```
CREATE DATABASE Air_Cargo_Analysis;

USE Air_Cargo_Analysis;
SET sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));

CREATE TABLE route_details (route_id int NOT NULL,
flight_num int NOT NULL,
origin_airport varchar(20),
destination_airport varchar(20),
aircraft_id varchar(10),
distance int NOT NULL,
UNIQUE(route_id), CHECK (distance>0));

SELECT*FROM passengers_on_flights
WHERE route_id BETWEEN 1 AND 25;

SELECT COUNT(class_id = 'Business') AS BUSINESS_CLASS_COUNT,
SUM(no_of_tickets*price_per_ticket) AS REVENUE_TOTAL  FROM ticket_details
WHERE class_id = 'Bussiness';

SELECT CONCAT(first_name, " ", last_name) AS FULL_NAME FROM customer;

SELECT customer_id, CONCAT(first_name, " " , last_name) AS NAME, COUNT(no_of_tickets) AS
NO_OF_TICKETS
FROM customer
JOIN ticket_details USING (customer_id)
GROUP BY customer_id, NAME
ORDER BY NO_OF_TICKETS;

SELECT customer_id, first_name, last_name FROM customer
JOIN ticket_details USING(customer_id)
WHERE brand = 'Emirates';
```

```
SELECT customer.customer_id, customer.first_name, customer.last_name,
passengers_on_flights.class_id

FROM customer

JOIN passengers_on_flights ON customer.customer_id = passengers_on_flights.customer_id

WHERE passengers_on_flights.class_id = 'Economy Plus';


SELECT customer.customer_id, customer.first_name, customer.last_name,
passengers_on_flights.class_id

FROM customer

JOIN passengers_on_flights ON customer.customer_id = passengers_on_flights.customer_id

GROUP BY customer_id

HAVING class_id = 'Economy Plus'

ORDER BY customer_id;


SELECT IF(SUM(no_of_tickets * Price_per_ticket)>10000,'Revenue Crossed 10000','Revenue Less
Than 10000')

AS REVENUE_STATUS FROM ticket_details;


CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'new_password';

GRANT ALL ON Air_Cargo_Analysis.* TO 'new_user'@'localhost';


SELECT customer_id,class_id,brand , MAX(Price_per_ticket) OVER (PARTITION BY class_id) AS
max_price FROM ticket_details;


SELECT * FROM passengers_on_flights WHERE route_id = 4 ;

SELECT * FROM passengers_on_flights HAVING route_id = 4 ;


SELECT customer_id, aircraft_id , class_id , sum(no_of_tickets * Price_per_ticket)

AS total_price

FROM ticket_details

GROUP BY aircraft_id WITH ROLLUP ;
```

```
DROP VIEW IF EXISTS business_class ;
CREATE VIEW business_class AS
SELECT customer_id, class_id, brand
FROM ticket_details
WHERE class_id = 'bussiness' ;
SELECT * FROM business_class ;

DROP PROCEDURE IF EXISTS passenger_details;
DELIMITER &&
CREATE PROCEDURE passenger_details(route int)
BEGIN
SELECT * FROM route_details
WHERE route_id=route;
END &&;
CALL passenger_details(2);

DROP PROCEDURE IF EXISTS travelled_distance()
DELIMITER //
CREATE PROCEDURE travelled_distance()
BEGIN
SELECT * FROM routes WHERE distance_miles>2000;
END //
CALL travelled_distance();

DROP PROCEDURE IF EXISTS group_of_distance
DELIMITER //
CREATE PROCEDURE group_of_distance (IN distance int,OUT CATEGORY VARCHAR(40))
BEGIN
DECLARE DIST INT DEFAULT 1;
SELECT distance_miles INTO DIST FROM routes WHERE distance_miles=distance;
```

```
IF DIST BETWEEN 0 AND 2000 THEN
SET CATEGORY='SHORT DISTANCE TRAVELLED';
ELSEIF DIST BETWEEN 2000 AND 6500 THEN
SET CATEGORY='INTERMEDIATE DISTANCE TRAVELLED';
ELSE SET CATEGORY='LONG DISTANCE TRAVELLED';
END IF ;
END //
CALL group_of_distance(1558, @CATEGORY);
SELECT @CATEGORY

DROP FUNCTION IF EXISTS Complementary_Services
DELIMITER //
CREATE FUNCTION Complementary_Services(class_id VARCHAR(40))
RETURNS VARCHAR(10) DETERMINISTIC
BEGIN
DECLARE SERVICE VARCHAR(20);
IF class_id = 'Economy Plus' OR 'Business' THEN
SET SERVICE  = 'YES';
ELSE SET SERVICE = 'NO';
END IF;
RETURN SERVICE;
END //
SELECT p_date,customer_id,class_id, Complementary_Services(class_id) AS SERVICE FROM
ticket_details;

DROP PROCEDURE IF EXISTS First_Scott
DELIMITER //
CREATE PROCEDURE First_Scott()
BEGIN
DECLARE a VARCHAR(50);
DECLARE b VARCHAR(50);
```

```
DECLARE cursor_1 CURSOR FOR SELECT first_name, last_name FROM customer
WHERE last_name = 'Scott';
OPEN cursor_1;
REPEAT FETCH cursor_1 INTO a,b;
UNTIL b=0
END REPEAT;
SELECT a AS first_name , b as last_name;
CLOSE cursor_1;
END //
CALL First_Scott()
```