**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

**Operations performed:**

1. **Create an ER diagram for the given airlines database.**
   1. From the menu bar click on Database-> Reverse Engineer. (SS10)
   2. Reverse Engineer Database dialogue box will appear. Set the parameters to connect to a database and then click on Next. (SS11)
   3. In the next dialogue box connection with the database is made and relevant information is fetched. Click Next. (SS12)
   4. Select the airlines schema and it will be included. Click on Next. (SS13)
   5. Click next after the objects from the selected schema are retrieved. (SS14)
   6. Select “Import the object” option from the dialogue box and click on Execute. (SS15)
   7. Reverse Engineering Progress dialogue box will appear. Click on next once the operations are completed successfully. (SS16)
   8. Click on Finish. (SS17)
   9. The tables are ready. (SS18)
   10. Right-click on the table and select the edit <table> option. (SS19)
   11. Then set the primary key and foreign key for each table and make the ER Diagram. (SS20- SS22)
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,

flight\_num int,

origin\_airport varchar(10),

destination\_airport varchar(10),

aircraft\_id varchar(10),

distance\_miles int,

check(flight\_num>1),

unique(route\_id),

check(distance\_miles>0)

);

desc route\_details;

1. **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

customer\_id, route\_id

FROM

passengers\_on\_flights

WHERE

route\_id BETWEEN 01 AND 25

ORDER BY customer\_id;

1. **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) AS 'Number of Passengers',

sum(no\_of\_tickets\*Price\_per\_ticket) AS 'Total Revenue'

FROM

ticket\_details

WHERE

class\_id = 'Bussiness';

1. **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(COALESCE(TRIM(first\_name), ''),

' ',

COALESCE(TRIM(last\_name), '')) AS 'Full Name'

FROM

customer;

1. **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.customer\_id),

CONCAT(COALESCE(TRIM(c.first\_name), ''),

' ',

COALESCE(TRIM(c.last\_name), '')) AS 'Full Name'

FROM

customer c

JOIN

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

WHERE

p\_date IS NOT NULL

AND no\_of\_tickets IS NOT NULL

ORDER BY c.customer\_id;

1. **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

customer c

JOIN

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

WHERE

t.brand='Emirates'

ORDER BY c.customer\_id;

1. **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, class\_id from passengers\_on\_flights group by customer\_id, class\_id having class\_id= 'Economy Plus';

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

SELECT

SUM(no\_of\_tickets \* Price\_per\_ticket) AS 'Revenue',

IF(SUM(no\_of\_tickets \* Price\_per\_ticket) > 10000,

'The revenue has crossed 10000',

'Sorry, The revenue has not crossed 10000') AS Message

FROM

ticket\_details;

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

CREATE USER 'Surbhi'@'localhost' IDENTIFIED BY 'Surbhi@8';

show privileges;

Grant alter,create,delete,drop,index,insert,select,update,trigger,alter routine,

create routine, execute, create temporary tables ON airlines.\* to 'Surbhi'@'localhost';

FLUSH PRIVILEGES;

Show Grants for 'Surbhi'@'localhost';

1. **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 Maximum\_Ticket\_Price

from ticket\_details t;

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

Select \* from passengers\_on\_flights where route\_id=4;

create index pof on passengers\_on\_flights(route\_id);

Select \* from passengers\_on\_flights where route\_id=4;

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

Select \* from passengers\_on\_flights where route\_id=4;

-- On executing the query from the side bar of the result grid select Execution Plan.

1. **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, sum(no\_of\_tickets\*Price\_per\_ticket) AS 'Total Price' from ticket\_details group by customer\_id with rollup;

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

Create view bcb As

Select customer\_id,brand, class\_id from ticket\_details where class\_id='Bussiness';

Select \* from bcb order by customer\_id;

1. **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 GetDetails(origin text, destination text)

Begin

DECLARE passengers\_table\_exists INT;

DECLARE customer\_table\_exists INT;

SELECT COUNT(\*) INTO passengers\_table\_exists FROM information\_schema.tables

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

SELECT COUNT(\*) INTO customer\_table\_exists FROM information\_schema.tables

WHERE table\_schema = DATABASE() AND table\_name = 'customer';

IF passengers\_table\_exists = 0 or customer\_table\_exists = 0 THEN

SELECT 'Error Message: passengers\_on\_flights table does not exist.' AS Message;

ElSEIF customer\_table\_exists = 0 THEN

SELECT 'Error Message: customer table does not exist.' AS Message;

ELSE

Select c.customer\_id,c.first\_name,c.last\_name,c.gender,p.aircraft\_id,p.seat\_num,

p.class\_id,p.travel\_date

from passengers\_on\_flights p join customer c on p.customer\_id=c.customer\_id

where depart=origin AND arrival=destination

order by customer\_id;

End If;

END$$

call GetDetails('CRW','COD');

1. **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 RouteDetails()

Begin

Select \* from routes where distance\_miles>2000;

END$$

call RouteDetails();

1. **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 distance()

Begin

with categorise as

(Select \*,

case

when distance\_miles<= 2000 then 'Short Distance Travel (SDT)'

when distance\_miles>= 6500 then 'Long Distance Travel (LDT)'

else 'Intermediate Distance Travel (IDT)'

END

As 'Category'

from routes)

Select Category, count(\*) As 'Number of routes' from categorise

group by Category order by count(\*);

END$$

call distance();

1. **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***

Create function service(class text) returns text deterministic

Begin

declare cs text;

if class='Bussiness' or class='Economy Plus' then Set cs= 'Yes';

else set cs='No';

end if;

Return(cs);

End$$

Delimiter $$;

Select p\_date AS 'Purchase Date',customer\_id AS 'Customer Id',class\_id AS Class ,service(class\_id) AS 'Complimentary Services' from ticket\_details order by customer\_id;

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.**

Delimiter $$

Create procedure cur\_extract()

Begin

Declare a text;

Declare b text;

Declare name\_cur cursor for

Select first\_name,last\_name from customer where last\_name='Scott';

open name\_cur;

repeat

fetch name\_cur into a,b;

until b=0

end repeat;

Select a as 'First Name', b as 'Last Name';

close name\_cur;

End$$

Delimiter $$;

call cur\_extract();