

ADVANCE DATA BASE ASSESSMENT REPORT

Marvis Osazee Osazuwa

Title: Airport Ticketing System Database Design

1. Introduction

This report presents the design and implementation of a relational database system for managing flight ticketing and e-boarding at an airport. Acting as a database consultant, I was tasked with creating a system that handles reservations, ticket issuance, employees, passengers, flights, baggage, and additional services.

The solution is built using Microsoft SQL Server and developed entirely with T-SQL. It includes normalized tables (up to 3NF), constraints, views, stored procedures, triggers, and user-defined functions. This report outlines the design process, key SQL implementations, and considerations for data integrity, concurrency, security, and backup.

2. Database Design Process

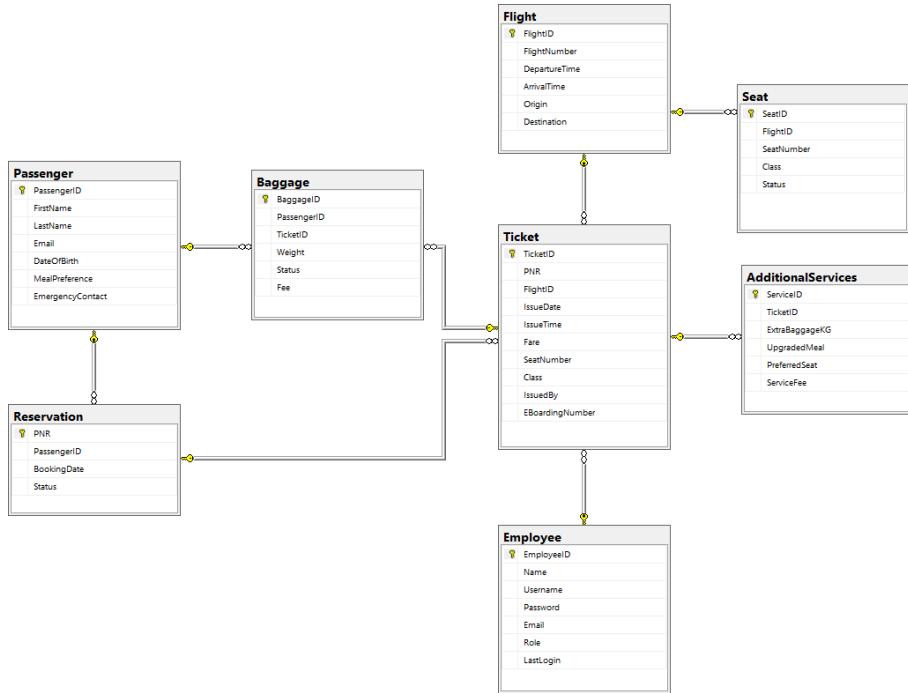
2.1 Database Normalization

I have designed a normalized database for the airport ticketing system based on the client requirements. The design follows 3rd Normal Form (3NF) to eliminate redundancy and ensure data integrity. Here's my approach:

1. **First Normal Form (1NF):** Ensured each table has a primary key and all attributes contain atomic values
2. **Second Normal Form (2NF):** Removed partial dependencies by separating entities into different tables
3. **Third Normal Form (3NF):** Removed transitive dependencies to ensure non-key attributes depend only on the primary key.

2.2 Database Schema

Entity-Relationship Diagram



The ER (Entity-Relationship) diagram visually represents the structure and relationships of data within a database. Foreign keys in the ER diagram enable the joining of tables and retrieval of related data, ensuring consistency and meaningful connections across different entities.

2.3 Tables Structure

The figures below show the structure of tables in the database.

```
1  -- Create the database
2  CREATE DATABASE AirWaveExpressTicketingSystem;
3  GO
4
5  USE AirWaveExpressTicketingSystem;
6  GO
7
8  -- Create the Employee table
9  CREATE TABLE Employee (
10     EmployeeID VARCHAR(10) PRIMARY KEY,
11     Name NVARCHAR(100) NOT NULL,
12     Username NVARCHAR(50) NOT NULL UNIQUE,
13     Password NVARCHAR(100) NOT NULL,
14     Email NVARCHAR(100) NOT NULL UNIQUE,
15     Role NVARCHAR(20) NOT NULL CHECK (Role IN ('Ticketing Staff', 'Ticketing Supervisor')),
16     LastLogin DATETIME NULL
17 );
18 GO
19
20 -- Create a new sequence for generating unique EmployeeID
21 CREATE SEQUENCE EmployeeSeq
22     START WITH 200 -- Start from 200 to avoid conflict with existing EmployeeID
23     INCREMENT BY 1;
24 GO
25 -- Create a new trigger to generate the EmployeeID using the sequence
26 CREATE TRIGGER trgGenerateEmployeeID
27 ON Employee
28 INSTEAD OF INSERT
29 AS
30 BEGIN
31     DECLARE @NextID INT;
32
33     -- Get the next value from the sequence
34     SELECT @NextID = NEXT VALUE FOR EmployeeSeq;
35
36     -- Insert the new employee with the generated EmployeeID (prefix 'EM' + sequence number)
37     INSERT INTO Employee (EmployeeID, Username, Password, Name, Email, Role, LastLogin)
38     SELECT 'EM' + RIGHT('000' + CAST(@NextID AS VARCHAR(10)), 3), Username, Password, Name, Email, Role, LastLogin
39     FROM INSERTED;
40 END;
41 GO
42 -- Create Passenger Table
43 CREATE TABLE Passenger (
44     PassengerID INT PRIMARY KEY IDENTITY(2000,1),
45     FirstName NVARCHAR(50) NOT NULL,
46     LastName NVARCHAR(50) NOT NULL,
47     Email NVARCHAR(100) NOT NULL UNIQUE,
48     DateOfBirth DATE NOT NULL,
49     MealPreference NVARCHAR(20) CHECK (MealPreference IN ('Vegetarian', 'Non-Vegetarian')),
50     EmergencyContact NVARCHAR(50) NULL
51 );
52 GO
53
54 -- Create Flight Table
55 CREATE TABLE Flight (
56     FlightID INT PRIMARY KEY IDENTITY(3000,1),
57     FlightNumber VARCHAR(10),
58     DepartureTime DATETIME NOT NULL,
59     ArrivalTime DATETIME NOT NULL,
60     Origin NVARCHAR(50) NOT NULL,
61     Destination NVARCHAR(50) NOT NULL,
62     CONSTRAINT CHK_ArrivalAfterDeparture CHECK (ArrivalTime > DepartureTime)
63 );
64 GO
65
66 -- Create sequence to generate numeric part of FlightNumber
67 CREATE SEQUENCE FlightSeq
68     START WITH 100 -- Starting value for FlightNumber
69     INCREMENT BY 1; -- Increment by 1
70 GO
71
72 -- Create trigger to automatically generate FlightNumber based on FlightSeq
73 CREATE TRIGGER trgGenerateFlightNumber
74 ON Flight
75 INSTEAD OF INSERT
76 AS
77 BEGIN
78     DECLARE @NextID INT;
79
80     -- Get the next value from the sequence
81     SELECT @NextID = NEXT VALUE FOR FlightSeq;
82
83     -- Insert the record with the generated FlightNumber
84     INSERT INTO Flight (FlightNumber, DepartureTime, ArrivalTime, Origin, Destination)
85     SELECT 'A' + RIGHT('000' + CAST(@NextID AS VARCHAR(3)), 3), DepartureTime, ArrivalTime, Origin, Destination
86     FROM INSERTED;
87 END;
88 GO
89
90 -- Create Reservation Table
91 CREATE TABLE Reservation (
92     PNR CHAR(10) PRIMARY KEY,
93     PassengerID INT NOT NULL,
94     BookingDate DATETIME NOT NULL,
95     Status NVARCHAR(20) NOT NULL CHECK (Status IN ('Confirmed', 'Pending', 'Cancelled')),
96     FOREIGN KEY (PassengerID) REFERENCES Passenger(PassengerID)
97 );
98 GO
```

```

100  -- Create Ticket Table
101  CREATE TABLE Ticket (
102      TicketID INT PRIMARY KEY IDENTITY(4000,1),
103      PNR CHAR(10) NOT NULL,
104      FlightID INT NOT NULL,
105      IssueDate DATE NOT NULL DEFAULT GETDATE(),
106      IssueTime TIME NOT NULL DEFAULT CONVERT(TIME, GETDATE()),
107      Fare DECIMAL(10,2) NOT NULL,
108      SeatNumber NVARCHAR(10) NULL,
109      Class NVARCHAR(20) NOT NULL CHECK (Class IN ('Economy', 'Business', 'FirstClass')),
110      IssuedBy VARCHAR(10) NOT NULL,
111      EBoardingNumber NVARCHAR(20) UNIQUE NOT NULL,
112      FOREIGN KEY (PNR) REFERENCES Reservation(PNR),
113      FOREIGN KEY (FlightID) REFERENCES Flight(FlightID),
114      FOREIGN KEY (IssuedBy) REFERENCES Employee(EmployeeID)
115  );
116  GO
117
118  -- AdditionalServices Table
119  CREATE TABLE AdditionalServices (
120      ServiceID INT PRIMARY KEY IDENTITY(5000,1),
121      TicketID INT NOT NULL,
122      ExtraBaggageKG DECIMAL(5,2) NULL DEFAULT 0,
123      UpgradedMeal BIT NULL DEFAULT 0,
124      PreferredSeat BIT NULL DEFAULT 0,
125      ServiceFee DECIMAL(10,2) NOT NULL DEFAULT 0,
126      FOREIGN KEY (TicketID) REFERENCES Ticket(TicketID)
127  );
128  GO
129
130  -- Create Baggage Table
131  CREATE TABLE Baggage (
132      BaggageID VARCHAR(15) PRIMARY KEY, -- Alphanumeric BaggageID (BG + PassengerID)
133      PassengerID INT NOT NULL, -- Foreign Key referencing PassengerID
134      TicketID INT NOT NULL,
135      Weight DECIMAL(5,2) NOT NULL,
136      Status NVARCHAR(20) NOT NULL CHECK (Status IN ('CheckedIn', 'Loaded')),
137      Fee DECIMAL(10,2) NOT NULL,
138      FOREIGN KEY (PassengerID) REFERENCES Passenger(PassengerID), -- Foreign key constraint
139      FOREIGN KEY (TicketID) REFERENCES Ticket(TicketID) -- Assuming Ticket table exists
140  );
141  GO
142
143  -- Create trigger to generate BaggageID
144  CREATE TRIGGER trgGenerateBaggageID
145  ON Baggage
146  INSTEAD OF INSERT
147  AS
148  BEGIN
149      DECLARE @PassengerID INT;
150
151      -- Get the PassengerID from the inserted row
152      SELECT @PassengerID = PassengerID FROM INSERTED;
153
154      -- Insert the record with the generated BaggageID (BG + PassengerID)
155      INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee)
156      SELECT 'BG' + CAST(@PassengerID AS VARCHAR(10)), PassengerID, TicketID, Weight, Status, Fee
157      FROM INSERTED;
158  END;
159  GO

```

2.4 Justification of Design Decisions

1. Employee Table

- a. EmployeeID is auto generated with a trigger and sequence, giving IDs like "EM200"
- b. Only two roles are allowed 'Ticketing Staff' or 'Ticketing Supervisor'
- c. Both Username and Email must be unique to avoid duplicates
- d. Passwords are stored as text for now but should be hashed in a real system
- e. LastLogin is optional and tracks the last time the staff logged in

2. Passenger Table

- a. PassengerID starts from 2000 and increases automatically
- b. Email must be unique—no two passengers can share the same email

- c. Meal preferences are optional and limited to 'Vegetarian' or 'Non-Vegetarian'
 - d. Emergency contact info is also optional
3. **Flight Table**
- a. Each flight gets a unique ID starting from 3000
 - b. The system ensures that ArrivalTime is always after DepartureTime
 - c. Flight numbers are auto created using a trigger and look like "AW101", "AW102", etc.
4. **Reservation Table**
- a. PNR (Passenger Name Record) is used as the primary key—it's a unique code
 - b. Status is limited to 'Confirmed', 'Pending', or 'Cancelled'
 - c. Each reservation is linked to a specific passenger
5. **Ticket Table**
- a. TicketID auto-increments from 4000
 - b. IssueDate and IssueTime are recorded automatically when the ticket is created
 - c. SeatNumber is optional if a seat hasn't been selected
 - d. E-Boarding numbers must be unique for each ticket
 - e. Ticket Class is restricted to 'Economy', 'Business', or 'FirstClass'
 - f. Each ticket is tied to a reservation, flight, and issuing employee
6. **Additional Services Table**
- a. Tracks extra services linked to a ticket
 - b. Includes yes/no options like upgraded meals or preferred seats (default is 'No')
 - c. Service fees are calculated, starting at 0 by default
7. **Baggage Table**
- a. BaggageID is auto generated with a trigger and looks like "BG2000" for PassengerID 2000
 - b. Baggage status can only be 'CheckedIn' or 'Loaded'
 - c. Weight is stored with decimal accuracy
 - d. Each baggage entry is linked to both a passenger and a ticket
8. **Seat Table**
- a. SeatID auto-increments from 1
 - b. Each seat is linked to a specific flight via FlightID
 - c. Class is limited to 'Economy', 'Business', or 'FirstClass'
 - d. Seat status can be 'Available', 'Reserved', or 'Occupied'—default is 'Available'
 - e. No two seats can have the same number on the same flight (enforced with a unique constraint)

2.5 Data Population

The figures below show the population of the database and display of the respective tables. It was populated with data of 5 employees, 50 passengers and 2 flights that

served as basis for the data in other tables.

