Stored Procedure

```
CREATE PROCEDURE GetDelayedFlights(
    IN p_threshold INT,
    IN p_start_date DATE,
    IN p_end_date DATE,
    IN p_airline_code VARCHAR(10),
    IN p_max_results INT
)
BEGIN
    -- Set default values if parameters are NULL
    SET p_threshold = IFNULL(p_threshold, 15);
    SET p_max_results = IFNULL(p_max_results, 100);

    -- First result set: Detailed flight information
    SELECT
        f.flightId,
        f.airlineCode,
        f.flightNumber,
        al.name AS airline_name,
        f.originAirport,
        orig.name AS origin_name,
        orig.city AS origin_city,
        f.destAirport,
        dest.name AS dest_name,
        dest.city AS dest_city,
        fs.flightDate,
        f.scheduledDepartureTime,
        f.scheduledArrivalTime,
        fs.actualDepartureTime,
        fs.actualArrivalTime,
        fs.departureDelay,
        fs.arrivalDelay,
        CASE
            WHEN fs.cancelled = 1 THEN 'Cancelled'
            WHEN fs.diverted = 1 THEN 'Diverted'
            WHEN fs.departureDelay > p_threshold THEN 'Delayed'
            ELSE 'On Time'
        END AS flight_status,
        CASE
            WHEN fs.weatherDelay > 0 THEN 'Weather'
            WHEN fs.carrierDelay > 0 THEN 'Carrier'
            WHEN fs.nasDelay > 0 THEN 'Air Traffic'
            WHEN fs.securityDelay > 0 THEN 'Security'
```

```
            WHEN fs.lateAircraftDelay > 0 THEN 'Late Aircraft'
            ELSE 'Unknown'
        END AS delay_reason,
        GREATEST(
            COALESCE(fs.weatherDelay, 0),
            COALESCE(fs.carrierDelay, 0),
            COALESCE(fs.nasDelay, 0),
            COALESCE(fs.securityDelay, 0),
            COALESCE(fs.lateAircraftDelay, 0)
        ) AS primary_delay_minutes
    FROM
        Flight f
    JOIN
        Flight_Status fs ON f.flightId = fs.flightId
    JOIN
        Airline al ON f.airlineCode = al.airlineCode
    JOIN
        Airport orig ON f.originAirport = orig.airportCode
    JOIN
        Airport dest ON f.destAirport = dest.airportCode
    WHERE
        (fs.departureDelay > p_threshold OR fs.cancelled = 1)
        AND (p_start_date IS NULL OR fs.flightDate >= p_start_date)
        AND (p_end_date IS NULL OR fs.flightDate <= p_end_date)
        AND (p_airline_code IS NULL OR f.airlineCode = p_airline_code)
    ORDER BY
        fs.departureDelay DESC
    LIMIT p_max_results;

    -- Second result set: Summary statistics
    SELECT
        COUNT(*) AS total_delayed_flights,
        SUM(fs.cancelled) AS cancelled_flights,
        SUM(fs.diverted) AS diverted_flights,
        ROUND(AVG(fs.departureDelay), 2) AS avg_departure_delay,
        ROUND(AVG(fs.arrivalDelay), 2) AS avg_arrival_delay,
        ROUND(AVG(CASE WHEN fs.carrierDelay > 0 THEN fs.carrierDelay ELSE NULL END),
2) AS avg_carrier_delay,
        ROUND(AVG(CASE WHEN fs.weatherDelay > 0 THEN fs.weatherDelay ELSE NULL
END), 2) AS avg_weather_delay,
        ROUND(AVG(CASE WHEN fs.nasDelay > 0 THEN fs.nasDelay ELSE NULL END), 2) AS
avg_nas_delay
    FROM
        Flight f
```

```
    JOIN
        Flight_Status fs ON f.flightId = fs.flightId
    WHERE
        (fs.departureDelay > p_threshold OR fs.cancelled = 1)
        AND (p_start_date IS NULL OR fs.flightDate >= p_start_date)
        AND (p_end_date IS NULL OR fs.flightDate <= p_end_date)
        AND (p_airline_code IS NULL OR f.airlineCode = p_airline_code);
END
```

This stored procedure retrieves and analyzes delayed or cancelled flight data within specified parameters. It returns two result sets: detailed flight information with delay reasons and status classifications, plus aggregate statistics on delays and cancellations. The procedure allows filtering by date range, airline, and minimum delay threshold while providing default values for optional parameters.

