**SQL PROJECT-2**

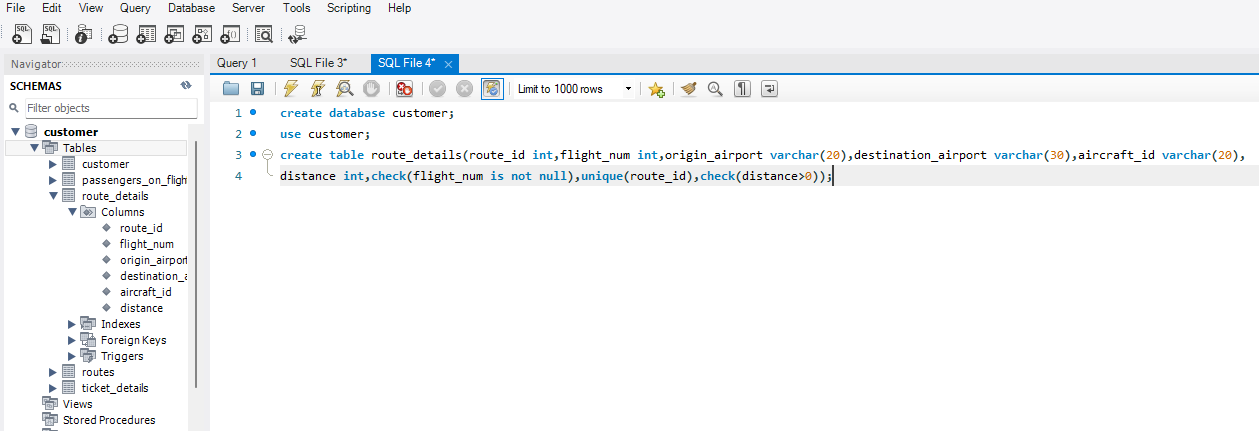
1. Write a query to create a route\_details table using suitable data types for the fields, such as route\_id, flight\_num, origin\_airport, destination\_airport, aircraft\_id, and distance\_miles. Implement the check constraint for the flight number and unique constraint for the route\_id fields. Also, make sure that the distance miles field is greater than 0.

**CODE:** create database customer;

use customer;

create table route\_details(route\_id int,flight\_num int,origin\_airport varchar(20),destination\_airportvarchar(30),aircraft\_idvarchar(20),distance int,check(flight\_num is not null),unique(route\_id),check(distance>0));

**OUTPUT:**

****

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.

**CODE:** select \* from passengers\_on\_flights where route\_id between 1 and 25 ;

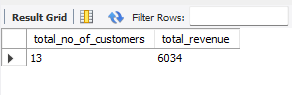
**OUTPUT:**

****

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

**CODE:** select count(customer\_id) as total\_no\_of\_customers ,sum(price\_per\_ticket) as total\_revenue from ticket\_details where class\_id="bussiness";

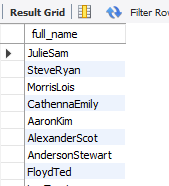
**OUTPUT:**

****

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

**CODE:** select concat(first\_name , last\_name) as full\_name from customer;

**OUTPUT:**

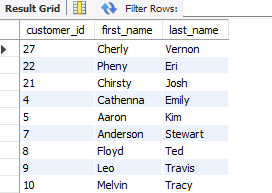
****

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

**CODE**: select td.customer\_id ,c.first\_name,c.last\_name from ticket\_details as td

left join customer as c on td.customer\_id=c.customer\_id;

**OUTPUT:**

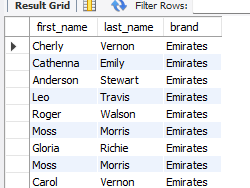


1. Write a query to identify the customer’s first name and last name based on their customer ID and brand (Emirates) from the ticket\_details table.

**CODE:** select c.first\_name,c.last\_name,td.brand from ticket\_details as td

left join customer as c on td.customer\_id=c.customer\_id where brand="emirates";

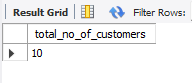
**OUTPUT:**

****

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.

**CODE:** select count(customer\_id) as total\_no\_of\_customers from passengers\_on\_flights group by class\_id having class\_id="economy plus";

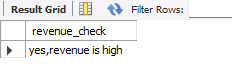
**OUTPUT:**

****

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

**CODE**: select if(sum(price\_per\_ticket)>10000 ,"yes,revenue is high","no") as revenue\_check from ticket\_details;

**OUTPUT:**

****

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

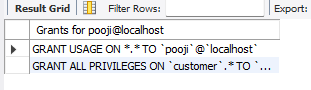
**CODE:** select user,host from mysql.user;

create user pooji@localhost;

show grants for pooji@localhost;

grant all on customer.\* to pooji@localhost;

**OUTPUT:**

****

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

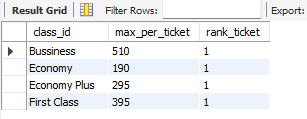
**CODE:** with poojitha as(

select class\_id,price\_per\_ticket as max\_per\_ticket,rank()over(partition by class\_id order by price\_per\_ticket desc) as rank\_ticket

from ticket\_details )

select distinct \* from poojitha where rank\_ticket =1;

**OUTPUT:**

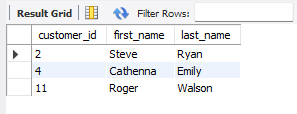
****

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.

