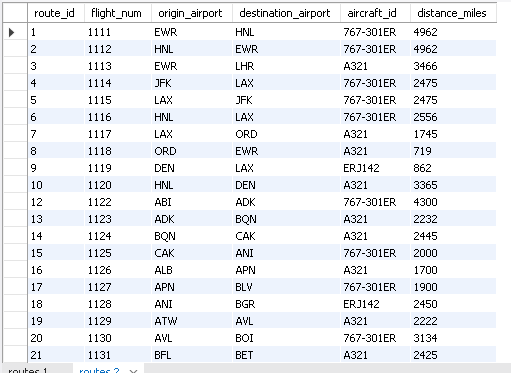
**Air Cargo Analysis**

#Q1. Create an ER diagram for the given airlines database.

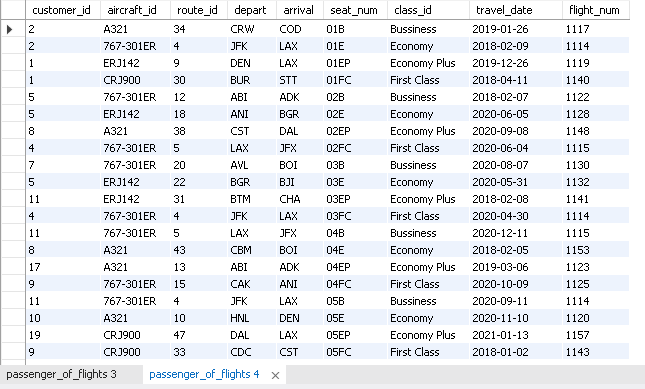
#Steps -- from Menu -> Database -> Reverse Engineer -> provide required password -> select database (airCargo) -> Click Next, Next -> Execute

#Q2. 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

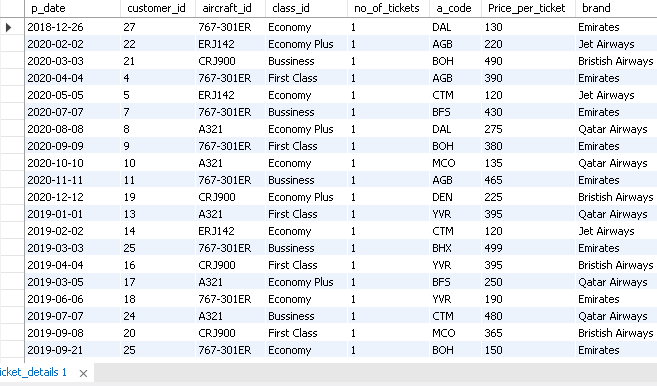
1. routes



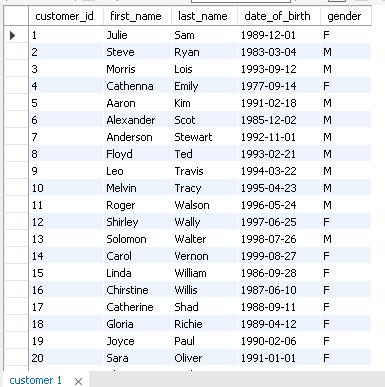
1. passenger\_of\_flights



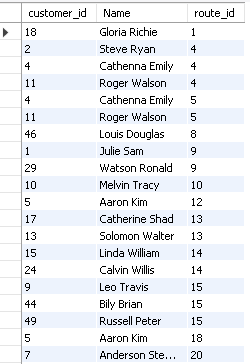
1. ticket\_details

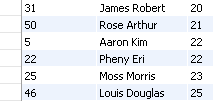


1. customers



Q3. 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.

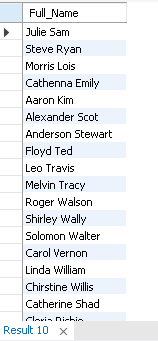




#Q4. Write a query to identify the number of passengers and total revenue in business class from the ticket\_details table.



#Q5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.



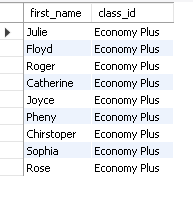
#Q6. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket\_details tables.



#Q7. 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.



#Q8. 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.