Trigger:

```
DELIMITER //

CREATE TRIGGER before_flight_insert
BEFORE INSERT ON Flight
FOR EACH ROW
BEGIN
    DECLARE min_distance INT DEFAULT 100;
    DECLARE flight_hours DECIMAL(10,2);

    IF NEW.originAirport != NEW.destAirport THEN
        IF NEW.distance IS NULL OR NEW.distance = 0 THEN
            SET flight_hours = TIMESTAMPDIFF(MINUTE,
                                NEW.scheduledDepartureTime,
                                NEW.scheduledArrivalTime) / 60;

            IF flight_hours > 0 AND flight_hours < 24 THEN
                SET NEW.distance = GREATEST(ROUND(flight_hours * 500), min_distance);
            ELSE
                SET NEW.distance = min_distance;
            END IF;
        END IF;
    ELSE
        SET NEW.distance = min_distance;
    END IF;

    IF NEW.distance < min_distance THEN
```

```
        SET NEW.distance = min_distance;
    END IF;
END//

DELIMITER;
```

This trigger automatically calculates and sets a flight's distance when a new flight is inserted into the database, ensuring certain business rules are followed.

What the Trigger Does:

Sets a minimum flight distance of 100.

Declares a variable to store calculated flight hours.

First checks if origin and destination airports are different.

If distance is NULL or 0, it calculates flight hours based on scheduled departure/arrival times.

Estimates distance using flight hours × 500.

Ensures calculated distance is at least the minimum (100).

If origin and destination are the same, sets distance to minimum (for round trips or maintenance flights).

Transaction

```
CREATE PROCEDURE UpdateFlightStatusWithPrediction(
    IN p_flight_id VARCHAR(36),
    IN p_status_id VARCHAR(36),
    IN p_departure_delay FLOAT,
    IN p_arrival_delay FLOAT,
    IN p_cancelled BOOLEAN,
    IN p_weather_delay FLOAT,
    IN p_carrier_delay FLOAT
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
```

```
        RESIGNAL;
    END;

    -- Start transaction with REPEATABLE READ isolation level
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
    START TRANSACTION;

    -- Update flight status
    UPDATE Flight_Status
    SET
        departureDelay = p_departure_delay,
        arrivalDelay = p_arrival_delay,
        cancelled = p_cancelled,
        weatherDelay = p_weather_delay,
        carrierDelay = p_carrier_delay,
        actualDepartureTime = CASE
            WHEN p_departure_delay IS NOT NULL AND p_cancelled = 0 THEN
                (SELECT DATE_ADD(scheduledDepartureTime, INTERVAL p_departure_delay
MINUTE)
                 FROM Flight WHERE flightId = p_flight_id)
            ELSE NULL
        END,
        actualArrivalTime = CASE
            WHEN p_arrival_delay IS NOT NULL AND p_cancelled = 0 THEN
                (SELECT DATE_ADD(scheduledArrivalTime, INTERVAL p_arrival_delay MINUTE)
                 FROM Flight WHERE flightId = p_flight_id)
            ELSE NULL
        END
    WHERE statusId = p_status_id;

    -- Create a delay prediction if delay is significant
    IF (p_departure_delay > 15 OR p_arrival_delay > 15) AND p_cancelled = 0 THEN
        INSERT INTO Delay_Prediction (
            predictionId,
            flightId,
            predictionTime,
            predictedDepartureDelay,
            predictedArrivalDelay,
            notificationSent,
            predictionReason
        )
        VALUES (
            UUID(),
            p_flight_id,
```

```
            NOW(),
            p_departure_delay * 1.1,
            p_arrival_delay * 1.1,
            FALSE,
            CASE
                WHEN p_weather_delay > 0 THEN 'Weather delay predicted to increase'
                WHEN p_carrier_delay > 0 THEN 'Carrier delay predicted to increase'
                ELSE 'General delay predicted to increase'
            END
        );
    END IF;

    COMMIT;
END
```

This stored procedure updates flight status information while also creating delay predictions when significant delays occur. It's designed to handle flight operations data with transactional integrity.

Key Functions:
Flight Status Update:
Updates all key delay metrics in the Flight_Status table:

Departure delay in minutes

Arrival delay in minutes

Cancellation status

Weather-related delay

Carrier-related delay

Calculates and sets actual departure/arrival times by adding delays to scheduled times (only if flight isn't cancelled)

Delay Prediction Creation:
When delays exceed 15 minutes and flight isn't cancelled:

Creates a new record in Delay_Prediction table

Predicts delays will worsen by 10% (multiplies by 1.1)

Sets prediction reason based on delay type (weather, carrier, or general)

Marks notification as not sent yet (FALSE)

Transaction Safety:
Uses REPEATABLE READ isolation level to prevent dirty reads

Includes comprehensive error handling that rolls back on failure

Ensures all updates happen atomically.