Air Cargo Analysis:

```
#create database air_cargo;
use air_cargo;
select * from customer;
select * from passengers_on_flights;
select * from routes;
select * from ticket_details;
#2.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.
CREATE TABLE route_details (
   route_id INT PRIMARY KEY,
   flight_num VARCHAR(10) NOT NULL CHECK (flight_num REGEXP '^[A-Z0-9]+$'),
   origin_airport VARCHAR(50) NOT NULL,
   destination_airport VARCHAR(50) NOT NULL,
   aircraft_id VARCHAR(20) NOT NULL,
   distance_miles INT NOT NULL CHECK (distance_miles > 0),
   UNIQUE (route_id)
);
insert into route_details (route_id,flight_num,origin_airport,
destination_airport,aircraft_id,distance_miles)
select route_id,flight_num,origin_airport,destination_airport,aircraft_id,distance_miles
from routes;
#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 * from passengers_on_flights where route_id > 1 and route_id < 25;
#4.Write a query to identify the number of passengers and total revenue in business class from the
ticket_details table.
select * from ticket_details;
SELECT
   COUNT(*) AS number_of_passengers,
   SUM(Price_per_ticket) AS total_revenue
```

```
FROM
   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
   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
   customer.customer_id,
   customer.first_name,
   customer.last_name
FROM
   customer
INNER JOIN
   ticket_details ON customer.customer_id = ticket_details.customer_id;
```

#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
          customer.customer_id,
   customer.first_name,
   customer.last_name
FROM
   customer
INNER JOIN
   ticket_details ON customer.customer_id = ticket_details.customer_id
WHERE
   ticket_details.brand = 'Emirates';
```

```
#AND

#   customer.customer_id = 4; we can select the specific customer id too.

#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.customer_id,

   customer.first_name,

   customer.last_name

FROM

   customer

INNER JOIN

   passengers_on_flights ON customer.customer_id = passengers_on_flights.customer_id

WHERE

   passengers_on_flights.class_id = 'Economy plus'

GROUP BY

   customer.customer_id,

   customer.first_name,

   customer.last_name

HAVING

          COUNT(passengers_on_flights.route_id) > 0;

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

SELECT

   IF(SUM(Price_per_ticket) > 10000, 'Revenue has crossed 10000', 'Revenue has not crossed 10000') AS revenue_status

FROM

   ticket_details;

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

SELECT

   aircraft_id,

   class_id,
   Price_per_ticket,
   MAX(Price_per_ticket) OVER(PARTITION BY class_id) AS max_ticket_price
FROM
   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

```
SELECT customer_id, aircraft_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.

```
explain select
    customer_id,
    aircraft_id,
    route_id
FROM
    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_price
FROM
    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_customers as
SELECT
   customer.customer_id,
   customer.first_name,
   customer.last_name,
   ticket_details.brand
FROM
   ticket_details
JOIN
   customer ON ticket_details.customer_id = customer.customer_id
WHERE
   ticket_details.class_id = 'Bussiness';


SELECT * FROM air_cargo.business_class_customers;
```

#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 GetLongDistanceRoutes()
BEGIN
   SELECT
      route_id,
      flight_num,
      origin_airport,
      destination_airport,
      aircraft_id,
      distance_miles
   FROM
      routes
   WHERE
       distance_miles > 2000;
END //

DELIMITER ;

call GetLongDistanceRoutes();
```

#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 CategorizeFlightDistances()
BEGIN
   SELECT
       flight_num,
       origin_airport,
       destination_airport,
       aircraft_id,
       distance_miles,
       CASE
          WHEN distance_miles >= 0 AND distance_miles <= 2000 THEN 'Short Distance Travel (SDT)'
          WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN 'Intermediate Distance Travel
(IDT)'
          WHEN distance_miles > 6500 THEN 'Long Distance Travel (LDT)'
          ELSE 'Unknown Category'
       END AS travel_category
   FROM
       routes;
END //

DELIMITER ;
```

```
call CategorizeFlightDistances();


/*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*/

 #creating a stored function that returns 'Yes' or 'No' based on the class

DELIMITER //

CREATE FUNCTION GetComplimentaryServices(class_id VARCHAR(255))

RETURNS VARCHAR(3)

DETERMINISTIC

BEGIN

   DECLARE result VARCHAR(3);


   IF class_id IN ('Business', 'Economy Plus') THEN

      SET result = 'Yes';

   ELSE

      SET result = 'No';

   END IF;


   RETURN result;

END //


DELIMITER ;

#creating a stored procedure that uses the function to extract the required details from the
ticket_details table.

DELIMITER //


CREATE PROCEDURE GetTicketDetails()

BEGIN

   SELECT
```

```
        p_date,
        customer_id,
        class_id,
        GetComplimentaryServices(class_id) AS complimentary_services
    FROM
        ticket_details;
END //

DELIMITER ;

call GetTicketDetails;
```