**SQL Training**

Course-End Project Problem Statement



**Course-End Project: Air Cargo Analysis**

**Problem Statement Scenario:**

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 a 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*

1. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.

**SQL SCRIPT**

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

gender CHAR(1)

);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/customer.csv' INTO TABLE customers

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

-- create table for passenger flight

CREATE TABLE flights (

customer\_id INT,

aircraft\_id VARCHAR(10),

route\_id INT,

depart VARCHAR(3),

arrival VARCHAR(3),

seat\_num VARCHAR(4),

class\_id VARCHAR(10),

travel\_date DATE,

flight\_num INT

);

ALTER TABLE flights

MODIFY COLUMN class\_id varchar(20);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/passengers\_on\_flights.csv' INTO TABLE flights

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

-- create table for routes

CREATE TABLE routes (

route\_id INT,

flight\_num INT,

origin\_airport VARCHAR(3),

destination\_airport VARCHAR(3),

aircraft\_id VARCHAR(10),

distance\_miles INT

);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/routes.csv' INTO TABLE routes

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

-- import csv ticket details

CREATE TABLE tickets (

p\_date DATE,

customer\_id INT,

aircraft\_id VARCHAR(15),

class\_id VARCHAR(15),

no\_of\_tickets INT,

a\_code VARCHAR(3),

Price\_per\_ticket DECIMAL(10,2),

brand VARCHAR(20)

);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.3/Uploads/ticket\_details.csv' INTO TABLE tickets

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

/\*

2. Write a query to create a 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, -- ensures route\_id is unique

flight\_num VARCHAR(10) CHECK (flight\_num REGEXP '^[A-Z0-9]{2,10}$'), -- reg expression

origin\_airport VARCHAR(50) NOT NULL,

destination\_airport VARCHAR(50) NOT NULL,

aircraft\_id INT REFERENCES flights(aircraft\_id),

distance\_miles INT CHECK (distance\_miles > 0)

);

/\* flight number has a check constraint that ensures that the value matches a regular expression pattern.

The pattern ‘^[A-Z0-9]{2,10}$’ means that the value must start and end with an alphanumeric character (A-Z or 0-9)

and have a length between 2 and 10 characters. google regular expression for 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 \* FROM flights

WHERE route\_id BETWEEN 1 AND 25

ORDER by route\_id, customer\_id;

/\*

4. 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 passengers, SUM(Price\_per\_ticket \* no\_of\_tickets) AS revenue

FROM tickets

WHERE class\_id = 'Bussiness'

GROUP BY class\_id;

/\*

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 customers;

/\*

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 customers c

INNER JOIN tickets t ON c.customer\_id = t.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 \* from tickets;

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

FROM customers c

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

WHERE t.brand ='Emirates';

-- not working

/\*

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 customer\_id

FROM flights

WHERE class\_id = 'Economy Plus'

GROUP BY customer\_id

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 p\_date, customer\_id, class\_id, Price\_per\_ticket \* no\_of\_tickets AS revenue,

IF(Price\_per\_ticket \* no\_of\_tickets > 10000, 'YES', 'NO') AS crossed\_10000

FROM tickets;

/\* 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 'abc1234';

FLUSH PRIVILEGES;

GRANT ALL PRIVILEGES ON airlines.\* TO 'new\_user'@'localhost' ;

/\*

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

SELECT customer\_id, price\_per\_ticket, MAX(price\_per\_ticket) OVER (PARTITION BY class\_id) AS max\_price

FROM tickets

GROUP BY class\_id, price\_per\_ticket,customer\_id;

/\*

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 flights (route\_id);

SELECT \* FROM flights

WHERE route\_id = 4;

/\*

14. Total price per customer across aircraft IDs with rollup: \*/

SELECT customer\_id, SUM(price\_per\_ticket \* no\_of\_tickets) AS total\_price

FROM tickets

GROUP BY customer\_id WITH ROLLUP;

/\* WITH ROLLUP modifier to the GROUP BY clause. This will create an extra row at the

end of the result set, with a NULL value for the customer\_id and the

sum of all total\_price values for all customers. This is the grand total of all tickets. \*/

-- Task 15. View with business class customers and brands:

CREATE VIEW business\_class\_view AS

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

FROM customers c

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

WHERE t.class\_id = 'Bussiness';

