Task 3: Joins and Queries

```
-- Query 1
SELECT T.TrainName, R.StartStation, R.EndStation, S.DepartureTime, S.ArrivalTime
FROM Trains T
INNER JOIN Schedules S ON T.TrainID = S.TrainID
INNER JOIN Routes R ON S.RouteID = R.RouteID;

-- Query 2
SELECT T.*
FROM Trains T
LEFT JOIN Schedules S ON T.TrainID = S.TrainID
LEFT JOIN Bookings B ON S.ScheduleID = B.ScheduleID
WHERE B.BookingID IS NULL;

-- Query 3
SELECT P.*
FROM Passengers P
RIGHT JOIN Bookings B ON P.PassengerID = B.PassengerID
JOIN Schedules S ON B.ScheduleID = S.ScheduleID
JOIN Routes R ON S.RouteID = R.RouteID
WHERE R.Distance > 500;

-- Query 4
SELECT S.*, B.BookingID
```

```
FROM Schedules S
LEFT JOIN Bookings B ON S.ScheduleID = B.ScheduleID;
```

Task 4: Normalization to 3NF

All tables are already designed to meet 3NF:

- Atomic values
- No partial dependencies
- No transitive dependencies

Task 5: Sub Queries

```
-- Query 5
SELECT R.RouteID, COUNT(B.PassengerID) AS TotalPassengers
FROM Routes R
JOIN Schedules S ON R.RouteID = S.RouteID
JOIN Bookings B ON S.ScheduleID = B.ScheduleID
GROUP BY R.RouteID;

-- Query 6
SELECT AVG(PassengerCount) AS AvgPassengers
FROM (
    SELECT COUNT(*) AS PassengerCount
    FROM Bookings
    GROUP BY ScheduleID
```

```
) AS Sub;


-- Query 7
SELECT TrainID
FROM (
    SELECT S.TrainID, COUNT(*) AS BookingCount
    FROM Bookings B
    JOIN Schedules S ON B.ScheduleID = S.ScheduleID
    GROUP BY S.TrainID
) AS Sub
ORDER BY BookingCount DESC
LIMIT 1;


-- Query 8
SELECT R.RouteID, COUNT(*) AS TotalSeatsBooked
FROM Bookings B
JOIN Schedules S ON B.ScheduleID = S.ScheduleID
JOIN Routes R ON S.RouteID = R.RouteID
WHERE B.BookingDate BETWEEN '2023-01-01' AND '2023-12-31'
GROUP BY R.RouteID;


-- Query 9
SELECT * FROM Bookings
WHERE PassengerID IN (
    SELECT PassengerID FROM Passengers WHERE Age > 60
);
```

-- Next tasks continued in next part