**CODE**: select pf.customer\_id,c.first\_name,c.last\_name from passengers\_on\_flights as pf

left join customer c on pf.customer\_id=c.customer\_id where pf.route\_id=4;

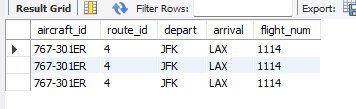
**OUTPUT:**

****

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

**CODE:** select aircraft\_id,route\_id,depart,arrival,flight\_num from passengers\_on\_flights where route\_id=4;

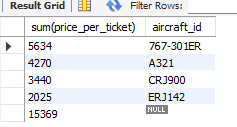
**OUTPUT:**

****

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

**CODE:** select sum(price\_per\_ticket),aircraft\_id from ticket\_details group by aircraft\_id with rollup;

**OUTPUT:**

****

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

**CODE:** create view busi\_class as

select customer\_id , brand from ticket\_details where class\_id="bussiness" ;

select \* from busi\_class;

**OUTPUT:**

****

**15**.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.

**CODE:** create procedure GetPassengersByRouteRange(IN start\_route INT, IN end\_route INT)

begin

if exists (select 1 from passengers\_on\_flights) then

select \* FROM passengers\_on\_flights

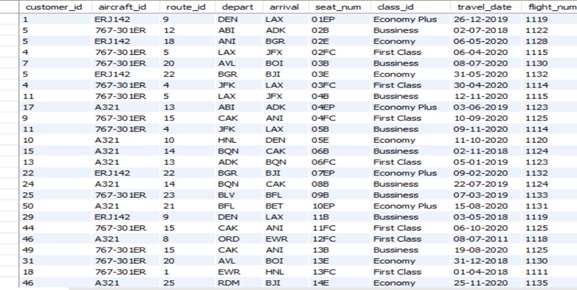
where route\_id between start\_route AND end\_route;

else signal sqlstate '45000' set message\_text = 'The table does not exist';

end if;

end

**OUTPUT:**



**16.** 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

**CODE:** Delimiter %%

create procedure distance()

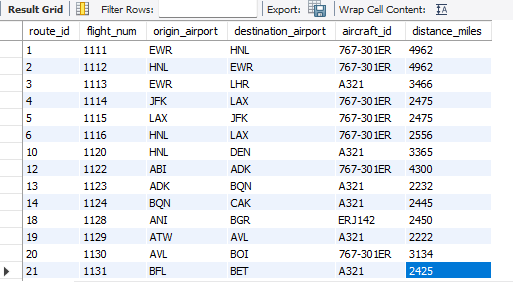
begin

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

end%%

call distance;

**OUTPUT:**

****

**17.** 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.

**CODE**: create procedure GroupDistanceTravel()

begin

select

flight\_num,

case

when distance\_miles between 0 and 2000 then 'sdt'

when distance\_miles > 2000 and distance\_miles <= 6500 then 'idt'

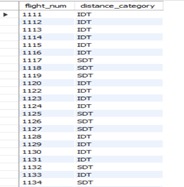
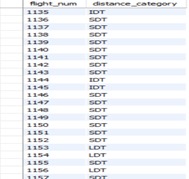
else 'ldt'

end as distance\_category

from routes;

end

**OUTPUT:**

**18.** 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

**CODE:**

delimiter //

create function get\_complimentary\_services(class\_id varchar(20))

returns varchar(3)

deterministic

[1:52 pm, 8/27/2024] kavya besant: 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 ;

code2:

create procedure display\_ticket\_details\_with\_services()

begin

select

p\_date,

customer\_id,

class\_id,

get\_complimentary\_services(class\_id) as complimentary\_services

from ticket\_details;

**OUTPUT:**

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

**CODE:** create procedure getfirstscottcustomer()

begin

declare done int default 0;

declare c\_id int;

declare c\_first\_name varchar(50);

declare c\_last\_name varchar(50);

declare cur cursor for

select customer\_id, first\_name, last\_name

from customer

where last\_name like '%scott';

declare continue handler for not found set done = 1;

open cur;

fetch cur into c\_id, c\_first\_name, c\_last\_name;

if not done then

select c\_id as customer\_id, c\_first\_name as first\_name, c\_last\_name as last\_name;

else

select 'no customer with last name ending in scott' as message;

end if;

close cur;

end

**OUTPUT:**

