create table employee.route\_details (

route\_id int , flight\_num int , origin\_airport varchar(100),

destination\_airport varchar(100), aircraft\_id int, distance\_miles int ,

unique(route\_id) , check (distance\_miles > 0), check (flight\_num > 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.\*/

select customer\_id, route\_id from employee.passengers\_on\_flights

where route\_id between 1 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(class\_id) as Bussiness\_class , sum(price\_per\_ticket) as total\_bussiness\_class\_revenue

from employee.ticket\_details

where class\_id = 'Bussiness';

/\*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 employee.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 DISTINCT(C.customer\_id) FROM employee.ticket\_details T

LEFT JOIN employee.customer C ON (C.customer\_id = T.customer\_id) WHERE T.customer\_id IS NOT NULL;

/\*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 first\_name , last\_name from employee.customer C

left join employee.ticket\_details T on (T.customer\_id = C.customer\_id) WHERE T.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 distinct(customer\_id) from employee.passengers\_on\_flights

group by customer\_id

having max(class\_id) = 'Economy Plus';

/\*9. Write a query to identify whether the revenue has crossed 10000

using the IF clause on the ticket\_details table.\*/

select if(revenue > 10000 , 'yes' ,'no') as revenue\_crossed\_10000

from (select sum(price\_per\_ticket ) as revenue from employee.ticket\_details)

as reverue\_summary;

/\*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 your\_database.\* 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. \*/

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

FROM employee.ticket\_details;

/\*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 indx1 on employee.passengers\_on\_flights(route\_id);

select customer\_id,route\_id 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\*/

SELECT \* FROM employee.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 customer\_id,aircraft\_id,SUM(Price\_per\_ticket)AS Total\_sales

FROM employee.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. \*/

create view Business\_class as

select customer\_id , class\_id , brand FROM employee.ticket\_details

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 &&

create procedure get\_pasenger\_by\_route\_range(

IN start\_route int,

IN end\_route int)

Begin declare error\_msg varchar(260);

select\* from employee.passengers\_on\_flights where route\_id between start\_route and end\_route;

if not exists (select 1 from employee.routes where table\_name = 'passengers\_on\_flights') then

set error\_msg ='table flights does not exist';

signal sqlstate '45000' set message\_text = error\_msg;

end if;

end &&

/\*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 routes2000()

begin

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

end $$

delimiter ;

/\*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 group\_distance()

BEGIN

SELECT flight\_number, distance\_miles, CASE

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

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

WHEN distance\_miles > 6500 THEN 'LDT'

END AS distance\_category

FROM employee.routes;

END $$

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 $$

create procedure complementary(out p\_date date, out customer\_ID int, out class\_ID varchar(20), out service varchar(10))

begin

select p\_date, customer\_ID, class\_ID , case

when class\_ID = 'Business' then 'YES'

when class\_ID = 'Economy Plus' then 'YES'

else 'No'

end as complementary

INTO p\_date, customer\_ID, class\_ID, service

from ticket\_details;

end $$

delimiter ;

/\*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 firstcursor()

BEGIN

DECLARE nlast VARCHAR(25);

DECLARE done INT DEFAULT FALSE;

DECLARE cursor\_1 CURSOR FOR

SELECT customer\_id, first\_name, last\_name FROM employee.customer

WHERE last\_name = 'Scott';

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cursor\_1;

read\_loop: LOOP

FETCH cursor\_1 INTO nlast;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT nlast AS last\_name;

END LOOP;

CLOSE cursor\_1;

END $$

DELIMITER ;