The screenshot shows the SSMS interface with the following details:

- SQL Server Object Explorer:** Shows the database structure under `(localdb)\MSSQLLocalDB`, including the `AirWaveExpressTicketingSystem` database which contains tables like `Flight`, `Employee`, `Passenger`, etc.
- Script Editor:** The script pane displays T-SQL code for inserting flight records and employees into the `Flight` and `Employee` tables.
- Results Pane:** Shows the output of the `SELECT * FROM Employee` query, listing six employees with their details.
- Status Bar:** Shows the message "Query executed successfully at 03:24:07" and the session information "(localdb)\MSSQLLocalDB (15... | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 6 rows".

```

VALUES ('EM206', 'ifeanyiUgwu', 'safePass789', 'ifeanyi Ugwu', 'ifeanyi.ugwu@airwave.com', 'Ticketing Staff', '2025-04-23 11:00:00')
GO
-- Insert sample flight records
INSERT INTO Flight (FlightNumber, DepartureTime, ArrivalTime, Origin, Destination)
VALUES ('AW001', '2025-05-01 08:00:00', '2025-05-01 12:00:00', 'Tokyo Narita', 'Los Angeles');
GO
INSERT INTO Flight (FlightNumber, DepartureTime, ArrivalTime, Origin, Destination)
VALUES ('AW002', '2025-05-02 09:00:00', '2025-05-02 13:00:00', 'Lisbon', 'New York JFK');
GO
Select* from Employee;

```

EmployeeID	Name	Username	Password	Email	Role	LastLogin
1	Chinedu Okeke	chinedu.okeke	securePass123	chinedu.okeke@airwave.com	Ticketing Staff	2025-04-23 08:30:00.000
2	Adsobi Nwosu	adsobi.nwosu	strongPass456	adsobi.nwosu@airwave.com	Ticketing Supervisor	2025-04-23 15:00:00.000
3	Uchedukwu Eze	uchedukwu.eze	passWord321	uchedukwu.eze@airwave.com	Ticketing Staff	2025-04-23 11:00:00.000
4	Nkechi Okafor	nkechi.okafor	passWord654	nkechi.okafor@airwave.com	Ticketing Supervisor	2025-04-23 11:30:00.000
5	Chukwudi Nnaji	chukwudi.nnaji	passWord987	chukwudi.nnaji@airwave.com	Ticketing Staff	2025-04-23 12:00:00.000
6	Ifeanyi Ugwu	ifeanyiUgwu	safePass789	ifeanyi.ugwu@airwave.com	Ticketing Staff	2025-04-23 10:00:00.000

The screenshot shows the SSMS interface with the following details:

- SQL Server Object Explorer:** Shows the database structure under `(localdb)\MSSQLLocalDB`, including the `AirWaveExpressTicketingSystem` database which contains tables like `Passenger`.
- Script Editor:** The script pane displays T-SQL code for inserting passenger records into the `Passenger` table.
- Results Pane:** Shows the output of the `SELECT * FROM Passenger` query, listing seven passengers with their details.
- Status Bar:** Shows the message "Query executed successfully at 03:42:00" and the session information "(localdb)\MSSQLLocalDB (15... | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 46 rows".

```

INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Arisa', 'Wong', 'arisawong@example.com', '1985-05-15', 'Vegetarian', 'Yuki Tanaka')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Diego', 'Torres', 'diegotorres@example.com', '1990-08-22', 'Non-Vegetarian', 'Ken Salo')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Valentina', 'Diaz', 'valentinadiaz@example.com', '1982-03-10', 'Vegetarian', 'Maria Silva')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('James', 'Miller', 'jamesmiller@example.com', '1995-11-30', 'Non-Vegetarian', 'Pedro Costa')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Sophia', 'Davis', 'sophiadavis@example.com', '1998-07-19', 'Vegetarian', 'Lucas Oliveira')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('John', 'Kumar', 'johnkumar@example.com', '1988-04-25', 'Non-Vegetarian', 'Rafael Santos')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Adi', 'Singh', 'adi.singh@example.com', '1992-01-05', 'Vegetarian', 'Wanju Kamau')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Nikos', 'Papadopoulos', 'nikos.papadopoulos@example.com', '1993-09-01', 'Non-Vegetarian', 'Eleni Georgiou')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Sakura', 'Yamamoto', 'sakura.yamamoto@example.com', '1997-06-01', 'Non-Vegetarian', 'Ren Kobayashi')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Miguel', 'Ferreira', 'miguel.ferreira@example.com', '1999-12-01', 'Vegetarian', 'Beatriz Martins')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Lucas', 'Costa', 'lucas.costa@example.com', '2000-03-01', 'Non-Vegetarian', 'Mariana Lima')
INSERT INTO Passenger (FirstName, LastName, Email, DateOfBirth, MealPreference, EmergencyContact) VALUES ('Marta', 'Silva', 'marta.silva@example.com', '2001-07-01', 'Non-Vegetarian', 'Daniela Oliveira')
GO

```

PassengerID	FirstName	LastName	Email	DateOfBirth	MealPreference	EmergencyContact
1	Haruto	Tanaka	haruto.tanaka@gmail.com	1985-05-15	Vegetarian	Yuki Tanaka
2	Yu	Sato	yu.sato@yahoo.com	1990-08-22	Non-Vegetarian	Ken Salo
3	João	Silva	joao.silva@aol.com	1982-03-10	Vegetarian	Maria Silva
4	Ana	Costa	ana.costa@gmx.com	1995-11-30	Non-Vegetarian	Pedro Costa
5	Gabriel	Oliveira	gabriel.oliveira@hotmail.com	1988-07-19	Vegetarian	Lucas Oliveira
6	Isabela	Santos	isabela.santos@proton.me	1992-04-25	Non-Vegetarian	Rafael Santos
7	Mwangi	Kamau	mwangi.kamau@gmail.com	1980-01-05	Vegetarian	Wanju Kamau

SQL Server Object Explorer

```

234 GO
235
236 -- Insert sample flight records
237 INSERT INTO Flight (FlightNumber, DepartureTime, ArrivalTime, Origin, Destination)
238 VALUES ('AW001', '2025-05-01 08:00:00', '2025-05-01 12:00:00', 'Tokyo Narita', 'Los Angeles');
239 GO
240
241 INSERT INTO Flight (FlightNumber, DepartureTime, ArrivalTime, Origin, Destination)
242 VALUES ('AW002', '2025-05-02 09:00:00', '2025-05-02 13:00:00', 'Lisbon', 'New York JFK');
243 GO
244
245 SELECT * FROM Flight;

```

FlightID	FlightNumber	DepartureTime	ArrivalTime	Origin	Destination
1	3000	AW100	2025-05-01 08:00:00.000	2025-05-01 12:00:00.000	Tokyo Narita Los Angeles
2	3001	AW101	2025-05-02 09:00:00.000	2025-05-02 13:00:00.000	Lisbon New York JFK
3	3002	AW102	2025-05-01 08:00:00.000	2025-05-01 12:00:00.000	Tokyo Narita Los Angeles
4	3003	AW103	2025-05-02 09:00:00.000	2025-05-02 13:00:00.000	Lisbon New York JFK

Query executed successfully at 03:28:56

SQL Server Object Explorer

```

291 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0041', '2043', '2025-04-28 11:30:00', 'Cancelled');
292 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0045', '2044', '2025-04-28 12:00:00', 'Pending');
293 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0046', '2045', '2025-04-29 08:30:00', 'Confirmed');
294 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0047', '2046', '2025-04-29 09:00:00', 'Pending');
295 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0048', '2047', '2025-04-29 10:00:00', 'Cancelled');
296 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0049', '2048', '2025-04-29 11:15:00', 'Confirmed');
297 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status) VALUES ('PNR0050', '2049', '2025-04-29 12:00:00', 'Pending');
298 GO
299
300

```

SELECT * FROM Reservation

PNR	PassengerID	BookingDate	Status	
1	PNR0001	2000	2025-04-20 08:00:00.000	Confirmed
2	PNR0002	2001	2025-04-20 09:00:00.000	Pending
3	PNR0003	2002	2025-04-20 10:00:00.000	Confirmed
4	PNR0004	2003	2025-04-20 11:00:00.000	Cancelled
5	PNR0005	2004	2025-04-20 12:00:00.000	Pending
6	PNR0006	2005	2025-04-21 08:15:00.000	Confirmed
7	PNR0007	2006	2025-04-21 09:30:00.000	Pending
8	PNR0008	2007	2025-04-21 10:45:00.000	Cancelled
9	PNR0009	2008	2025-04-21 11:30:00.000	Confirmed
10	PNR0010	2009	2025-04-21 12:00:00.000	Pending
11	PNR0011	2010	2025-04-22 08:00:00.000	Confirmed
12	PNR0012	2011	2025-04-22 09:00:00.000	Cancelled
13	PNR0013	2012	2025-04-22 10:30:00.000	Pending
14	PNR0014	2013	2025-04-22 11:30:00.000	Confirmed
15	PNR0015	2014	2025-04-22 12:15:00.000	Pending
16	PNR0016	2015	2025-04-23 08:30:00.000	Confirmed

Query executed successfully at 03:50:00

SQL Server Object Explorer

```

446
447     INSERT INTO Ticket (PNR, FlightID, Fare, SeatNumber, Class, IssuedBy, EBoardingNumber)
448     VALUES ('PNR0049', 3000, 383.75, 'A49', 'Business', 'EM202', 'EBN0049');
449
450     INSERT INTO Ticket (PNR, FlightID, Fare, SeatNumber, Class, IssuedBy, EBoardingNumber)
451     VALUES ('PNR0050', 3001, 387.5, 'A50', 'FirstClass', 'EM203', 'EBN0050');
452     GO
453
454     select * from Ticket

```

T-SQL

	TicketID	PNR	FlightID	IssueDate	IssueTime	Fare	SeatNumber	Class	IssuedBy	EBoardingNumber
1	4088	PNR0001	3000	2025-04-25	04:07:10.630000	203.75	A1	Business	EM202	EBN0001
2	4089	PNR0002	3001	2025-04-25	04:07:10.630000	207.50	A2	FirstClass	EM203	EBN0002
3	4090	PNR0003	3002	2025-04-25	04:07:10.630000	211.25	A3	Economy	EM204	EBN0003
4	4091	PNR0004	3003	2025-04-25	04:07:10.630000	215.00	A4	Business	EM205	EBN0004
5	4093	PNR0006	3001	2025-04-25	04:07:10.630000	222.50	A6	Economy	EM201	EBN0006
6	4094	PNR0007	3002	2025-04-25	04:07:10.630000	226.25	A7	Business	EM202	EBN0007
7	4095	PNR0008	3003	2025-04-25	04:07:10.630000	230.00	A8	FirstClass	EM203	EBN0008
8	4096	PNR0009	3000	2025-04-25	04:07:10.630000	233.75	A9	Economy	EM204	EBN0009
9	4097	PNR0010	3001	2025-04-25	04:07:10.630000	237.50	A10	Business	EM205	EBN0010
10	4099	PNR0012	3003	2025-04-25	04:07:10.630000	245.00	A12	Economy	EM201	EBN0012
11	4100	PNR0013	3000	2025-04-25	04:07:10.630000	248.75	A13	Business	EM202	EBN0013
12	4101	PNR0014	3001	2025-04-25	04:07:10.630000	252.50	A14	FirstClass	EM203	EBN0014
13	4102	PNR0015	3002	2025-04-25	04:07:10.630000	256.25	A15	Economy	EM204	EBN0015
14	4103	PNR0016	3003	2025-04-25	04:07:10.630000	260.00	A16	Business	EM205	EBN0016
15	4105	PNR0018	3001	2025-04-25	04:07:10.630000	267.50	A18	Economy	EM201	EBN0018
16	4106	PNR0019	3002	2025-04-25	04:07:10.630000	271.25	A19	Business	EM202	EBN0019
17	4107	PNR0020	3003	2025-04-25	04:07:10.630000	275.00	A20	FlightClass	EM203	EBN0020

Query executed successfully at 04:07:36

Ready Select Repository 04/07 25/04/2025

SQL Server Object Explorer

```

546
547     VALUES (4132, 1.13, 0, 0, 5.65);
548
549     INSERT INTO AdditionalServices (TicketID, ExtraBaggageKG, UpgradedMeal, PreferredSeat, ServiceFee)
550     VALUES (4133, 29.78, 0, 0, 148.9);
551     GO
552
553     select * from AdditionalServices

```

T-SQL

	ServiceID	TicketID	ExtraBaggageKG	UpgradedMeal	PreferredSeat	ServiceFee
4	5053	4091	26.33	0	1	141.65
5	5055	4093	28.17	0	1	150.85
6	5056	4094	13.83	0	1	79.15
7	5057	4095	21.25	0	0	106.25
8	5058	4096	2.11	1	0	30.55
9	5059	4097	21.36	0	1	116.80
10	5061	4099	8.26	1	0	61.30
11	5062	4100	24.61	1	1	153.05
12	5063	4101	27.33	1	0	156.65
13	5064	4102	5.70	0	0	28.50
14	5065	4103	29.14	0	1	155.70
15	5067	4105	22.74	0	0	113.70
16	5068	4106	13.18	1	1	95.90
17	5069	4107	2.76	0	1	23.80
18	5070	4108	11.04	0	1	65.20
19	5071	4109	19.82	1	1	129.10
20	5073	4111	10.21	1	1	81.05
21	5074	4112	7.23	0	1	46.15
22	5075	4113	6.27	1	1	61.35
23	5076	4114	3.98	1	1	49.90

Query executed successfully at 04:19:09

Ready Select Repository 04/20 25/04/2025

The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. In the center, the 'aIRWAVE.sql' file is open in the T-SQL editor. A red arrow points to the 'select * from Baggage' command in the results pane. Below the results, a message indicates the query was executed successfully at 05:11:23.

```

570     INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee) VALUES ('BG3015', 2809, 4097, 19.96, 'Checkedin', 12.5)
571     INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee) VALUES ('BG3016', 2827, 4115, 25.42, 'Checkedin', 12.5)
572     INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee) VALUES ('BG3017', 2835, 4123, 15.53, 'Loaded', 12.5)
573     INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee) VALUES ('BG3018', 2841, 4129, 27.37, 'Checkedin', 12.5)
574     INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee) VALUES ('BG3019', 2844, 4132, 16.19, 'Checkedin', 12.5)
575     GO
576
577     select * from Baggage
  
```

	BaggageID	PassengerID	TicketID	Weight	Status	Fee
1	BG2000	2000	4088	24.01	Loaded	90.10
2	BG2001	2001	4089	27.67	Checkedin	126.70
3	BG2002	2002	4090	26.19	Checkedin	111.90
4	BG2003	2003	4091	27.75	Loaded	127.50
5	BG2005	2005	4093	25.98	Loaded	259.80
6	BG2008	2008	4096	24.95	Loaded	249.50
7	BG2010	2010	4101	26.82	Loaded	268.20
8	BG2013	2013	4101	28.78	Checkedin	267.80
9	BG2015	2015	4103	18.29	Checkedin	182.90
10	BG2018	2018	4106	25.21	Checkedin	102.10
11	BG2019	2019	4107	24.90	Loaded	98.00
12	BG2020	2020	4108	27.32	Checkedin	123.20
13	BG2021	2021	4109	22.19	Checkedin	76.90
14	BG2023	2023	4111	23.80	Loaded	238.00
15	BG2024	2024	4112	21.00	Loaded	210.00
16	BG2026	2026	4114	26.19	Loaded	261.90
17	BG2032	2032	4120	28.78	Loaded	137.80
18	BG2033	2033	4121	21.91	Checkedin	74.10

Query executed successfully at 05:11:23 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 18 rows

2.6 Database Objects

- Sequence

The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. In the center, the 'aIRWAVE.sql' file is open in the T-SQL editor. A red arrow points to the 'CREATE SEQUENCE FlightSeq' section of the script, which defines a sequence starting at 100 with an increment of 1.

```

54     -- Create Flight Table
55     CREATE TABLE Flight (
56         FlightID INT PRIMARY KEY IDENTITY(3000,1),
57         FlightNumber VARCHAR(10),
58         DepartureTime DATETIME NOT NULL,
59         ArrivalTime DATETIME NOT NULL,
60         Origin NVARCHAR(50) NOT NULL,
61         Destination NVARCHAR(50) NOT NULL,
62         CONSTRAINT CHK_ArrivalAfterDeparture CHECK (ArrivalTime > DepartureTime)
63     );
64     GO
65
66     -- Create sequence to generate numeric part of FlightNumber
67     CREATE SEQUENCE FlightSeq
68         START WITH 100 -- Starting value for FlightNumber
69         INCREMENT BY 1; -- Increment by 1
70     GO
71
72     -- Create trigger to automatically generate FlightNumber based on FlightSeq
73     CREATE TRIGGER trgGenerateFlightNumber
74     ON Flight
75     INSTEAD OF INSERT
76     AS
77     BEGIN
78         DECLARE @NextID INT;
  
```

Query executed successfully at 03:03:42 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 46 rows

This sequence generates incremental numbers used in a trigger to automatically create formatted FlightNumber values like AW100, AW101.

