```
-- QUERY 1- Create an ER diagram for the given airlines database.

CREATE DATABASE Airlines;

USE Airlines;

select * from customer;                    # CUSTOMER TABLE created.

select * from passengers_on_flights;          # PASSENGERS TABLE CREATED.

select * from ticket_details;               # TICKET DETAILS TABLE CREATED.

select * from routes;                   # ROUTE TABLE CREATED.

show tables;

describe passengers_on_flights;

-- QUERY 2

-- DONE ALREADY--

# QUERY 3

## Write a query to display all the passengers (customers) who have travelled in routes 01 to 25.

select customer_id,route_id from passengers_on_flights

where route_id between 1 and 25

order by route_id asc;

-- QUERY 4--Write a query to identify the number of passengers and total revenue in business class

-- from the ticket_details table

SELECT COUNT(no_of_tickets) AS total_passengers, SUM((no_of_tickets)*(Price_per_ticket)) AS
business_class_revenue

FROM ticket_details

WHERE class_id ='Bussiness';

-- QUERY 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 first_name,last_name, concat(first_name,' ',last_name) as FULL_NAME
```

```
FROM customer;

-- QUERY 6- Write a query to extract the customers who have registered and booked a ticket.

SELECT  distinct(customer_id),first_name from customer

inner join ticket_details using(customer_id);

-- QUERY 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,brand from customer

inner join ticket_details using(customer_id)

where brand='Emirates';

-- QUERY 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,class_id,route_id

from passengers_on_flights

group by customer_id,route_id

having class_id='Economy Plus';

-- QUERY 9-Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket_details table.

SELECT IF(SUM((no_of_tickets)*(Price_per_ticket)) > 1000,'YES','NO')

FROM ticket_details;

-- QUERY 10-Write a query to create and grant access to a new user to perform operations on a database.

CREATE USER 'NEWUSER'@'LOCALHOST'

identified BY '123456';

GRANT execute ON airlines.*

TO NEWUSER@LOCALHOST;
```

```sql
-- QUERY-11-Write a query to find the maximum ticket price for each class using window functions on the ticket_details table.

select customer_id,class_id,Price_per_ticket,brand,

MAX(Price_per_ticket) over(partition by brand) as high_price

from ticket_details;

-- QUERY 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.

select * from passengers_on_flights

where route_id='4';

 CREATE INDEX route_id_4 ON passengers_on_flights(route_id);

SELECT*FROM passengers_on_flights

WHERE route_id = '4';

-- QUERY 13-For the route ID 4, write a query to view the execution plan of the passengers_on_flights table.

explain select * from passengers_on_flights where route_id='4';

-- QUERY 14-Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs

-- using rollup function

SELECT distinct(aircraft_id),SUM(no_of_tickets*Price_per_ticket) as total_revnue

FROM ticket_details

group by aircraft_id  with rollup;

-- QUERY 15-Write a query to create a view with only business class customers along with the brand of airlines.

create view `business_class_customers` as select class_id,brand,customer_id

from ticket_details

where class_id='bussiness';
```

```
-- QUERY 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 routesdetails (IN range_route INT,OUT routes_details varchar(100))

BEGIN

DECLARE range_route INT;

SELECT *

FROM passengers_on_flights WHERE route_id=range_route;

case routes_details

WHEN range_route BETWEEN 1 AND 10 THEN SET routes_details = 'route1to10';

WHEN range_route BETWEEN 10 AND 20 THEN SET routes_details = 'route10to20';

WHEN range_route BETWEEN 20 AND 30 THEN SET routes_details = 'route20to30';

WHEN range_route BETWEEN 30 AND 40 THEN SET routes_details = 'route30to40';

WHEN range_route BETWEEN 40 AND 50 THEN SET routes_details = 'route40to50';

END CASE;

END $$

delimiter ;

call routesdetails('9',@routes_details);

-- QUERY 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 travel_details(in route_dist int)

return varchar(100)

begin
```

```
select * from routes

where distance_miles>2000;

end;

delimiter ;

-- QUERY 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 category_dist(in covered_dist int,out category_dist varchar(50))

begin

select * from routes where distance_miles=covered_dist;

IF (covered_dist>=0 and covered_dist<=2000) then

 set category_dist='SDT';

ELSEIF (covered_dist>2000 and covered_dist<=6500) then

 set category_dist='IDT';

ELSEIF (covered_dist>=6500) then

set category_dist='LDT';

END IF;

end $$

delimiter ;

-- QUERY 19-

DELIMITER $$

CREATE FUNCTION complimentary_services (class_id VARCHAR (100))

RETURNS VARCHAR(100)

deterministic
```

```
BEGIN

DECLARE complimentary_services VARCHAR(100);

IF class_id = 'Business' THEN SET complimentary_services = 'Yes';

ELSEIF class_id = 'Economy Plus' THEN SET complimentary_services = 'Yes';

ELSEIF class_id = 'Economy' THEN SET complimentary_services = 'No';

ELSEIF class_id = 'First Class' THEN SET complimentary_services = 'No';

END IF;

RETURN (complimentary_services);

END $$;

DELIMITER ;

SELECT p_date, customer_id, class_id, complimentary_services(class_id) AS complimentary_services

FROM ticket_details;


-- QUERY 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 my_cursor ()

BEGIN

DECLARE a VARCHAR (100);

DECLARE b VARCHAR (100);

DECLARE my_cursor CURSOR FOR SELECT last_name, first_name FROM customer

WHERE last_name = 'Scott';

OPEN my_cursor;

REPEAT FETCH my_cursor INTO a,b;

UNTIL b = 0 END REPEAT;
```

```
SELECT a AS last_name, b AS first_name;

CLOSE my_cursor;

END;

END $$;

DELIMITER ;

call my_cursor();
```