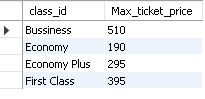
#Q9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket\_details table.



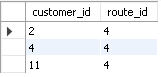
#Q10. Write a query to create and grant access to a new user to perform operations on a database.



#Q11. Write a query to find the maximum ticket price for each class using window functions on the ticket\_details table.

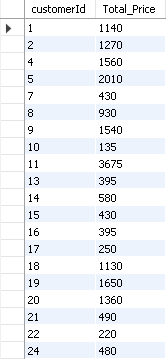


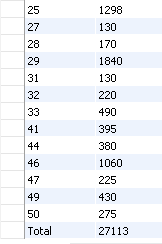
#Q12. Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers\_on\_flights table.



#Q13. For the route ID 4, write a query to view the execution plan of the passengers\_on\_flights table.

#Q14. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.

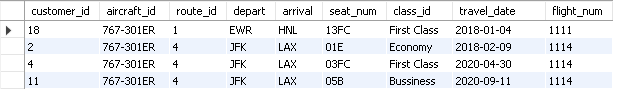




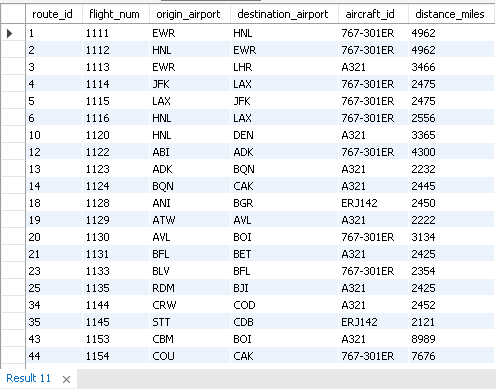
#Q15. Write a query to create a view with only business class customers along with the brand of airlines.



#Q16. 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.



#Q17. 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.



#Q18. 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\_travelled\_distance (flight\_no int, out info varchar(100))

Begin

declare dist int;

Select distance\_miles into dist from routes

where flight\_num = flight\_no;

if dist >= 0 and dist <= 2000 then set info = 'short distance travel';

elseif dist > 2000 and dist <= 6500 then set info = 'intermediate distance travel';

elseif dist > 6500 then set info = 'long-distance travel';

end if;

End //

Delimiter ;

call flight\_travelled\_distance(1111, @information);

Select @information as status;



#Q19. 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 varchar(100)) Returns varchar(3) deterministic

Begin

Declare comp\_Services varchar(3);

if class = 'Bussiness' or class = 'Economy Plus' then set comp\_Services = 'Yes';

else set comp\_Services = 'No';

End if;

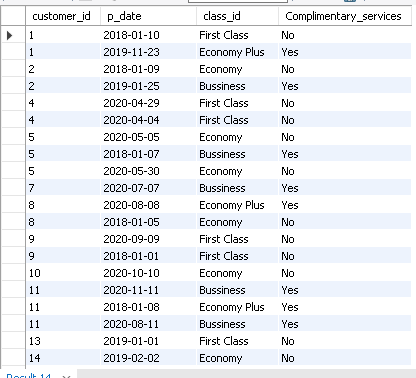
Return comp\_Services;

End //

Delimiter ;

Select customer\_id, p\_date, class\_id, complimentary\_services(class\_id) as Complimentary\_services

from ticket\_details;



#Q20. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.

Drop table if exists Scott\_customers;

Create temporary table Scott\_customers (

customerid int not null, firstname varchar(100), lastname varchar(100), dateofbirth date, gender varchar(1));

Delimiter //

Create procedure Get\_customer ()

Begin

declare done int default 0;

declare customerid int;

declare firstname, lastname varchar(100);

declare dateofbirth date;

declare gender varchar(1);

declare get\_cust Cursor for Select distinct \* from customer where last\_name like '%Scott';

declare continue handler for not found set done = 1;

open get\_cust;

label: loop

Fetch get\_cust into customerid, firstname, lastname, dateofbirth, gender;

Insert into Scott\_customers values(customerid, firstname, lastname, dateofbirth, gender);

If done = 1 then leave label;

end if;

end loop;

close get\_cust;

End //

Delimiter ;

call Get\_customer;

Select \* from Scott\_customers;