- *Stored Procedures*

```

CREATE PROCEDURE sp_IssueTicket
    @PNR NVARCHAR(10),
    @FlightID INT,
    @Fare DECIMAL(10,2),
    @SeatNumber NVARCHAR(10) = NULL,
    @Class NVARCHAR(20),
    @EmployeeID INT,
    @ExtraBaggageKG DECIMAL(5,2) = 0,
    @UpgradedMeal BIT = 0,
    @PreferredSeat BIT = 0
AS
BEGIN
    DECLARE @EBoardingNumber NVARCHAR(20)
    DECLARE @TicketID INT
    SET @EBoardingNumber = E + CAST(@FlightID AS NVARCHAR(10)) + CAST(DATEPART(YEAR, GETDATE()) AS NVARCHAR(4)) + CAST(DATEPART(MONTH, GETDATE()) AS NVARCHAR(2)) + CAST(DATEPART(DAY, GETDATE()) AS NVARCHAR(2))
    SET @TicketID = SCOPE_IDENTITY()

    -- Insert the new ticket
    INSERT INTO Ticket (PNR, FlightID, Fare, SeatNumber, Class, IssuedBy, EBoardingNumber)
    VALUES (@PNR, @FlightID, @Fare, @SeatNumber, @Class, @EmployeeID, @EBoardingNumber)

    -- Get the TicketID of the newly inserted ticket
    SELECT @TicketID AS TicketID

    -- Insert additional services if any
    INSERT INTO AdditionalServices (TicketID, ExtraBaggageKG, UpgradedMeal, PreferredSeat)
    VALUES (@TicketID, @ExtraBaggageKG, @UpgradedMeal, @PreferredSeat)

    -- Return the TicketID and EBoardingNumber
    SELECT @TicketID AS TicketID, @EBoardingNumber AS EBoardingNumber
END
GO

-- Procedure to authenticate employee
CREATE PROCEDURE sp_AuthenticateEmployee
    @Username NVARCHAR(50),
    @Password NVARCHAR(50)
AS
SELECT *
FROM Employee
WHERE Username = @Username AND Password = @Password

```

Query completed with errors.

This is the procedure to issue a new ticket.

- *Views*

```

CREATE VIEW vw_EmployeeActivity AS
SELECT
    e.EmployeeID,
    e.Name,
    e.Role,
    COUNT(t.TicketID) AS TicketsIssued,
    SUM(t.Fare + ISNULL(a.ServiceFee, 0)) AS TotalRevenueGenerated
FROM Employee e
LEFT JOIN Ticket t ON e.EmployeeID = t.IssuedBy
LEFT JOIN AdditionalServices a ON t.TicketID = a.TicketID
GROUP BY e.EmployeeID, e.Name, e.Role
GO

-- Trigger to calculate and update service fee when additional services are modified
CREATE TRIGGER tr_UpdateServiceFee
ON AdditionalServices
AFTER INSERT, UPDATE
AS

```

Command(s) completed successfully.

View on employee activity.

- *Triggers*

```

141 GO
142
143 -- Create trigger to generate BaggageID
144 CREATE TRIGGER trgGenerateBaggageID
145 ON Baggage
146 INSTEAD OF INSERT
147 AS
148 BEGIN
149     DECLARE @PassengerID INT;
150
151     -- Get the PassengerID from the inserted row
152     SELECT @PassengerID = PassengerID FROM INSERTED;
153
154     -- Insert the record with the generated BaggageID (BG + PassengerID)
155     INSERT INTO Baggage (BaggageID, PassengerID, TicketID, Weight, Status, Fee)
156     SELECT 'BG' + CAST(@PassengerID AS VARCHAR(10)), PassengerID, TicketID, Weight, Status, Fee
157     FROM INSERTED;
158
159 END;
160 GO

```

Trigger to generate Baggage ID by adding 'BG' to the passengerID.

- *User-Defined Functions*

```

764 RAISERROR('Ticket details cannot be modified after issuance', 16, 1)
765 ROLLBACK TRANSACTION
766
767 END
768 GO
769
770 --DATABASE OBJECTS - USER DEFINED FUNCTIONS
771 |-- Function to calculate total fare for a ticket
772 CREATE FUNCTION fn_CalculateTotalFare (@TicketID INT)
773 RETURNS DECIMAL(10,2)
774 AS
775 BEGIN
776     DECLARE @TotalFare DECIMAL(10,2)
777
778     SELECT @TotalFare = t.Fare + ISNULL(a.ServiceFee, 0)
779     FROM Ticket t
780     LEFT JOIN AdditionalServices a ON t.TicketID = a.TicketID
781     WHERE t.TicketID = @TicketID
782
783     RETURN @TotalFare
784
785 END
786 GO

```

Command(s) completed successfully.

Query executed successfully at 05:20:20

This function is to calculate total fare for a ticket

2.7 Data Integrity and Security Considerations

1. **Constraints:**
 - 1.1. CHECK constraints ensure valid values for roles, statuses, and meal preferences
 - 1.2. UNIQUE constraints prevent duplicate usernames, emails, and boarding numbers
 - 1.3. FOREIGN KEY constraints maintain referential integrity
2. **Security:**
 - 2.1. Authentication stored procedure handles employee login
 - 2.2. In production, passwords would be hashed and salted
 - 2.3. Role-based access would be implemented at application level
3. **Transaction Management:**
 - 3.1. Critical operations like ticket issuance are wrapped in transactions
 - 3.2. Triggers help maintain data consistency

3 Implementation of T-SQL Requirements

3.1 Q2. Adding Constraint for Reservation Date Validation

I added a constraint to ensure that reservation dates cannot be in the past. This is an important business rule to maintain data integrity in the ticketing system.

- *Implementation*

The screenshot shows the SSMS interface with the following details:

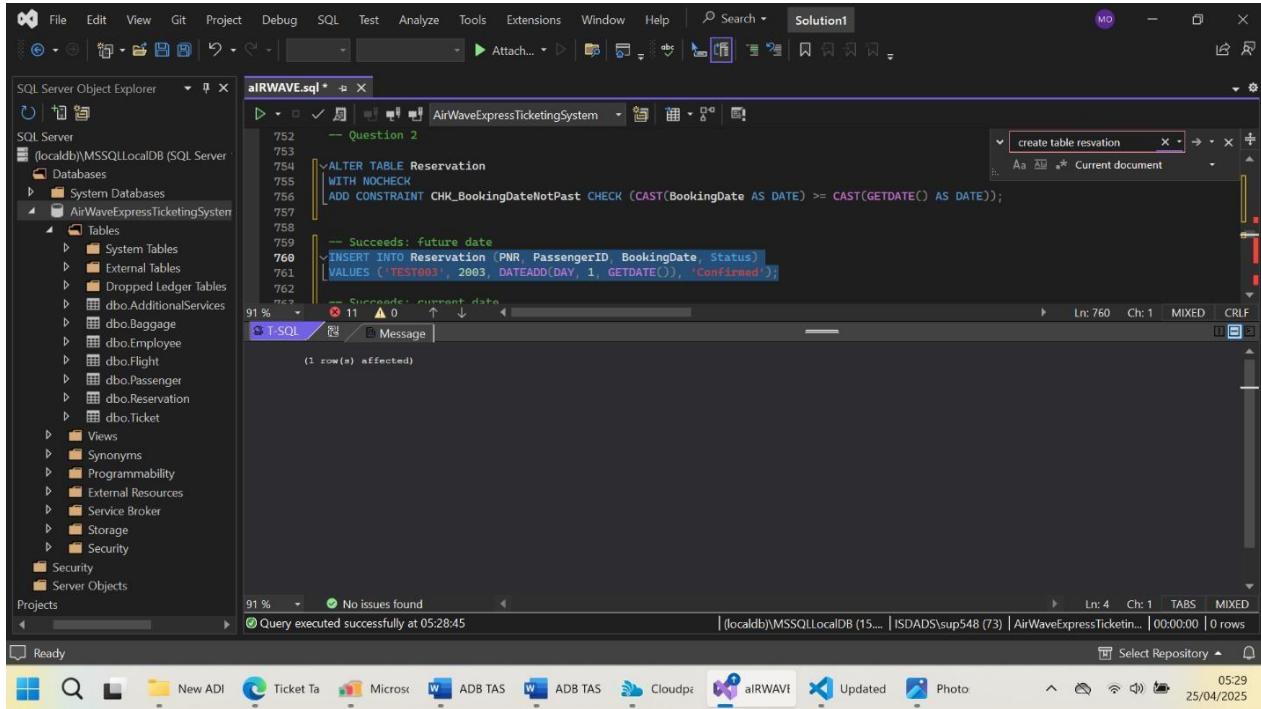
- Object Explorer:** Shows the database structure under "AirWaveExpressTicketingSystem".
- Script Editor:** The current file is "aIRWAVE.sql". The code in the editor is:

```
752 -- Question 2
753
754 ALTER TABLE Reservation
755   WITH NOCHECK
756   ADD CONSTRAINT CHK_BookingDateNotPast CHECK (CAST(BookingDate AS DATE) >= CAST(GETDATE() AS DATE));
757
758
759 -- Succeeds: future date
760 INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
761 VALUES ('TEST003', 2003, DATEADD(DAY, 1, GETDATE()), 'Confirmed');
```

- Status Bar:** Shows "Query executed successfully at 05:28:45".
- Taskbar:** Includes icons for New ADI, Ticket Ta, Microsoft, ADB TAS, Cloudp..., aIRWAVE (selected), Updated, Photo, and system icons.

- *Testing the Constraint*

Let's verify the constraint works properly:



The screenshot shows the SQL Server Object Explorer on the left and a T-SQL query window on the right. The query window contains the following code:

```

ALTER TABLE Reservation
WITH NOCHECK
ADD CONSTRAINT CHK_BookingDateNotPast CHECK (CAST(BookingDate AS DATE) >= CAST(GETDATE() AS DATE));

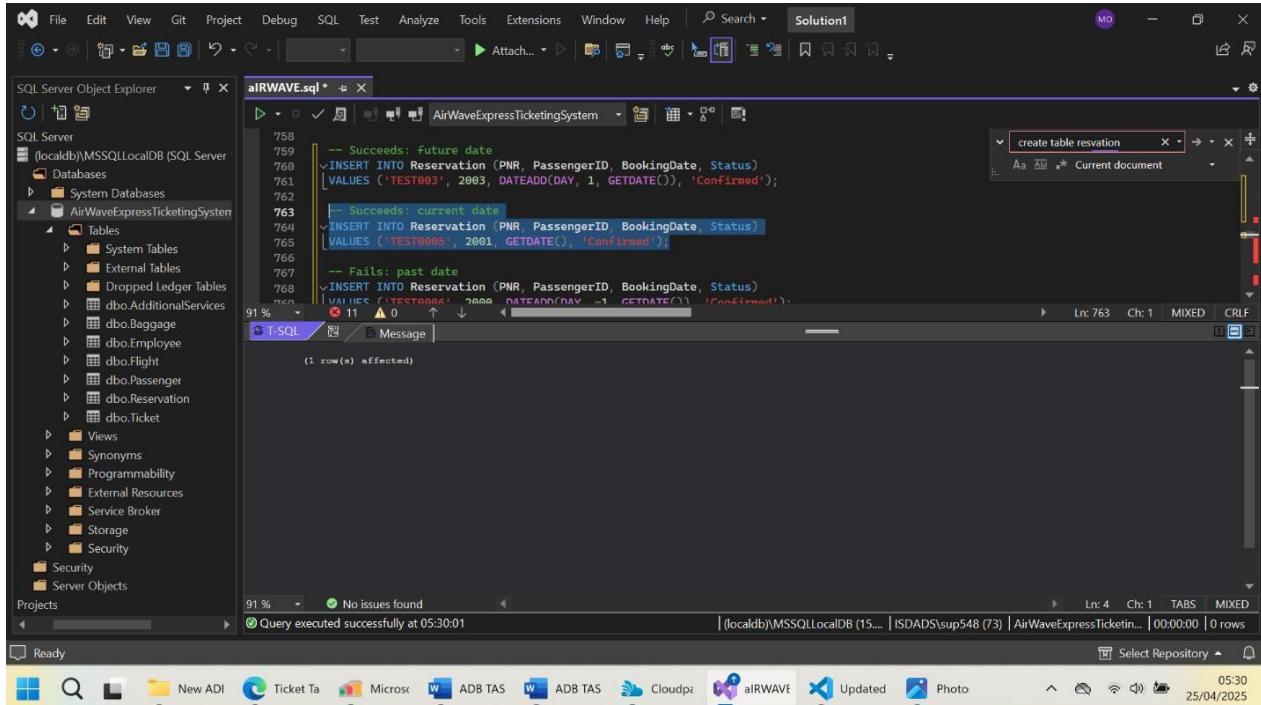
-- Succeeds: future date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST003', 2003, DATEADD(DAY, 1, GETDATE()), 'Confirmed');

-- Succeeds: current date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST005', 2001, GETDATE(), 'Confirmed');

```

The status bar at the bottom indicates "Query executed successfully at 05:28:45".

