```sql
-- Air Cargo Analysis Project

-- DESCRIPTION:
/* Air Cargo is an aviation company that provides air transportation services for passengers and freight. Air Cargo uses its aircraft to provide
different services with the help of partnerships or alliances with other airlines. The company wants to prepare reports on regular passengers, busiest routes,
 ticket sales details, and other scenarios to improve the ease of travel and booking for customers.*/

-- Project Objective:
/* You, as a DBA expert, need to focus on identifying the regular customers to provide offers, analyze the busiest route which helps to increase the number of
 aircraft required and prepare an analysis to determine the ticket sales details. This will ensure that the company improves its operability and becomes more
 customer-centric and a favorable choice for air travel. */

-- Following operations should be performed:

-- Task 01: Create an ER diagram for the given airlines database.
CREATE DATABASE air_cargo;
USE air_cargo;
-- Right click on database - Select Tables - Select Table Data Import Wizard - Select path all datasets.
SELECT*FROM air_cargo.customer;
SELECT*FROM air_cargo.passengers_on_flights;
SELECT*FROM air_cargo.routes;
SELECT*FROM air_cargo.ticket_details;

SET FOREIGN_KEY_CHECKS=0;
SET GLOBAL FOREIGN_KEY_CHECKS=0;

ALTER TABLE air_cargo.passengers_on_flights ADD FOREIGN KEY (customer_id) REFERENCES air_cargo.customer (customer_id);
DESCRIBE air_cargo.customer;
DESCRIBE air_cargo.passengers_on_flights;
ALTER TABLE air_cargo.passengers_on_flights ADD FOREIGN KEY (route_id) REFERENCES air_cargo.routes(route_id);
DESCRIBE air_cargo.routes;
ALTER TABLE air_cargo.ticket_details ADD FOREIGN KEY (customer_id) REFERENCES air_cargo.passengers_on_flights (customer_id);
ALTER TABLE air_cargo.ticket_details ADD FOREIGN KEY (customer_id) REFERENCES air_cargo.customer (customer_id);
DESCRIBE air_cargo.ticket_details;
-- ER diagram Select Database tab - Reverse Engineer Select Right click - Choose Default options and Create EER Diagram as below screenshort.

/* Task 02: Write a query to create 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 air_cargo.route_details
(
route_id INT UNIQUE,
flight_num INT check(flight_num >0),
origin_airport VARCHAR(50),
destination_airport VARCHAR(50),
aircraft_id INT,
distance_miles INT CHECK (distance_miles >0)
)
 ENGINE=INNODB;
DESCRIBE air_cargo.route_details;

-- Task 03: 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  BETWEEN 1 AND 25 ORDER BY route_id DESC
;

```

```sql
-- Task 04: Write a query to identify the number of passengers and total revenue in business class from the ticket_details table.
SELECT COUNT(customer_id) AS number_of_passengers, SUM(Price_per_ticket) AS total_revenue_in_business FROM air_cargo.ticket_details WHERE class_id = 'Bussiness';

-- Task 05: 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;

-- Task 06: Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket_details tables.
SELECT c.customer_id, t.no_of_tickets, t.class_id
FROM air_cargo.customer c JOIN ticket_details t
ON c.customer_id = t.customer_id
WHERE no_of_tickets > 0;

-- Task 07: 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 c.first_name, last_name, t.customer_id, t.brand
FROM customer c JOIN ticket_details t
ON c.customer_id = t.customer_id
WHERE brand = 'Emirates';

-- Task 08: 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 c.first_name, c.last_name, p.class_id
FROM customer c JOIN passengers_on_flights p
ON c.customer_id = p.customer_id
GROUP BY c.first_name,  p.class_id
HAVING p.class_id = "Economy Plus";

-- Task 09: Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket_details table.
SELECT SUM(Price_per_ticket) AS total_revenue,
IF(SUM(Price_per_ticket) > 10000, "Yes - revenue has crossed 10000", "NO - revenue has not crossed 10000") AS revenue_crossed_status FROM ticket_details;

-- Task 10: Write a query to create and grant access to a new user to perform operations on a database.
GRANT
ALL ON *.* TO 'root'@'localhost';
-- Grant access to air_cargo dataset
GRANT
ALL ON air_cargo TO 'root'@'localhost';

-- Task 11: Write a query to find the maximum ticket price for each class using window functions on the ticket_details table
SELECT class_id AS class, MAX(Price_per_ticket) AS maximum_ticket_price FROM air_cargo.ticket_details GROUP BY class_id ORDER BY class_id;

-- Task 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, route_id FROM air_cargo.passengers_on_flights WHERE route_id = 4;

-- Task 13: For the route ID 4, write a query to view the execution plan of the passengers_on_flights table.
SELECT * FROM air_cargo.passengers_on_flights WHERE route_id = 4;

-- Task 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_of_all_tickets FROM
 air_cargo.ticket_details  GROUP BY customer_id with rollup;

-- Task 15: Write a query to create a view with only business class customers along with the brand of airlines.
CREATE VIEW Bussiness_Class AS SELECT customer_id, brand FROM ticket_details WHERE class_id = 'Bussiness';
SELECT*FROM Bussiness_Class;

```

```sql
/* Task 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_total_passengers_()
BEGIN
DECLARE total_passengers INT DEFAULT 0;
SELECT COUNT(*) INTO total_passengers FROM air_cargo.passengers_on_flights;
SELECT total_passengers;
END &&
DELIMITER ;
SHOW PROCEDURE STATUS;

-- Task 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 $$
CREATE PROCEDURE distance_miles()
BEGIN
SELECT*FROM routes WHERE distance_miles > 2000;
END $$
CALL distance_miles();

/* Task 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 $$
CREATE FUNCTION groups_distance(dist int)
RETURNS VARCHAR(10) DETERMINISTIC
BEGIN
DECLARE distance_categories CHAR(3);
IF dist BETWEEN 0 AND 2000 THEN SET distance_categories = 'SDT';
ELSEIF dist BETWEEN 2001 AND 6500 THEN SET distance_categories = 'IDT';
ELSEIF dist > 6500 THEN SET distance_categories = 'LDT';
END IF;
RETURN (distance_categories);
END $$
DELIMITER $$;

/* Task 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 */
SELECT p_date, customer_id, class_id,
CASE
WHEN class_id = 'Business'
OR  class_id = 'Economy Plus' THEN 'YES' ELSE 'NO'
END AS Complimentary_Service
FROM ticket_details ORDER BY customer_id;

-- Task 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.
SELECT*FROM customer WHERE last_name = 'Scott';
```