INT

)

BEGIN

DECLARE table\_exists INT;

-- Check if the table exists

SELECT COUNT(\*) INTO table\_exists

FROM information\_schema.tables

WHERE table\_name = 'passengers\_on\_flights';

IF table\_exists = 0 THEN

SIGNAL SQLSTATE '45000'

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

ELSE

-- If the table exists, retrieve passenger details within the specified route range

SET @sql = CONCAT('

SELECT \*

FROM passengers\_on\_flights

WHERE route\_id BETWEEN ', min\_route\_id, ' AND ', max\_route\_id);

PREPARE stmt FROM @sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END IF;

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 //

DROP PROCEDURE IF EXISTS GetRoutesByDistance;

CREATE PROCEDURE GetRoutesByDistance()

BEGIN

SELECT \*

FROM routes

WHERE distance\_miles > 2000;

END //

DELIMITER ;

CALL GetRoutesByDistance();

*/\**

*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 //

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 distance\_category

FROM routes;

//

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 //

drop function if exists IsComplimentaryService;

CREATE FUNCTION IsComplimentaryService(class\_id VARCHAR(50))

RETURNS VARCHAR(3)

deterministic

BEGIN

DECLARE result VARCHAR(3);

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

SET result = 'Yes';

ELSE

SET result = 'No';

END IF;

RETURN result;

END;

//

DELIMITER ;

DELIMITER //

drop procedure if exists GetTicketDetailsWithComplimentaryService;

CREATE PROCEDURE GetTicketDetailsWithComplimentaryService()

BEGIN

SELECT

p\_date Purchase\_date,

customer\_id,

class\_id,

IsComplimentaryService(class\_id) AS complimentary\_service

FROM ticket\_details;

END;

//

DELIMITER ;

CALL GetTicketDetailsWithComplimentaryService();

*/\**

*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 //

drop procedure if exists first\_record;

create procedure first\_record()

begin

DECLARE DONE INT DEFAULT 0;

DECLARE c\_id int;

declare f\_name varchar(50);

declare l\_name varchar(50);

declare cursor\_1 cursor for

select

customer\_id,first\_name,last\_name

from customer

where last\_name like '%Scott'

limit 1;

declare continue handler for not found set done =1;

open cursor\_1;

fetch cursor\_1 into c\_id, f\_name, l\_name;

if not done then

select c\_id, f\_name, l\_name;

else

select 'No customer with last name ending with Scott found.';

end if;

close cursor\_1;

end//

CALL first\_record();