This is successful (future date)



The screenshot shows the SQL Server Object Explorer on the left and a T-SQL query window on the right. The query window contains the following code:

```

-- Succeeds: future date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST003', 2003, DATEADD(DAY, 1, GETDATE()), 'Confirmed');

-- Succeeds: current date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST005', 2001, GETDATE(), 'Confirmed');

-- Fails: past date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST006', 2000, DATEADD(DAY, -1, GETDATE()), 'Confirmed');

```

The status bar at the bottom indicates "Query executed successfully at 05:30:01".

This is successful (current date)

```

-- Succeeds: current date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST003', 2003, DATEADD(DAY, 1, GETDATE()), 'Confirmed');

-- Fails: past date
INSERT INTO Reservation (PNR, PassengerID, BookingDate, Status)
VALUES ('TEST005', 2001, GETDATE(), 'Confirmed');

-- The INSERT statement conflicted with the CHECK constraint "CHK_BookingDateNotPast".

```

Msg 547, Level 16, State 0, Line 768
The INSERT statement conflicted with the CHECK constraint "CHK_BookingDateNotPast". The conflict occurred in database "AirWaveExpressTicketingSystem", table "dbo.Reservation".
The statement has been terminated.

-- This fails (past date)

The third insert should fail with an error like:

The INSERT statement conflicted with the CHECK constraint "CHK_BookingDateNotPast".

3.2 Q3. Identifying Passengers with Pending Reservations and Age > 40 Years

I created a query to identify passengers who meet both criteria: having pending reservations and being over 40 years old.

- *Implementation*

```

--QUESTION 3 - create a query to identify passengers who meet both criteria: having pending reservations and being over 40 years old
WITH PassengerAge AS (
    SELECT
        p.PassengerID,
        p.FirstName + ' ' + p.LastName AS PassengerName,
        p.DateOfBirth,
        DATEDIFF(YEAR, p.DateOfBirth, GETDATE()) AS Age
    FROM
        Passenger p
)
SELECT
    pa.PassengerID,
    pa.PassengerName,
    pa.DateOfBirth,
    pa.Age,
    r.BookingDate,
    r.Status AS ReservationStatus
FROM
    PassengerAge pa
JOIN
    Reservation r ON pa.PassengerID = r.PassengerID
WHERE
    r.Status = 'Pending'
    AND pa.Age > 40;

```

	PassengerID	PassengerName	Email	DateOfBirth	Age	PNR	BookingDate	ReservationStatus
1	2006	Mwangi Kamau	mwangi.kamau@gmail.com	1980-01-05	45	PNR0007	2025-04-21 09:30:00.000	Pending
2	2014	Carlos Hernandez	carlos.hernandez@aol.com	1983-08-18	41	PNR0015	2025-04-22 12:15:00.000	Pending
3	2032	Diego Torres	diego.torres@aol.com	1983-10-10	41	PNR0033	2025-04-26 10:00:00.000	Pending
4	2024	Ivan Ivanov	ivan.ivanov@gmail.com	1984-02-28	41	PNR0025	2025-04-24 12:00:00.000	Pending

Query executed successfully at 05:32:24 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 4 rows

The result table from the query shows the passengers' name Mwangi Kamau, Carlos Hernandez, Diego Torres and Ivan Ivanov. Their birth years are 1980/01/05, 1983/08/18, 1983/10/10, 1984/02/28 respectively.

3.3 Q4. Stored Procedures and Functions for Ticketing System

Here are the implementations for the required database operations:

- *Search Passengers by Last Name*

The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. The 'Tables' node is expanded, showing various tables like Passenger, Reservation, Ticket, etc. In the center, the 'aiRWAVE.sql' editor window is open, showing the T-SQL code for creating a stored procedure:

```

817     CREATE PROCEDURE sp_SearchPassengersByLastName
818         @LastName NVARCHAR(50)
819     AS
820     BEGIN
821         SELECT
822             p.PassengerID,
823             p.FirstName,
824             p.LastName,
825             p.Email,
826             p.DateOfBirth,
827             r.TicketID,
828             r.IssueDate,
829             r.IssueTime,
830             f.FlightNumber,
831             f.DepartureTime,
832             f.Origin,
833             f.Destination
834         FROM
835             Passenger p
836         JOIN
837             Reservation r ON p.PassengerID = r.PassengerID
838         JOIN
839             Ticket t ON r.PNR = t.PNR
840         JOIN
841             Flight f ON t.FlightID = f.FlightID
842         WHERE
843             p.LastName LIKE '%' + @LastName + '%'
844         ORDER BY
845             r.IssueDate DESC, r.IssueTime DESC;
846     END
847     GO
848
849     --SAMPLE EXAMPLE FOR ABOVE PROCEDURE
850     EXEC sp_SearchPassengersByLastName @LastName = 'Sato';
851 
```

The 'Results' tab at the bottom shows the output of the query:

PassengerID	FirstName	LastName	Email	DateOfBirth	TicketID	IssueDate	IssueTime	FlightNumber	DepartureTime	Origin	Destination
2001	Yui	Sato	yui.sato@yahoo.com	1990-08-22	4089	2025-04-25	04:07:10.6300000	AW101	2025-05-02 09:00:00.000	Lisbon	New York JFK

A message at the bottom indicates: "Query executed successfully at 05:34:09".

Usage Example:

```
EXEC sp_SearchPassengersByLastName @LastName = 'Sato';
```

- *List Business Class Passengers with Meal Requirements (Today's Reservations)*

The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. The 'Tables' node is expanded, showing various tables like Passenger, Reservation, Ticket, etc. In the center, the 'aiRWAVE.sql' editor window is open, showing the T-SQL code for creating a stored procedure:

```

853     CREATE PROCEDURE sp_GetBusinessClassPassengersWithMealsToday
854     AS
855     BEGIN
856         SELECT
857             p.PassengerID,
858             p.FirstName + ' ' + p.LastName AS PassengerName,
859             r.PNR,
860             t.TicketID,
861             t.Class,
862             f.FlightNumber,
863             f.DepartureTime,
864             f.Origin,
865             f.Destination
866         FROM
867             Passenger p
868         JOIN
869             Reservation r ON p.PassengerID = r.PassengerID
870         JOIN
871             Ticket t ON r.PNR = t.PNR
872         JOIN
873             Flight f ON t.FlightID = f.FlightID
874         WHERE
875             CAST(r.BookingDate AS DATE) = CAST(GETDATE() AS DATE)
876             AND t.Class = 'Business'
877             AND r.MealPreference IS NOT NULL
878         ORDER BY
879             p.LastName, p.FirstName;
880     END
881     GO
882
883     --USAGE EXAMPLE
884     EXEC sp_GetBusinessClassPassengersWithMealsToday;
885
886     --QUESTION 4C - Insert New Employee
887 
```

The 'Results' tab at the bottom shows the output of the query:

PassengerID	PassengerName	MealPreference	PNR	TicketID	Class	FlightNumber	DepartureTime	Origin	Destination
2027	Sara Ali	Non-Vegetarian	PNR0028	4115	Business	AW103	2025-05-02 09:00:00.000	Lisbon	New York JFK

A message at the bottom indicates: "Query executed successfully at 05:36:13".

Usage Example:

```
EXEC sp_GetBusinessClassPassengersWithMealsToday;
```

- *Insert New Employee*

The screenshot shows the SSMS interface with the following details:

- Solution Explorer:** Shows the database structure for "AirWaveExpressTicketingSystem" including Tables, Views, Synonyms, Programmability, External Resources, and Security.
- Object Explorer:** Shows the same database structure as the Solution Explorer.
- Code Editor:** Displays the T-SQL code for the stored procedure `sp_InsertNewEmployee`. The code includes validation logic for roles, a TRY-CATCH block for inserting the employee, and a SELECT statement to return the inserted employee's details.
- Status Bar:** Shows the message "Query executed successfully at 05:37:44".
- Taskbar:** Shows various open tabs and icons, including "Ticket Tab", "Microso", "ADB TASK", "Cloudpac", "airWAVE.", "Updated_...", and "Photo:".

The screenshot shows the SSMS interface with the following details:

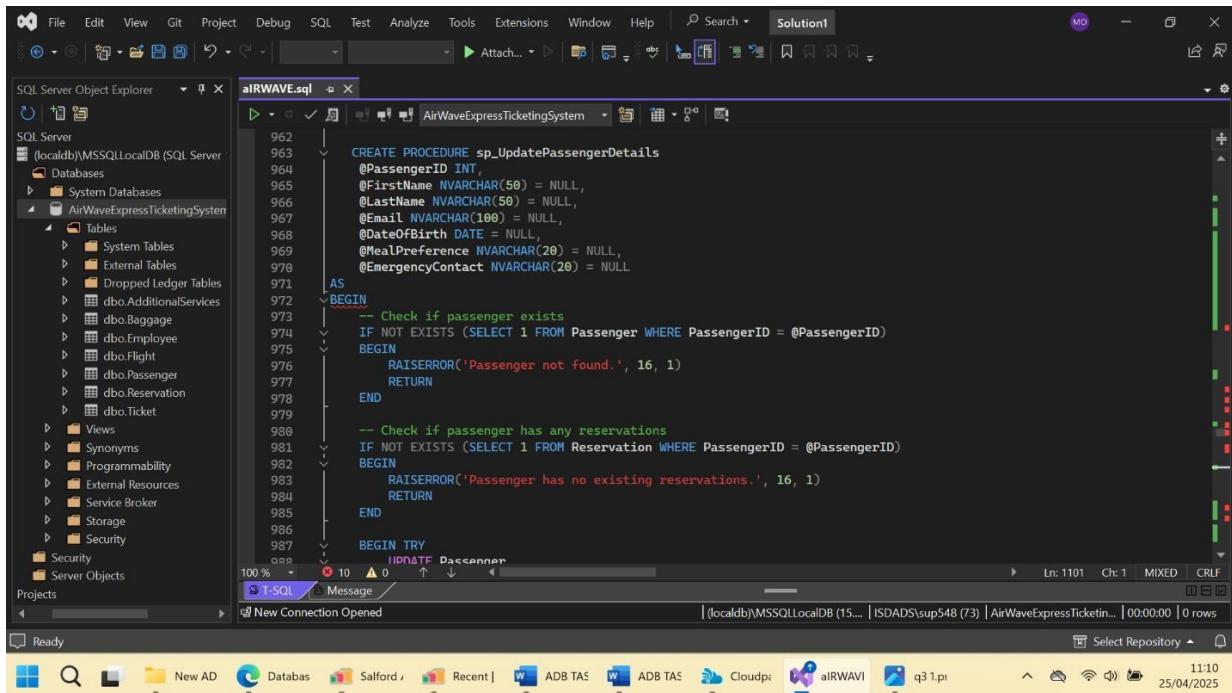
- Solution Explorer:** Shows the database structure for "AirWaveExpressTicketingSystem".
- Object Explorer:** Shows the same database structure as the Solution Explorer.
- Code Editor:** Displays the T-SQL code for the stored procedure `sp_InsertNewEmployee`, including the execution command and the resulting output.
- Results Grid:** Shows the output of the query, which is a table with columns: EmployeeID, Name, Username, Password, Email, Role, and LastLogin. The table contains 7 rows of data.
- Status Bar:** Shows the message "Query executed successfully at 05:37:44".
- Taskbar:** Shows various open tabs and icons, including "Ticket Tab", "Microso", "ADB TASK", "Cloudpac", "airWAVE.", "Updated_...", and "Photo:".

EmployeeID	Name	Username	Password	Email	Role	LastLogin
EM200	Chinedu Okeke	chinedu.okeke	securePass123	chinedu.okeke@airwave.com	Ticketing Staff	2025-04-23 08:30:00.000
EM201	Adabi Nwosu	adabi.nwosu	strongPass456	adabi.nwosu@airwave.com	Ticketing Supervisor	2025-04-23 09:15:00.000
EM202	Uchechukwu	uchechukwu	passWord321	uchechukwu.ezo@airwave	Ticketing Staff	2025-04-23 11:00:00.000
EM203	Nkachi Okafor	nkachiOkafor	passWord654	nkachiOkafor@airwave.com	Ticketing Supervisor	2025-04-23 11:30:00.000
EM204	Chukwudi Nnaji	chukwudinna	password987	chukwudi.nnaji@airwave.com	Ticketing Staff	2025-04-23 12:00:00.000
EM205	Ifeanyi Ugwu	ifeanyiUgwu	safePass789	ifeanyi.ugwu@airwave.com	Ticketing Staff	2025-04-23 10:00:00.000
EM206	Ochukwu C	ochukwu233	documentPass456	ochukwu.chukwu@airwave	Ticketing Staff	NULL

Usage Example:

```
EXEC sp_InsertNewEmployee
    @Username = 'carchuks223',
    @Password = 'securepass456',
    @Name = 'Christianar Chuksar',
    @Email = 'christianar.chuksar@airwave.com',
    @Role = 'Ticketing Staff';
```

- d) Update Passenger Details



```
CREATE PROCEDURE sp_UpdatePassengerDetails
    @PassengerID INT,
    @FirstName NVARCHAR(50) = NULL,
    @LastName NVARCHAR(50) = NULL,
    @Email NVARCHAR(100) = NULL,
    @DateOfBirth DATE = NULL,
    @MealPreference NVARCHAR(20) = NULL,
    @EmergencyContact NVARCHAR(20) = NULL
AS
BEGIN
    -- Check if passenger exists
    IF NOT EXISTS (SELECT 1 FROM Passenger WHERE PassengerID = @PassengerID)
    BEGIN
        RAISERROR('Passenger not found.', 16, 1)
        RETURN
    END
    -- Check if passenger has any reservations
    IF NOT EXISTS (SELECT 1 FROM Reservation WHERE PassengerID = @PassengerID)
    BEGIN
        RAISERROR('Passenger has no existing reservations.', 16, 1)
        RETURN
    END
    BEGIN TRY
        UPDATE Passenger
    END TRY
    BEGIN CATCH
        -- Handle errors
    END CATCH
END
```

```

986 BEGIN TRY
987     UPDATE Passenger
988     SET
989         FirstName = ISNULL(@FirstName, FirstName),
990         LastName = ISNULL(@LastName, LastName),
991         Email = ISNULL(@Email, Email),
992         DateOfBirth = ISNULL(@DateOfBirth, DateOfBirth),
993         MealPreference = @MealPreference, -- NULL is allowed
994         EmergencyContact = @EmergencyContact -- NULL is allowed
995     WHERE
996         PassengerID = @PassengerID;
997
998     SELECT
999         PassengerID,
1000        FirstName,
1001        LastName,
1002        Email,
1003        DateOfBirth,
1004        MealPreference,
1005        EmergencyContact,
1006        'Passenger details updated successfully. Reservation verified.' AS Message
1007     FROM
1008        Passenger
1009     WHERE
1010        PassengerID = @PassengerID;
1011
1012 END TRY
1013 BEGIN CATCH
1014     IF ERROR_NUMBER() = 2627 -- Unique constraint violation
1015     BEGIN
1016         RAISERROR('Email address already exists. Please use a different email.', 16, 1)
1017     END
1018     ELSE
1019     BEGIN
1020         DECLARE @ErrMsg NVARCHAR(4000) = ERROR_MESSAGE();
1021         RAISERROR(@ErrMsg, 16, 1);
1022     END
1023 END CATCH
1024
1025 GO
1026
1027 --USAGE EXAMPLE
1028 EXEC sp_UpdatePassengerDetails
1029     @PassengerID = 2005,
1030     @LastName = 'Santos',
1031     @MealPreference = 'Non-Vegetarian',
1032     @EmergencyContact = 'Rafael Santos';
1033
1034
1035
1036

```

USAGE EXAMPLE

EXEC sp_UpdatePassengerDetails

@PassengerID = 2005,
@LastName = 'Santos',
@MealPreference = 'Non-Vegetarian',

```
@EmergencyContact = 'Rafael Santos';
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The left pane displays the SQL Server Object Explorer, which includes the database structure for 'AirWaveExpressTicketingSystem'. The right pane shows a T-SQL script named 'aIRWAVE.sql' with line numbers 1015 through 1033. The script contains error handling logic for an email address already existing. Below the script, an 'EXEC' statement is shown with parameters: @PassengerID = 2065, @LastName = 'Santos', @MealPreference = 'Non-Vegetarian', and @EmergencyContact = 'Rafael Santos'. The results tab at the bottom shows a single row of data updated successfully.