-- Show view

SELECT \* FROM business\_class\_view;

/\*

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 get\_passengers\_by\_route\_range(

IN start\_route INT,

IN end\_route INT

)

BEGIN

DECLARE error\_msg VARCHAR(255);

IF NOT EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'flights') THEN

SET error\_msg = 'Table flights does not exist.';

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = error\_msg;

END IF;

SELECT \*

FROM flights

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

END &&

Call get\_passengers\_by\_route\_range(2,4); -- Call the procedure

/\*

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 get\_long\_distance\_routes()

BEGIN

SELECT \*

FROM routes

WHERE distance\_miles > 2000;

END &&

Call get\_long\_distance\_routes();

/\*

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 categorize\_flight\_distances()

BEGIN

DECLARE distance\_category VARCHAR(10);

SELECT

flight\_num,

CASE

WHEN distance\_miles >= 0 AND distance\_miles <= 2000 THEN 'SDT'

WHEN distance\_miles > 2000 AND distance\_miles <= 6500 THEN 'IDT'

ELSE 'LDT'

END AS distance\_category

FROM routes;

END &&

call categorize\_flight\_distances()

/\*

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 is\_complimentary\_service(class\_id varchar(15))

RETURNS VARCHAR(3)

BEGIN

DECLARE service VARCHAR(3);

CASE class\_id

WHEN 'Bussiness' THEN SET service = 'YES';

WHEN "First Class" THEN SET service = 'YES';

ELSE SET service = 'NO';

END CASE;

RETURN service;

END &&

DELIMITER &&

CREATE PROCEDURE get\_ticket\_details\_with\_services()

BEGIN

SELECT p\_date, customer\_id, class\_id, price\_per\_ticket \* no\_of\_tickets AS total\_price,

is\_complimentary\_service(class\_id) AS complimentary\_services

FROM tickets;

END &&

call get\_ticket\_details\_with\_services();

/\*

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 get\_first\_scott\_customer()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE c\_id INT;

DECLARE c\_name VARCHAR(255);

DECLARE cur CURSOR FOR SELECT customer\_id, CONCAT(first\_name, ' ', last\_name)

FROM customers

WHERE last\_name LIKE '%Scott';

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

FETCH cur INTO c\_id, c\_name;

WHILE NOT done DO

SELECT c\_id, c\_name;

FETCH cur INTO c\_id, c\_name;

END WHILE;

CLOSE cur;

IF c\_id IS NULL THEN

SELECT 'No customer found with last name ending in Scott.';

ELSE

SELECT c\_id, c\_name;

END IF;

END &&

CALL get\_first\_scott\_customer();

--Alternate method feeding parameters to the procedure

DELIMITER //

CREATE PROCEDURE get\_customers\_by\_last\_name(IN pattern VARCHAR(50))

BEGIN

DECLARE customer\_id INT;

DECLARE first\_name VARCHAR(50);

DECLARE last\_name VARCHAR(50);

DECLARE date\_of\_birth DATE;

DECLARE gender CHAR(1);

DECLARE done INT DEFAULT FALSE;

DECLARE customer\_cursor CURSOR FOR

SELECT customer\_id, first\_name, last\_name, date\_of\_birth, gender

FROM customers

WHERE last\_name LIKE pattern;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN customer\_cursor;

FETCH customer\_cursor INTO customer\_id, first\_name, last\_name, date\_of\_birth, gender;

customer\_loop: LOOP

select customer\_id, first\_name, last\_name, date\_of\_birth, gender;

FETCH customer\_cursor INTO customer\_id, first\_name, last\_name, date\_of\_birth, gender;

IF done THEN

LEAVE customer\_loop;

END IF;

-- Print customer details

SELECT CONCAT('Customer ID: ', customer\_id) AS customer\_id,

CONCAT('First Name: ', first\_name) AS first\_name,

CONCAT('Last Name: ', last\_name) AS last\_name,

CONCAT('Date of Birth: ', DATE\_FORMAT(date\_of\_birth, '%d-%m-%Y')) AS date\_of\_birth,

CONCAT('Gender: ', gender) AS gender;

END LOOP customer\_loop;

CLOSE customer\_cursor;

END //

DELIMITER ;

CALL get\_customers\_by\_last\_name('%Stewart');

SELECT \* FROM customers WHERE last\_name LIKE '%Stewart%';