```
CREATE TABLE route_details (
    route_id INT PRIMARY KEY,
    flight_num VARCHAR(10) CHECK (flight_num LIKE 'F%[0-9][0-9][0-9]'),
    origin_airport VARCHAR(50),
    destination_airport VARCHAR(50),
    aircraft_id INT,
    distance_miles DECIMAL(10, 2) CHECK (distance_miles > 0),
    UNIQUE (route_id)
);



SELECT DISTINCT p.customer_id, c.first_name, c.last_name
FROM passengers_on_flights p
JOIN Customer c ON p.customer_id = c.customer_id
WHERE p.route_id BETWEEN 1 AND 25;



SELECT
    SUM(no_of_tickets) AS total_passengers,
    SUM(price_per_ticket * no_of_tickets) AS total_revenue
FROM ticket_details
WHERE class_id = 'Business';



SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM Customer;



SELECT c.customer_id, c.first_name, c.last_name
FROM Customer c
INNER JOIN ticket_details t ON c.customer_id = t.customer_id;
```

```
SELECT c.first_name, c.last_name
FROM Customer c
INNER JOIN ticket_details t ON c.customer_id = t.customer_id
WHERE t.brand = 'Emirates';



SELECT c.customer_id, c.first_name, c.last_name
FROM passengers_on_flights p
JOIN Customer c ON p.customer_id = c.customer_id
WHERE p.class_id = 'Economy Plus'
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(*) > 0;



SELECT
  CASE
    WHEN SUM(price_per_ticket * no_of_tickets) > 10000 THEN 'Revenue Crossed 10000'
    ELSE 'Revenue Not Crossed 10000'
  END AS revenue_status
FROM ticket_details;



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

GRANT ALL PRIVILEGES ON `your_database`.* TO 'new_user'@'localhost';

FLUSH PRIVILEGES;
```

```
SELECT
   class_id,
   MAX(price_per_ticket) OVER (PARTITION BY class_id) AS max_ticket_price
FROM ticket_details;



CREATE INDEX idx_route_id ON passengers_on_flights(route_id);

SELECT *
FROM passengers_on_flights
WHERE route_id = 4;

SELECT customer_id, depart, arrival
FROM passengers_on_flights
WHERE route_id = 4
LIMIT 1000;



EXPLAIN SELECT *
FROM passengers_on_flights
WHERE route_id = 4;

EXPLAIN SELECT *
FROM passengers_on_flights
WHERE route_id = 4;

EXPLAIN
SELECT *
FROM passengers_on_flights
WHERE route_id = 4;
```

```
-- Subquery 1: Group by customer_id
SELECT
    customer_id,
    NULL AS aircraft_id, -- NULL to distinguish from aircraft totals
    SUM(price_per_ticket * no_of_tickets) AS total_price
FROM ticket_details
GROUP BY customer_id

UNION ALL

-- Subquery 2: Group by aircraft_id
SELECT
    NULL AS customer_id, -- NULL to distinguish from customer totals
    aircraft_id,
    SUM(price_per_ticket * no_of_tickets) AS total_price
FROM ticket_details
GROUP BY aircraft_id

ORDER BY customer_id, aircraft_id;



CREATE VIEW business_class_customers AS
SELECT c.customer_id, c.first_name, c.last_name, t.brand AS airline_brand
FROM Customer c
INNER JOIN ticket_details t ON c.customer_id = t.customer_id
WHERE t.class_id = 'Business';



DELIMITER //
CREATE PROCEDURE GetPassengerDetailsByRouteRange(
```

```
    IN route_start INT,
    IN route_end INT
)
BEGIN
    DECLARE table_exists INT;

    -- Check if the table exists
    SELECT COUNT(*) INTO table_exists
    FROM information_schema.tables
    WHERE table_name = 'passengers_on_flights';

    -- If the table doesn't exist, return an error message
    IF table_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Table passengers_on_flights does not exist';
    ELSE
        -- If the table exists, execute the dynamic SQL query
        SET @sql = CONCAT(
            'SELECT * FROM passengers_on_flights WHERE route_id BETWEEN ',
            route_start, ' AND ', route_end
        );

        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        DEALLOCATE PREPARE stmt;
    END IF;
END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE PROCEDURE GetRoutesByDistance()
BEGIN
   SELECT *
   FROM routes
   WHERE Distance_miles > 2000;
END;
//
DELIMITER ;

CALL GetRoutesByDistance();



DELIMITER //
CREATE PROCEDURE GroupDistanceCategories()
BEGIN
   SELECT
      Flight_num,
      Origin_airport,
      Destination_airport,
      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)'
      END AS Distance_Category
   FROM routes;
END;
//
DELIMITER ;
```

```
CALL GroupDistanceCategories();



DELIMITER //
CREATE FUNCTION DetermineComplimentaryServices(class_id VARCHAR(255)) RETURNS
VARCHAR(3)
BEGIN
   DECLARE is_complimentary VARCHAR(3);

   IF class_id IN ('Business', 'Economy Plus') THEN
      SET is_complimentary = 'Yes';
   ELSE
      SET is_complimentary = 'No';
   END IF;

   RETURN is_complimentary;
END;
//
DELIMITER ;

DELIMITER //
CREATE PROCEDURE GetTicketDetailsWithComplimentaryServices()
BEGIN
   SELECT
      p_date AS Ticket_Purchase_Date,
      customer_id AS Customer_ID,
      class_id AS Class_ID,
      DetermineComplimentaryServices(class_id) AS Complimentary_Services
   FROM ticket_details;
END;
```

```
//
DELIMITER ;

CALL GetTicketDetailsWithComplimentaryServices();



DELIMITER //
CREATE PROCEDURE GetCustomerByLastName()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE customer_id INT;
    DECLARE first_name VARCHAR(255);
    DECLARE last_name VARCHAR(255);

    -- Declare a cursor to fetch records
    DECLARE customer_cursor CURSOR FOR
        SELECT customer_id, first_name, last_name
        FROM customer
        WHERE last_name LIKE '%Scott'
        LIMIT 1;

    -- Declare a handler for when no records are found
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN customer_cursor;

    -- Fetch the first matching record
    FETCH customer_cursor INTO customer_id, first_name, last_name;

    -- Close the cursor
    CLOSE customer_cursor;
```

```
    -- Check if a record was found
    IF done = 0 THEN
        SELECT customer_id, first_name, last_name;
    ELSE
        SELECT 'No customer with last name ending with Scott found.';
    END IF;
END;
//
DELIMITER ;

CALL GetCustomerByLastName();
```