```
1015      RAISERROR('Email address already exists. Please use a different email.', 16, 1)
1016      END
1017      ELSE
1018      BEGIN
1019          DECLARE @ErrMsg NVARCHAR(4000) = ERROR_MESSAGE();
1020          RAISERROR(@ErrMsg, 16, 1);
1021      END
1022  END CATCH
1023
1024 GO
1025
1026 --USAGE EXAMPLE
1027 EXEC sp_UpdatePassengerDetails
1028     @PassengerID = 2065,
1029     @LastName = 'Santos',
1030     @MealPreference = 'Non-Vegetarian',
1031     @EmergencyContact = 'Rafael Santos';
1032
1033
```

	PassengerID	FirstName	LastName	Email	DateOfBirth	MealPreference	EmergencyContact	Message
1	2065	Isabela	Santos	isabela.santos@proton.me	1992-04-25	Non-Vegetarian	Rafael Santos	Passenger details updated successfully. Reservat...

Query executed successfully at 05:46:05 | (localdb)\MSSQLLocalDB (15.... | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 1 rows

3.4 Q5. View for Employee-Issued E-Boarding Numbers with Revenue Details

I created a comprehensive view that shows all e-boarding numbers issued by a specific employee, along with the revenue details for each ticket, including base fare and additional service fees.

- *Implementation*

```

1037 --QUESTION 5 - View for Employee-Issued E-Boarding Numbers with Revenue Details
1038 CREATE VIEW vw_EmployeeTicketRevenue AS
1039
1040     SELECT
1041         e.EmployeeID,
1042         e.Name AS EmployeeName,
1043         e.Role,
1044         t.EBoardingNumber,
1045         t.IssueDate,
1046         t.IssueTime,
1047         p.PassengerID,
1048         p.FirstName + ' ' + p.LastName AS PassengerName,
1049         f.FlightID,
1050         f.FlightNumber,
1051         f.DepartureTime,
1052         f.Origin,
1053         f.Destination,
1054         t.Fare AS BaseFare,
1055         ISNULL(a.ExtraBaggageKG * 100, 0) AS ExtraBaggageKG,
1056         ISNULL(a.ExtraBaggageKG * 100, 0) AS BaggageFee,
1057         CASE WHEN a.UpgradeMeal = 1 THEN 20 ELSE 0 END AS MealUpgradeFee,
1058         CASE WHEN a.PreferredSeat = 1 THEN 30 ELSE 0 END AS PreferredSeatFee,
1059         ISNULL(a.ServiceFee, 0) AS TotalAdditionalFees,
1060         t.Fare + ISNULL(a.ServiceFee, 0) AS TotalRevenue,
1061         t.Class,
1062         t.SeatNumber
1063     FROM
1064         Employee e
1065     JOIN
1066         Ticket t ON e.EmployeeID = t.IssuedBy
1067     JOIN
1068         Reservation r ON t.PNR = r.PNR
1069     JOIN
1070         Passenger p ON r.PassengerID = p.PassengerID
1071     JOIN
1072         Flight f ON t.FlightID = f.FlightID
1073     LEFT JOIN
1074         AdditionalServices a ON t.TicketID = a.TicketID;
    
```

New Connection Opened | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 0 rows

```

1061     t.SeatNumber
1062
1063     FROM
1064         Employee e
1065     JOIN
1066         Ticket t ON e.EmployeeID = t.IssuedBy
1067     JOIN
1068         Reservation r ON t.PNR = r.PNR
1069     JOIN
1070         Passenger p ON r.PassengerID = p.PassengerID
1071     JOIN
1072         Flight f ON t.FlightID = f.FlightID
1073     LEFT JOIN
1074         AdditionalServices a ON t.TicketID = a.TicketID;
    
```

Usage Examples

- View all tickets issued by a specific employee (EmployeeID: EM201):

The screenshot shows the SSMS interface with the 'airWAVE.sql' file open. The code is a query that joins the 'Passenger' and 'Flight' tables, then left joins the 'AdditionalServices' table. It includes a comment for usage examples and a select statement to find all tickets issued by employee ID 1000, ordered by issue date and time.

```

1070   JOIN
1071     Flight f ON t.FlightID = f.FlightID
1072   LEFT JOIN
1073     AdditionalServices a ON t.TicketID = a.TicketID;
1074
1075 --Usage Examples - View all tickets issued by a specific employee (EmployeeID 1000):
1076 SELECT * FROM vw_EmployeeTicketRevenue
1077 WHERE EmployeeID = 'EM201'
1078 ORDER BY IssueDate DESC, IssueTime DESC;

```

The results grid displays 7 rows of data, showing ticket details for employee EM201 across different flights and dates.

	EmployeeID	EmployeeName	Role	EBoardingNumber	IssueDate	IssueTime	PassengerID	PassengerName	FlightID	FlightNumber	DepartureTime	Origin
1	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0030	2025-04-25	04:07:10.6333333	2029	Giulia Bianchi	3001	AW101	2025-05-02 09:00:00.000	Lisbon
2	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0036	2025-04-25	04:07:10.6333333	2035	Sophia Davis	3003	AW103	2025-05-02 09:00:00.000	Lisbon
3	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0042	2025-04-25	04:07:10.6333333	2041	Ren Kobayashi	3001	AW101	2025-05-02 09:00:00.000	Lisbon
4	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0006	2025-04-25	04:07:10.6300000	2005	Isabela Santos	3001	AW101	2025-05-02 09:00:00.000	Lisbon
5	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0012	2025-04-25	04:07:10.6300000	2011	Ava Williams	3003	AW103	2025-05-02 09:00:00.000	Lisbon
6	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0018	2025-04-25	04:07:10.6300000	2017	Emma Larsen	3001	AW101	2025-05-02 09:00:00.000	Lisbon
7	EM201	Adaobi Nwosu	Ticketing Supervisor	EBN0024	2025-04-25	04:07:10.6300000	2023	Amelia Taylor	3003	AW103	2025-05-02 09:00:00.000	Lisbon

Query executed successfully at 05:47:57 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 7 rows

- Calculate total revenue generated by an employee on a specific flight:

The screenshot shows the SSMS interface with the 'airWAVE.sql' file open. The code is a query that selects employee ID, name, flight number, total tickets issued, and total revenue generated, filtering for employee EM201 and flight AW101. The results grid shows 1 row of data.

```

1081
1082   SELECT
1083     EmployeeID,
1084     EmployeeName,
1085     FlightNumber,
1086     COUNT(EBoardingNumber) AS TicketsIssued,
1087     SUM(TotalRevenue) AS TotalRevenueGenerated
1088   FROM
1089     vw_EmployeeTicketRevenue
1090   WHERE
1091     EmployeeID = 'EM201'
1092     AND FlightNumber = 'AW101'
1093   GROUP BY
1094     EmployeeID, EmployeeName, FlightNumber

```

	EmployeeID	EmployeeName	FlightNumber	TicketsIssued	TotalRevenueGenerated
1	EM201	Adaobi Nwosu	AW101	4	1626.70

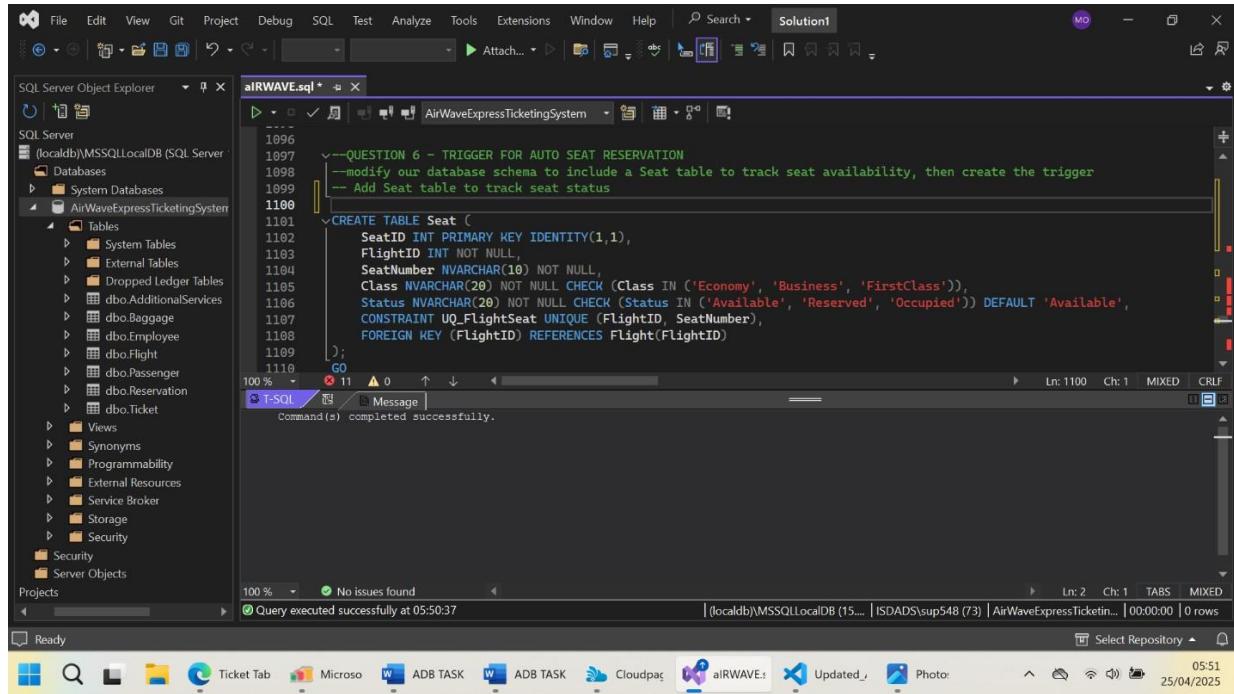
Query executed successfully at 05:49:45 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 1 rows

3.5 Q6. Trigger for Automatic Seat Reservation Update

I created a trigger that automatically updates the seat status to "Reserved" when a ticket is issued with a seat assignment. This ensures seat allocations are properly tracked in real time.

Implementation

First, we need to modify our database schema to include a Seat table to track seat availability, then create the trigger:



The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. The 'Tables' node is expanded, showing various tables like Seats, Tickets, Passengers, etc. In the center, the 'airWAVE.sql' script editor window is open, showing T-SQL code for creating a 'Seat' table. The code includes columns for SeatID (primary key), FlightID (not null), SeatNumber (not null), Class (check constraint), and Status (check constraint). A unique constraint 'UQ_FlightSeat' is defined on the combination of FlightID and SeatNumber. A foreign key constraint links FlightID to the 'Flight' table. The 'GO' command is present at the end of the script. Below the editor, the message 'Command(s) completed successfully.' is displayed. At the bottom, the status bar shows the command was executed at 05:50:37 and 0 rows were affected.

