Task 6: Stored Procedures and Functions

```
-- Stored Procedure to update available seats
DELIMITER //
CREATE PROCEDURE UpdateAvailableSeats(IN train_id INT)
BEGIN
    UPDATE Trains
    SET TotalSeats = TotalSeats - 1
    WHERE TrainID = train_id;
END;
//
DELIMITER ;

-- Function to calculate total travel time in hours
DELIMITER //
CREATE FUNCTION CalculateTravelTime(start DATETIME, end DATETIME)
RETURNS INT
DETERMINISTIC
BEGIN
    RETURN TIMESTAMPDIFF(HOUR, start, end);
END;
//
DELIMITER ;
```

Task 7: Views and Indexes

-- View

```
CREATE VIEW PassengerBookingsView AS
SELECT P.FirstName, P.LastName, T.TrainName, B.BookingDate, B.SeatNumber
FROM Passengers P
JOIN Bookings B ON P.PassengerID = B.PassengerID
JOIN Schedules S ON B.ScheduleID = S.ScheduleID
JOIN Trains T ON S.TrainID = T.TrainID;
```

-- Index

```
CREATE INDEX idx_booking_date ON Bookings(BookingDate);
```

Task 8: Temporary Tables

```
CREATE TEMPORARY TABLE TempSchedule AS
SELECT * FROM Schedules
WHERE DATE(DepartureTime) = '2023-10-15';

SELECT * FROM TempSchedule;
```

Task 9: Cursors

```
DELIMITER //
CREATE PROCEDURE CheckFrequentPassengers()
```

```
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pid INT;
    DECLARE cur CURSOR FOR
        SELECT PassengerID FROM Bookings
        GROUP BY PassengerID
        HAVING COUNT(*) > 5;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO pid;
        IF done THEN
            LEAVE read_loop;
        END IF;
        -- Action (e.g., SELECT or log)
    END LOOP;
    CLOSE cur;
END;
//
DELIMITER ;

Task 10: ACID and Transactions
------------------------------

START TRANSACTION;
UPDATE Trains SET TotalSeats = TotalSeats - 1 WHERE TrainID = 101;
```

```
SAVEPOINT BeforeDelete;

DELETE FROM Bookings WHERE BookingID = 5;

-- If error occurs:

-- ROLLBACK TO BeforeDelete;

-- COMMIT at the end:

COMMIT;
```

Task 11: Triggers

```
-- Trigger to assign seat number

DELIMITER //

CREATE TRIGGER AssignSeatBeforeInsert

BEFORE INSERT ON Bookings

FOR EACH ROW

BEGIN

        SET NEW.SeatNumber = (SELECT MAX(SeatNumber)+1 FROM Bookings WHERE ScheduleID =

NEW.ScheduleID);

END;

//

DELIMITER ;

-- Trigger to update seats

DELIMITER //

CREATE TRIGGER UpdateSeatsAfterBooking

AFTER INSERT ON Bookings

FOR EACH ROW
```

```
BEGIN
    UPDATE Trains
    SET TotalSeats = TotalSeats - 1
    WHERE TrainID = (SELECT TrainID FROM Schedules WHERE ScheduleID = NEW.ScheduleID);
END;
//
DELIMITER ;
```

Task 12: UNION and UNION ALL

```
-- Passengers booked on two different routes
SELECT * FROM Bookings WHERE ScheduleID IN (1)
UNION
SELECT * FROM Bookings WHERE ScheduleID IN (2);

-- All bookings on different dates
SELECT * FROM Bookings WHERE BookingDate = '2023-05-01'
UNION ALL
SELECT * FROM Bookings WHERE BookingDate = '2023-06-01';
```

Task 13: Copying Tables

```
-- Structure copy
CREATE TABLE OldPassengers LIKE Passengers;
```

-- Data copy

CREATE TABLE ArchivedBookings AS

SELECT * FROM Bookings;

Additional Assignment: Keys Practice

-- Super Keys:

(EmpID), (EmpID, FirstName), (EmpID, LastName), (EmpID, Email), etc.

-- Candidate Keys:

(EmpID)

-- Primary Key:

EmpID (Unique and Not Null)

-- Alternate Keys:

None in this dataset; if Email was unique, it would be an Alternate Key.

-- Composite Key:

In EmployeeProjects: Composite Key = (EmpID, ProjectID)

```
CREATE TABLE EmployeeProjects (
    EmpID INT,
    ProjectID VARCHAR(10),
    PRIMARY KEY (EmpID, ProjectID)
);
```