```
1096  
1097    --QUESTION 6 - TRIGGER FOR AUTO SEAT RESERVATION  
1098    --modify our database schema to include a Seat table to track seat availability, then create the trigger  
1099    --- Add Seat table to track seat status  
1100  
1101    CREATE TABLE Seat (  
1102        SeatID INT PRIMARY KEY IDENTITY(1,1),  
1103        FlightID INT NOT NULL,  
1104        SeatNumber NVARCHAR(10) NOT NULL,  
1105        Class NVARCHAR(20) NOT NULL CHECK (Class IN ('Economy', 'Business', 'FirstClass')),  
1106        Status NVARCHAR(20) NOT NULL CHECK (Status IN ('Available', 'Reserved', 'Occupied')) DEFAULT 'Available',  
1107        CONSTRAINT UQ_FlightSeat UNIQUE (FlightID, SeatNumber),  
1108        FOREIGN KEY (FlightID) REFERENCES Flight(FlightID)  
1109    );  
1110    GO  
1111  
1112    --QUESTION 7 - TRIGGER FOR AUTOMATIC SEAT RESERVATION  
1113    --create a trigger that automatically updates the seat status to "Reserved" when a ticket is issued with a seat assignment.  
1114    --This ensures seat allocations are properly tracked in real time.  
1115  
1116    --CREATE TRIGGER AutoReserve  
1117    --ON Ticket  
1118    --FOR INSERT  
1119    --AS  
1120    --BEGIN  
1121    --    UPDATE Seat  
1122    --    SET Status = 'Reserved'  
1123    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1124    --END  
1125  
1126    --ALTER TRIGGER AutoReserve  
1127    --ON Ticket  
1128    --FOR INSERT  
1129    --AS  
1130    --BEGIN  
1131    --    UPDATE Seat  
1132    --    SET Status = 'Reserved'  
1133    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1134    --END  
1135  
1136    --ALTER TRIGGER AutoReserve  
1137    --ON Ticket  
1138    --FOR UPDATE  
1139    --AS  
1140    --BEGIN  
1141    --    UPDATE Seat  
1142    --    SET Status = 'Reserved'  
1143    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1144    --END  
1145  
1146    --ALTER TRIGGER AutoReserve  
1147    --ON Ticket  
1148    --FOR UPDATE  
1149    --AS  
1150    --BEGIN  
1151    --    UPDATE Seat  
1152    --    SET Status = 'Available'  
1153    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1154    --END  
1155  
1156    --ALTER TRIGGER AutoReserve  
1157    --ON Ticket  
1158    --FOR UPDATE  
1159    --AS  
1160    --BEGIN  
1161    --    UPDATE Seat  
1162    --    SET Status = 'Occupied'  
1163    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1164    --END  
1165  
1166    --ALTER TRIGGER AutoReserve  
1167    --ON Ticket  
1168    --FOR UPDATE  
1169    --AS  
1170    --BEGIN  
1171    --    UPDATE Seat  
1172    --    SET Status = 'Available'  
1173    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1174    --END  
1175  
1176    --ALTER TRIGGER AutoReserve  
1177    --ON Ticket  
1178    --FOR UPDATE  
1179    --AS  
1180    --BEGIN  
1181    --    UPDATE Seat  
1182    --    SET Status = 'Occupied'  
1183    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1184    --END  
1185  
1186    --ALTER TRIGGER AutoReserve  
1187    --ON Ticket  
1188    --FOR UPDATE  
1189    --AS  
1190    --BEGIN  
1191    --    UPDATE Seat  
1192    --    SET Status = 'Available'  
1193    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1194    --END  
1195  
1196    --ALTER TRIGGER AutoReserve  
1197    --ON Ticket  
1198    --FOR UPDATE  
1199    --AS  
1200    --BEGIN  
1201    --    UPDATE Seat  
1202    --    SET Status = 'Occupied'  
1203    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1204    --END  
1205  
1206    --ALTER TRIGGER AutoReserve  
1207    --ON Ticket  
1208    --FOR UPDATE  
1209    --AS  
1210    --BEGIN  
1211    --    UPDATE Seat  
1212    --    SET Status = 'Available'  
1213    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1214    --END  
1215  
1216    --ALTER TRIGGER AutoReserve  
1217    --ON Ticket  
1218    --FOR UPDATE  
1219    --AS  
1220    --BEGIN  
1221    --    UPDATE Seat  
1222    --    SET Status = 'Occupied'  
1223    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1224    --END  
1225  
1226    --ALTER TRIGGER AutoReserve  
1227    --ON Ticket  
1228    --FOR UPDATE  
1229    --AS  
1230    --BEGIN  
1231    --    UPDATE Seat  
1232    --    SET Status = 'Available'  
1233    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1234    --END  
1235  
1236    --ALTER TRIGGER AutoReserve  
1237    --ON Ticket  
1238    --FOR UPDATE  
1239    --AS  
1240    --BEGIN  
1241    --    UPDATE Seat  
1242    --    SET Status = 'Occupied'  
1243    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1244    --END  
1245  
1246    --ALTER TRIGGER AutoReserve  
1247    --ON Ticket  
1248    --FOR UPDATE  
1249    --AS  
1250    --BEGIN  
1251    --    UPDATE Seat  
1252    --    SET Status = 'Available'  
1253    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1254    --END  
1255  
1256    --ALTER TRIGGER AutoReserve  
1257    --ON Ticket  
1258    --FOR UPDATE  
1259    --AS  
1260    --BEGIN  
1261    --    UPDATE Seat  
1262    --    SET Status = 'Occupied'  
1263    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1264    --END  
1265  
1266    --ALTER TRIGGER AutoReserve  
1267    --ON Ticket  
1268    --FOR UPDATE  
1269    --AS  
1270    --BEGIN  
1271    --    UPDATE Seat  
1272    --    SET Status = 'Available'  
1273    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1274    --END  
1275  
1276    --ALTER TRIGGER AutoReserve  
1277    --ON Ticket  
1278    --FOR UPDATE  
1279    --AS  
1280    --BEGIN  
1281    --    UPDATE Seat  
1282    --    SET Status = 'Occupied'  
1283    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1284    --END  
1285  
1286    --ALTER TRIGGER AutoReserve  
1287    --ON Ticket  
1288    --FOR UPDATE  
1289    --AS  
1290    --BEGIN  
1291    --    UPDATE Seat  
1292    --    SET Status = 'Available'  
1293    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1294    --END  
1295  
1296    --ALTER TRIGGER AutoReserve  
1297    --ON Ticket  
1298    --FOR UPDATE  
1299    --AS  
1300    --BEGIN  
1301    --    UPDATE Seat  
1302    --    SET Status = 'Occupied'  
1303    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1304    --END  
1305  
1306    --ALTER TRIGGER AutoReserve  
1307    --ON Ticket  
1308    --FOR UPDATE  
1309    --AS  
1310    --BEGIN  
1311    --    UPDATE Seat  
1312    --    SET Status = 'Available'  
1313    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1314    --END  
1315  
1316    --ALTER TRIGGER AutoReserve  
1317    --ON Ticket  
1318    --FOR UPDATE  
1319    --AS  
1320    --BEGIN  
1321    --    UPDATE Seat  
1322    --    SET Status = 'Occupied'  
1323    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1324    --END  
1325  
1326    --ALTER TRIGGER AutoReserve  
1327    --ON Ticket  
1328    --FOR UPDATE  
1329    --AS  
1330    --BEGIN  
1331    --    UPDATE Seat  
1332    --    SET Status = 'Available'  
1333    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1334    --END  
1335  
1336    --ALTER TRIGGER AutoReserve  
1337    --ON Ticket  
1338    --FOR UPDATE  
1339    --AS  
1340    --BEGIN  
1341    --    UPDATE Seat  
1342    --    SET Status = 'Occupied'  
1343    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1344    --END  
1345  
1346    --ALTER TRIGGER AutoReserve  
1347    --ON Ticket  
1348    --FOR UPDATE  
1349    --AS  
1350    --BEGIN  
1351    --    UPDATE Seat  
1352    --    SET Status = 'Available'  
1353    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1354    --END  
1355  
1356    --ALTER TRIGGER AutoReserve  
1357    --ON Ticket  
1358    --FOR UPDATE  
1359    --AS  
1360    --BEGIN  
1361    --    UPDATE Seat  
1362    --    SET Status = 'Occupied'  
1363    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1364    --END  
1365  
1366    --ALTER TRIGGER AutoReserve  
1367    --ON Ticket  
1368    --FOR UPDATE  
1369    --AS  
1370    --BEGIN  
1371    --    UPDATE Seat  
1372    --    SET Status = 'Available'  
1373    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1374    --END  
1375  
1376    --ALTER TRIGGER AutoReserve  
1377    --ON Ticket  
1378    --FOR UPDATE  
1379    --AS  
1380    --BEGIN  
1381    --    UPDATE Seat  
1382    --    SET Status = 'Occupied'  
1383    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1384    --END  
1385  
1386    --ALTER TRIGGER AutoReserve  
1387    --ON Ticket  
1388    --FOR UPDATE  
1389    --AS  
1390    --BEGIN  
1391    --    UPDATE Seat  
1392    --    SET Status = 'Available'  
1393    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1394    --END  
1395  
1396    --ALTER TRIGGER AutoReserve  
1397    --ON Ticket  
1398    --FOR UPDATE  
1399    --AS  
1400    --BEGIN  
1401    --    UPDATE Seat  
1402    --    SET Status = 'Occupied'  
1403    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1404    --END  
1405  
1406    --ALTER TRIGGER AutoReserve  
1407    --ON Ticket  
1408    --FOR UPDATE  
1409    --AS  
1410    --BEGIN  
1411    --    UPDATE Seat  
1412    --    SET Status = 'Available'  
1413    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1414    --END  
1415  
1416    --ALTER TRIGGER AutoReserve  
1417    --ON Ticket  
1418    --FOR UPDATE  
1419    --AS  
1420    --BEGIN  
1421    --    UPDATE Seat  
1422    --    SET Status = 'Occupied'  
1423    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1424    --END  
1425  
1426    --ALTER TRIGGER AutoReserve  
1427    --ON Ticket  
1428    --FOR UPDATE  
1429    --AS  
1430    --BEGIN  
1431    --    UPDATE Seat  
1432    --    SET Status = 'Available'  
1433    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1434    --END  
1435  
1436    --ALTER TRIGGER AutoReserve  
1437    --ON Ticket  
1438    --FOR UPDATE  
1439    --AS  
1440    --BEGIN  
1441    --    UPDATE Seat  
1442    --    SET Status = 'Occupied'  
1443    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1444    --END  
1445  
1446    --ALTER TRIGGER AutoReserve  
1447    --ON Ticket  
1448    --FOR UPDATE  
1449    --AS  
1450    --BEGIN  
1451    --    UPDATE Seat  
1452    --    SET Status = 'Available'  
1453    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1454    --END  
1455  
1456    --ALTER TRIGGER AutoReserve  
1457    --ON Ticket  
1458    --FOR UPDATE  
1459    --AS  
1460    --BEGIN  
1461    --    UPDATE Seat  
1462    --    SET Status = 'Occupied'  
1463    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1464    --END  
1465  
1466    --ALTER TRIGGER AutoReserve  
1467    --ON Ticket  
1468    --FOR UPDATE  
1469    --AS  
1470    --BEGIN  
1471    --    UPDATE Seat  
1472    --    SET Status = 'Available'  
1473    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1474    --END  
1475  
1476    --ALTER TRIGGER AutoReserve  
1477    --ON Ticket  
1478    --FOR UPDATE  
1479    --AS  
1480    --BEGIN  
1481    --    UPDATE Seat  
1482    --    SET Status = 'Occupied'  
1483    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1484    --END  
1485  
1486    --ALTER TRIGGER AutoReserve  
1487    --ON Ticket  
1488    --FOR UPDATE  
1489    --AS  
1490    --BEGIN  
1491    --    UPDATE Seat  
1492    --    SET Status = 'Available'  
1493    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1494    --END  
1495  
1496    --ALTER TRIGGER AutoReserve  
1497    --ON Ticket  
1498    --FOR UPDATE  
1499    --AS  
1500    --BEGIN  
1501    --    UPDATE Seat  
1502    --    SET Status = 'Occupied'  
1503    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1504    --END  
1505  
1506    --ALTER TRIGGER AutoReserve  
1507    --ON Ticket  
1508    --FOR UPDATE  
1509    --AS  
1510    --BEGIN  
1511    --    UPDATE Seat  
1512    --    SET Status = 'Available'  
1513    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1514    --END  
1515  
1516    --ALTER TRIGGER AutoReserve  
1517    --ON Ticket  
1518    --FOR UPDATE  
1519    --AS  
1520    --BEGIN  
1521    --    UPDATE Seat  
1522    --    SET Status = 'Occupied'  
1523    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1524    --END  
1525  
1526    --ALTER TRIGGER AutoReserve  
1527    --ON Ticket  
1528    --FOR UPDATE  
1529    --AS  
1530    --BEGIN  
1531    --    UPDATE Seat  
1532    --    SET Status = 'Available'  
1533    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1534    --END  
1535  
1536    --ALTER TRIGGER AutoReserve  
1537    --ON Ticket  
1538    --FOR UPDATE  
1539    --AS  
1540    --BEGIN  
1541    --    UPDATE Seat  
1542    --    SET Status = 'Occupied'  
1543    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1544    --END  
1545  
1546    --ALTER TRIGGER AutoReserve  
1547    --ON Ticket  
1548    --FOR UPDATE  
1549    --AS  
1550    --BEGIN  
1551    --    UPDATE Seat  
1552    --    SET Status = 'Available'  
1553    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1554    --END  
1555  
1556    --ALTER TRIGGER AutoReserve  
1557    --ON Ticket  
1558    --FOR UPDATE  
1559    --AS  
1560    --BEGIN  
1561    --    UPDATE Seat  
1562    --    SET Status = 'Occupied'  
1563    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1564    --END  
1565  
1566    --ALTER TRIGGER AutoReserve  
1567    --ON Ticket  
1568    --FOR UPDATE  
1569    --AS  
1570    --BEGIN  
1571    --    UPDATE Seat  
1572    --    SET Status = 'Available'  
1573    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1574    --END  
1575  
1576    --ALTER TRIGGER AutoReserve  
1577    --ON Ticket  
1578    --FOR UPDATE  
1579    --AS  
1580    --BEGIN  
1581    --    UPDATE Seat  
1582    --    SET Status = 'Occupied'  
1583    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1584    --END  
1585  
1586    --ALTER TRIGGER AutoReserve  
1587    --ON Ticket  
1588    --FOR UPDATE  
1589    --AS  
1590    --BEGIN  
1591    --    UPDATE Seat  
1592    --    SET Status = 'Available'  
1593    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1594    --END  
1595  
1596    --ALTER TRIGGER AutoReserve  
1597    --ON Ticket  
1598    --FOR UPDATE  
1599    --AS  
1600    --BEGIN  
1601    --    UPDATE Seat  
1602    --    SET Status = 'Occupied'  
1603    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1604    --END  
1605  
1606    --ALTER TRIGGER AutoReserve  
1607    --ON Ticket  
1608    --FOR UPDATE  
1609    --AS  
1610    --BEGIN  
1611    --    UPDATE Seat  
1612    --    SET Status = 'Available'  
1613    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1614    --END  
1615  
1616    --ALTER TRIGGER AutoReserve  
1617    --ON Ticket  
1618    --FOR UPDATE  
1619    --AS  
1620    --BEGIN  
1621    --    UPDATE Seat  
1622    --    SET Status = 'Occupied'  
1623    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1624    --END  
1625  
1626    --ALTER TRIGGER AutoReserve  
1627    --ON Ticket  
1628    --FOR UPDATE  
1629    --AS  
1630    --BEGIN  
1631    --    UPDATE Seat  
1632    --    SET Status = 'Available'  
1633    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1634    --END  
1635  
1636    --ALTER TRIGGER AutoReserve  
1637    --ON Ticket  
1638    --FOR UPDATE  
1639    --AS  
1640    --BEGIN  
1641    --    UPDATE Seat  
1642    --    SET Status = 'Occupied'  
1643    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1644    --END  
1645  
1646    --ALTER TRIGGER AutoReserve  
1647    --ON Ticket  
1648    --FOR UPDATE  
1649    --AS  
1650    --BEGIN  
1651    --    UPDATE Seat  
1652    --    SET Status = 'Available'  
1653    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1654    --END  
1655  
1656    --ALTER TRIGGER AutoReserve  
1657    --ON Ticket  
1658    --FOR UPDATE  
1659    --AS  
1660    --BEGIN  
1661    --    UPDATE Seat  
1662    --    SET Status = 'Occupied'  
1663    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1664    --END  
1665  
1666    --ALTER TRIGGER AutoReserve  
1667    --ON Ticket  
1668    --FOR UPDATE  
1669    --AS  
1670    --BEGIN  
1671    --    UPDATE Seat  
1672    --    SET Status = 'Available'  
1673    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1674    --END  
1675  
1676    --ALTER TRIGGER AutoReserve  
1677    --ON Ticket  
1678    --FOR UPDATE  
1679    --AS  
1680    --BEGIN  
1681    --    UPDATE Seat  
1682    --    SET Status = 'Occupied'  
1683    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1684    --END  
1685  
1686    --ALTER TRIGGER AutoReserve  
1687    --ON Ticket  
1688    --FOR UPDATE  
1689    --AS  
1690    --BEGIN  
1691    --    UPDATE Seat  
1692    --    SET Status = 'Available'  
1693    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1694    --END  
1695  
1696    --ALTER TRIGGER AutoReserve  
1697    --ON Ticket  
1698    --FOR UPDATE  
1699    --AS  
1700    --BEGIN  
1701    --    UPDATE Seat  
1702    --    SET Status = 'Occupied'  
1703    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1704    --END  
1705  
1706    --ALTER TRIGGER AutoReserve  
1707    --ON Ticket  
1708    --FOR UPDATE  
1709    --AS  
1710    --BEGIN  
1711    --    UPDATE Seat  
1712    --    SET Status = 'Available'  
1713    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1714    --END  
1715  
1716    --ALTER TRIGGER AutoReserve  
1717    --ON Ticket  
1718    --FOR UPDATE  
1719    --AS  
1720    --BEGIN  
1721    --    UPDATE Seat  
1722    --    SET Status = 'Occupied'  
1723    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1724    --END  
1725  
1726    --ALTER TRIGGER AutoReserve  
1727    --ON Ticket  
1728    --FOR UPDATE  
1729    --AS  
1730    --BEGIN  
1731    --    UPDATE Seat  
1732    --    SET Status = 'Available'  
1733    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1734    --END  
1735  
1736    --ALTER TRIGGER AutoReserve  
1737    --ON Ticket  
1738    --FOR UPDATE  
1739    --AS  
1740    --BEGIN  
1741    --    UPDATE Seat  
1742    --    SET Status = 'Occupied'  
1743    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1744    --END  
1745  
1746    --ALTER TRIGGER AutoReserve  
1747    --ON Ticket  
1748    --FOR UPDATE  
1749    --AS  
1750    --BEGIN  
1751    --    UPDATE Seat  
1752    --    SET Status = 'Available'  
1753    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1754    --END  
1755  
1756    --ALTER TRIGGER AutoReserve  
1757    --ON Ticket  
1758    --FOR UPDATE  
1759    --AS  
1760    --BEGIN  
1761    --    UPDATE Seat  
1762    --    SET Status = 'Occupied'  
1763    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1764    --END  
1765  
1766    --ALTER TRIGGER AutoReserve  
1767    --ON Ticket  
1768    --FOR UPDATE  
1769    --AS  
1770    --BEGIN  
1771    --    UPDATE Seat  
1772    --    SET Status = 'Available'  
1773    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1774    --END  
1775  
1776    --ALTER TRIGGER AutoReserve  
1777    --ON Ticket  
1778    --FOR UPDATE  
1779    --AS  
1780    --BEGIN  
1781    --    UPDATE Seat  
1782    --    SET Status = 'Occupied'  
1783    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1784    --END  
1785  
1786    --ALTER TRIGGER AutoReserve  
1787    --ON Ticket  
1788    --FOR UPDATE  
1789    --AS  
1790    --BEGIN  
1791    --    UPDATE Seat  
1792    --    SET Status = 'Available'  
1793    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1794    --END  
1795  
1796    --ALTER TRIGGER AutoReserve  
1797    --ON Ticket  
1798    --FOR UPDATE  
1799    --AS  
1800    --BEGIN  
1801    --    UPDATE Seat  
1802    --    SET Status = 'Occupied'  
1803    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1804    --END  
1805  
1806    --ALTER TRIGGER AutoReserve  
1807    --ON Ticket  
1808    --FOR UPDATE  
1809    --AS  
1810    --BEGIN  
1811    --    UPDATE Seat  
1812    --    SET Status = 'Available'  
1813    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1814    --END  
1815  
1816    --ALTER TRIGGER AutoReserve  
1817    --ON Ticket  
1818    --FOR UPDATE  
1819    --AS  
1820    --BEGIN  
1821    --    UPDATE Seat  
1822    --    SET Status = 'Occupied'  
1823    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1824    --END  
1825  
1826    --ALTER TRIGGER AutoReserve  
1827    --ON Ticket  
1828    --FOR UPDATE  
1829    --AS  
1830    --BEGIN  
1831    --    UPDATE Seat  
1832    --    SET Status = 'Available'  
1833    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1834    --END  
1835  
1836    --ALTER TRIGGER AutoReserve  
1837    --ON Ticket  
1838    --FOR UPDATE  
1839    --AS  
1840    --BEGIN  
1841    --    UPDATE Seat  
1842    --    SET Status = 'Occupied'  
1843    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1844    --END  
1845  
1846    --ALTER TRIGGER AutoReserve  
1847    --ON Ticket  
1848    --FOR UPDATE  
1849    --AS  
1850    --BEGIN  
1851    --    UPDATE Seat  
1852    --    SET Status = 'Available'  
1853    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1854    --END  
1855  
1856    --ALTER TRIGGER AutoReserve  
1857    --ON Ticket  
1858    --FOR UPDATE  
1859    --AS  
1860    --BEGIN  
1861    --    UPDATE Seat  
1862    --    SET Status = 'Occupied'  
1863    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1864    --END  
1865  
1866    --ALTER TRIGGER AutoReserve  
1867    --ON Ticket  
1868    --FOR UPDATE  
1869    --AS  
1870    --BEGIN  
1871    --    UPDATE Seat  
1872    --    SET Status = 'Available'  
1873    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1874    --END  
1875  
1876    --ALTER TRIGGER AutoReserve  
1877    --ON Ticket  
1878    --FOR UPDATE  
1879    --AS  
1880    --BEGIN  
1881    --    UPDATE Seat  
1882    --    SET Status = 'Occupied'  
1883    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1884    --END  
1885  
1886    --ALTER TRIGGER AutoReserve  
1887    --ON Ticket  
1888    --FOR UPDATE  
1889    --AS  
1890    --BEGIN  
1891    --    UPDATE Seat  
1892    --    SET Status = 'Available'  
1893    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1894    --END  
1895  
1896    --ALTER TRIGGER AutoReserve  
1897    --ON Ticket  
1898    --FOR UPDATE  
1899    --AS  
1900    --BEGIN  
1901    --    UPDATE Seat  
1902    --    SET Status = 'Occupied'  
1903    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1904    --END  
1905  
1906    --ALTER TRIGGER AutoReserve  
1907    --ON Ticket  
1908    --FOR UPDATE  
1909    --AS  
1910    --BEGIN  
1911    --    UPDATE Seat  
1912    --    SET Status = 'Available'  
1913    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1914    --END  
1915  
1916    --ALTER TRIGGER AutoReserve  
1917    --ON Ticket  
1918    --FOR UPDATE  
1919    --AS  
1920    --BEGIN  
1921    --    UPDATE Seat  
1922    --    SET Status = 'Occupied'  
1923    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1924    --END  
1925  
1926    --ALTER TRIGGER AutoReserve  
1927    --ON Ticket  
1928    --FOR UPDATE  
1929    --AS  
1930    --BEGIN  
1931    --    UPDATE Seat  
1932    --    SET Status = 'Available'  
1933    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1934    --END  
1935  
1936    --ALTER TRIGGER AutoReserve  
1937    --ON Ticket  
1938    --FOR UPDATE  
1939    --AS  
1940    --BEGIN  
1941    --    UPDATE Seat  
1942    --    SET Status = 'Occupied'  
1943    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1944    --END  
1945  
1946    --ALTER TRIGGER AutoReserve  
1947    --ON Ticket  
1948    --FOR UPDATE  
1949    --AS  
1950    --BEGIN  
1951    --    UPDATE Seat  
1952    --    SET Status = 'Available'  
1953    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1954    --END  
1955  
1956    --ALTER TRIGGER AutoReserve  
1957    --ON Ticket  
1958    --FOR UPDATE  
1959    --AS  
1960    --BEGIN  
1961    --    UPDATE Seat  
1962    --    SET Status = 'Occupied'  
1963    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1964    --END  
1965  
1966    --ALTER TRIGGER AutoReserve  
1967    --ON Ticket  
1968    --FOR UPDATE  
1969    --AS  
1970    --BEGIN  
1971    --    UPDATE Seat  
1972    --    SET Status = 'Available'  
1973    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1974    --END  
1975  
1976    --ALTER TRIGGER AutoReserve  
1977    --ON Ticket  
1978    --FOR UPDATE  
1979    --AS  
1980    --BEGIN  
1981    --    UPDATE Seat  
1982    --    SET Status = 'Occupied'  
1983    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1984    --END  
1985  
1986    --ALTER TRIGGER AutoReserve  
1987    --ON Ticket  
1988    --FOR UPDATE  
1989    --AS  
1990    --BEGIN  
1991    --    UPDATE Seat  
1992    --    SET Status = 'Available'  
1993    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
1994    --END  
1995  
1996    --ALTER TRIGGER AutoReserve  
1997    --ON Ticket  
1998    --FOR UPDATE  
1999    --AS  
2000    --BEGIN  
2001    --    UPDATE Seat  
2002    --    SET Status = 'Occupied'  
2003    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
2004    --END  
2005  
2006    --ALTER TRIGGER AutoReserve  
2007    --ON Ticket  
2008    --FOR UPDATE  
2009    --AS  
2010    --BEGIN  
2011    --    UPDATE Seat  
2012    --    SET Status = 'Available'  
2013    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
2014    --END  
2015  
2016    --ALTER TRIGGER AutoReserve  
2017    --ON Ticket  
2018    --FOR UPDATE  
2019    --AS  
2020    --BEGIN  
2021    --    UPDATE Seat  
2022    --    SET Status = 'Occupied'  
2023    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
2024    --END  
2025  
2026    --ALTER TRIGGER AutoReserve  
2027    --ON Ticket  
2028    --FOR UPDATE  
2029    --AS  
2030    --BEGIN  
2031    --    UPDATE Seat  
2032    --    SET Status = 'Available'  
2033    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
2034    --END  
2035  
2036    --ALTER TRIGGER AutoReserve  
2037    --ON Ticket  
2038    --FOR UPDATE  
2039    --AS  
2040    --BEGIN  
2041    --    UPDATE Seat  
2042    --    SET Status = 'Occupied'  
2043    --    WHERE FlightID = (SELECT FlightID FROM inserted)  
2044    --END  
2045  
2046   
```

```

1113    CREATE TRIGGER tr_ReserveSeatOnTicketIssue
1114        ON Ticket
1115        AFTER INSERT, UPDATE
1116        AS
1117        BEGIN
1118            SET NOCOUNT ON;
1119
1120            -- Only proceed if SeatNumber is specified in the inserted rows
1121            IF EXISTS (SELECT 1 FROM inserted WHERE SeatNumber IS NOT NULL)
1122            BEGIN
1123                -- Update seat status to 'Reserved' for newly assigned seats in inserted rows
1124                UPDATE s
1125                    SET s.Status = 'Reserved'
1126                    FROM Seat s
1127                    JOIN inserted i ON s.FlightID = i.FlightID AND s.SeatNumber = i.SeatNumber
1128                    WHERE i.SeatNumber IS NOT NULL;
1129
1130                -- Handle updates to ticket seat assignments:
1131                -- Free previously assigned seats if SeatNumber changes
1132                IF EXISTS (SELECT 1 FROM deleted)
1133                BEGIN
1134                    UPDATE s
1135                        SET s.Status = 'Available'
1136                        FROM Seat s
1137                        JOIN deleted d ON s.FlightID = d.FlightID AND s.SeatNumber = d.SeatNumber
1138                        LEFT JOIN inserted i ON d.TicketID = i.TicketID
1139                        WHERE d.SeatNumber IS NOT NULL
1140                            AND (i.SeatNumber IS NULL OR i.SeatNumber <> d.SeatNumber); -- Check if SeatNumber changed
1141
1142                END;
1143            END;
1144        GO

```

Query executed successfully at 05:51:59 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 0 rows

- *Testing the Trigger*

Let's test the functionality:

- First, populate some seats for our test flight

```

1217    SELECT @TicketID AS NewTicketID, @BoardingNumber AS BoardingNumber;
1218    END TRY
1219    BEGIN CATCH
1220        -- Rollback transaction if an error occurs
1221        ROLLBACK TRANSACTION;
1222        THROW;
1223    END CATCH
1224    GO
1225
1226    INSERT INTO Seat (FlightID, SeatNumber, Class, Status)
1227        VALUES
1228            (300, '12A', 'Economy', 'Available'),
1229            (300, '12B', 'Economy', 'Available'),
1230            (300, '1A', 'Business', 'Available'),
1231            (301, '3B', 'Business', 'Available');
1232    GO
1233
1234
1235    -- Update a ticket to change seats
1236

```

(4 rows) affected

Query executed successfully at 05:54:38 | (localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 0 rows

- Update a ticket to change seats

```

1226
1227     INSERT INTO Seat (FlightID, SeatNumber, Class, Status)
1228     VALUES
1229         (3800, '12A', 'Economy', 'Available'),
1230         (3800, '12B', 'Economy', 'Available'),
1231         (3800, '1A', 'Business', 'Available'),
1232         (3801, '3B', 'Business', 'Available');
1233     GO
1234
1235
1236     -- Update a ticket to change seats
1237     UPDATE Ticket
1238     SET SeatNumber = 'A47'
1239     WHERE TicketID = 4090;
1240
1241     SELECT * FROM Ticket
1242
1243     --QUESTION 7 - TOTAL BAGGAGE
1244     --create a table-valued function that returns the total number of checked-in baggage items
1245     --for a specified flight on a specific date, along with relevant details

```

(1 row(s) affected)

Query executed successfully at 05:57:29

100 % No issues found

1237 Ch: 1 MIXED CRLF

T-SQL Message

	TicketID	PNR	FlightID	IssueDate	IssueTime	Fare	SeatNumber	Class	IssuedBy	EBoardingNumber
1	4088	PNR0001	3000	2025-04-25	04:07:10.630000	203.75	A1	Business	EM202	EBN0001
2	4089	PNR0002	3001	2025-04-25	04:07:10.630000	207.50	A2	FirstClass	EM203	EBN0002
3	4090	PNR0003	3002	2025-04-25	04:07:10.630000	211.25	A47	Economy	EM204	EBN0003
4	4091	PNR0004	3003	2025-04-25	04:07:10.630000	215.00	A4	Business	EM205	EBN0004
5	4093	PNR0006	3001	2025-04-25	04:07:10.630000	222.50	A6	Economy	EM201	EBN0006
6	4094	PNR0007	3002	2025-04-25	04:07:10.630000	226.25	A7	Business	EM202	EBN0007
7	4095	PNR0008	3003	2025-04-25	04:07:10.630000	230.00	A8	FirstClass	EM203	EBN0008
8	4096	PNR0009	3000	2025-04-25	04:07:10.630000	233.75	A9	Economy	EM204	EBN0009
9	4097	PNR0010	3001	2025-04-25	04:07:10.630000	237.50	A10	Business	EM205	EBN0010
10	4099	PNR0012	3003	2025-04-25	04:07:10.630000	245.00	A12	Economy	EM201	EBN0012
11	4100	PNR0013	3000	2025-04-25	04:07:10.630000	248.75	A13	Business	EM202	EBN0013
12	4101	PNR0014	3001	2025-04-25	04:07:10.630000	252.50	A14	FirstClass	EM203	EBN0014
13	4102	PNR0015	3002	2025-04-25	04:07:10.630000	256.25	A15	Economy	EM204	EBN0015
14	4103	PNR0016	3003	2025-04-25	04:07:10.630000	260.00	A16	Business	EM205	EBN0016
15	4105	PNR0018	3001	2025-04-25	04:07:10.630000	267.50	A18	Economy	EM201	EBN0018
16	4106	PNR0019	3002	2025-04-25	04:07:10.630000	271.25	A19	Business	EM202	EBN0019
17	4107	PNR0020	3003	2025-04-25	04:07:10.630000	275.00	A20	FirstClass	EM203	EBN0020
18	4108	PNR0021	3000	2025-04-25	04:07:10.630000	278.75	A21	Economy	EM204	EBN0021
19	4109	PNR0022	3001	2025-04-25	04:07:10.630000	282.50	A22	Business	EM205	EBN0022
40	4111	PNR0024	3003	2025-04-25	04:07:10.630000	290.00	A24	Economy	EM203	EBN0024

Query executed successfully at 05:58:11

100 % No issues found

1241 Ch: 1 MIXED CRLF

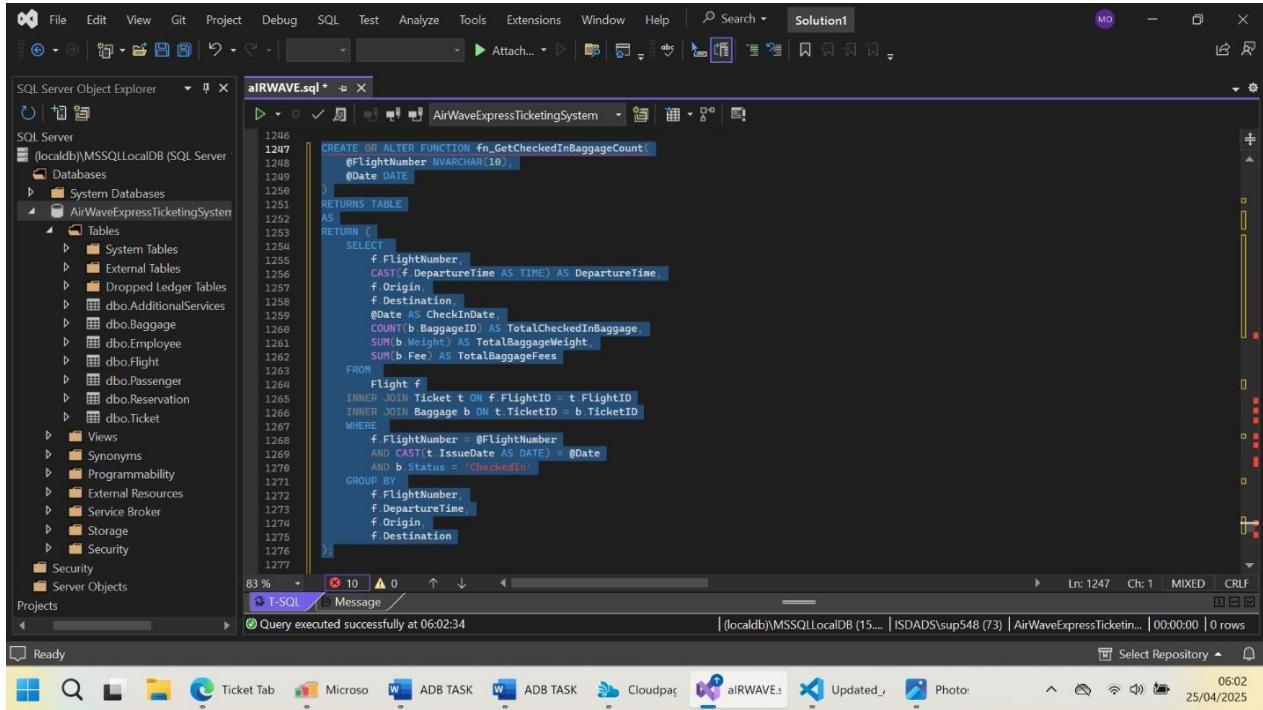
T-SQL Results Message

This solution provides a robust mechanism for managing seat allocations automatically whenever tickets are issued or modified, ensuring data consistency and preventing seat conflicts.

3.6 Q7. Function to Identify Checked-In Baggage for a Specific Flight and Date

I created a table-valued function that returns the total number of checked-in baggage items for a specified flight on a specific date, along with relevant details.

- *Implementation*



The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. The 'Tables' node is expanded, showing various tables like Flight, Ticket, and Baggage. The 'Script' button next to the 'Flight' table is highlighted. The main pane displays the T-SQL code for creating a function:

```
CREATE OR ALTER FUNCTION fn_GetCheckedInBaggageCount(
    @FlightNumber NVARCHAR(10),
    @Date DATE
)
RETURNS TABLE
AS
BEGIN
    RETURN (
        SELECT
            f.FlightNumber,
            CAST(f.DepartureTime AS TIME) AS DepartureTime,
            f.Origin,
            f.Destination,
            @Date AS CheckInDate,
            COUNT(b.BaggageID) AS TotalCheckedInBaggage,
            SUM(b.Weight) AS TotalBaggageWeight,
            SUM(b.Fee) AS TotalBaggageFees
        FROM
            Flight f
        INNER JOIN Ticket t ON f.FlightID = t.FlightID
        INNER JOIN Baggage b ON t.TicketID = b.TicketID
        WHERE
            f.FlightNumber = @FlightNumber
            AND CAST(t.IssueDate AS DATE) = @Date
            AND b.Status = 'CheckedIn'
        GROUP BY
            f.FlightNumber,
            f.DepartureTime,
            f.Origin,
            f.Destination
    )
END
```

The status bar at the bottom indicates the query was executed successfully at 06:02:34. The taskbar at the bottom shows several open tabs and windows, including 'Ticket Tab', 'Microso', 'ADB TASK', 'ADB TASK', 'Cloudpa...', 'aiRWA...', 'Updated...', 'Photo...', and the current window 'aiRWA...'. The system tray shows the date as 25/04/2025.

Usage Examples

Using the function for a specific flight and date:

Get checked-in baggage for flight AW101 on April 25, 2025

The screenshot shows the SQL Server Object Explorer on the left, displaying the database structure for 'AirWaveExpressTicketingSystem'. The 'Tables' node is expanded, showing various tables like dbo.Flight, dbo.Employee, etc. The 'Script' button is selected for the 'Flight' table. The main pane contains a T-SQL script for querying checked-in baggage. The 'Results' tab shows the output of the query:

FlightNumber	DepartureTime	Origin	Destination	CheckinDate	TotalCheckedInBaggage	TotalBaggageWeight	TotalBaggageFees
AW101	09:00:00.0000000	Lisbon	New York JFK	2025-04-25	4	98.55	\$45.50

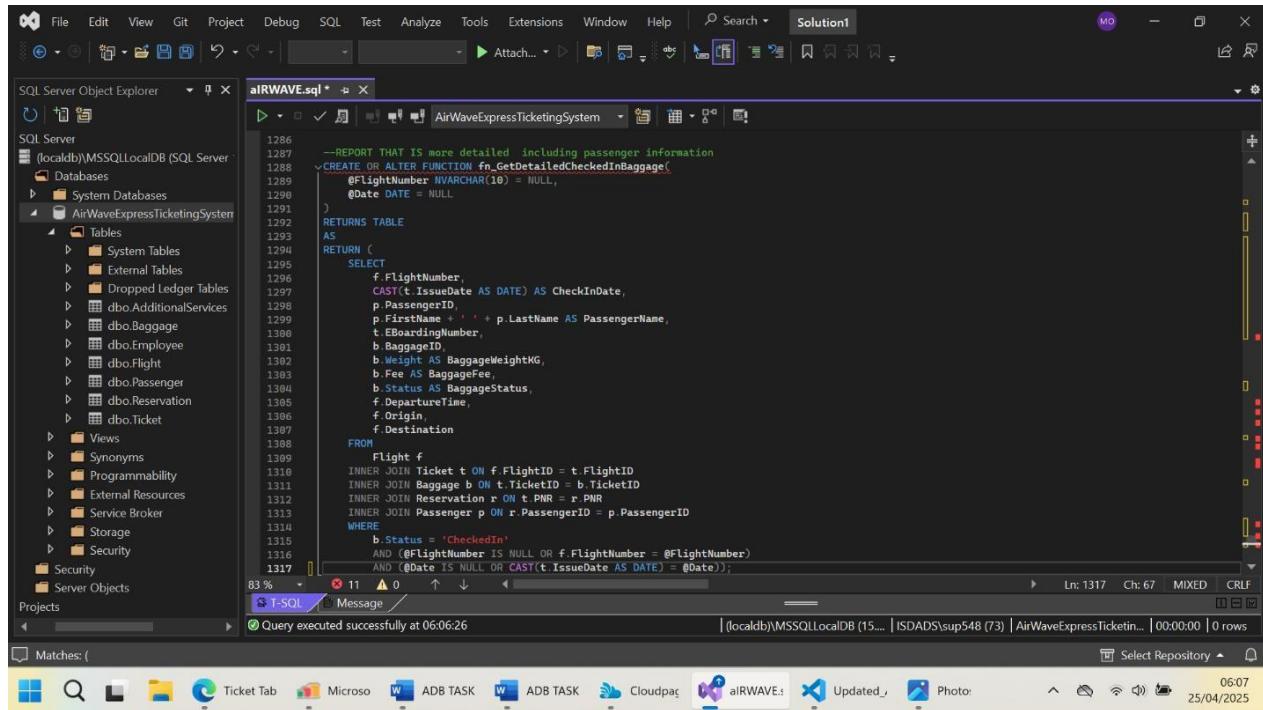
Below the results, a message indicates the query was executed successfully at 06:04:47.

Get checked-in baggage for current date

The screenshot is identical to the previous one, showing the same database structure and T-SQL script. The only difference is the execution time, which is now 06:05:42. The results table remains the same:

FlightNumber	DepartureTime	Origin	Destination	CheckinDate	TotalCheckedInBaggage	TotalBaggageWeight	TotalBaggageFees
AW101	09:00:00.0000000	Lisbon	New York JFK	2025-04-25	4	98.55	\$45.50

- Enhanced Version with Passenger Details



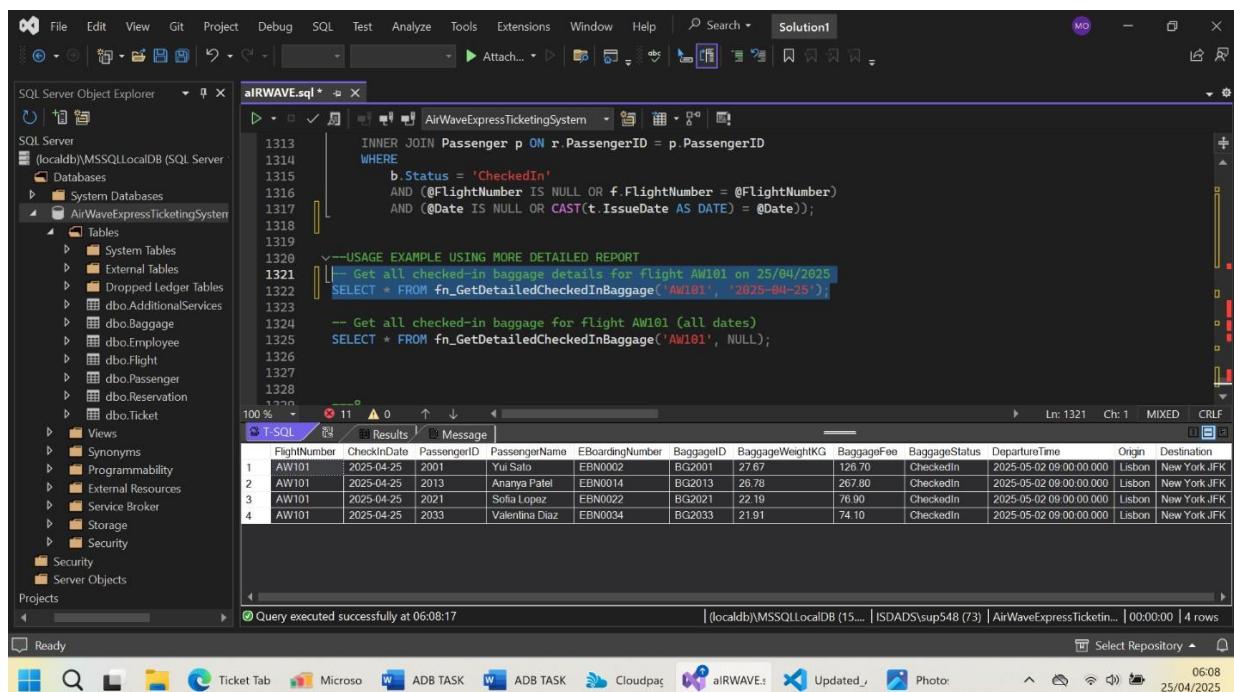
```

1286 --REPORT THAT IS more detailed including passenger information
1287 CREATE OR ALTER FUNCTION fn_GetDetailedCheckedInBaggage(
1288     @FlightNumber NVARCHAR(10) = NULL,
1289     @Date DATE = NULL
1290 )
1291 RETURNS TABLE
1292 AS
1293 RETURN (
1294     SELECT
1295         f.FlightNumber,
1296         CAST(t.IssueDate AS DATE) AS CheckinDate,
1297         p.PassengerID,
1298         p.Firstname + ' ' + p.Lastname AS PassengerName,
1299         t.BookingNumber,
1300         b.BaggageID,
1301         b.Weight AS BaggageWeightKG,
1302         b.Fee AS BaggageFee,
1303         b.Status AS BaggageStatus,
1304         f.DepartureTime,
1305         f.Origin,
1306         f.Destination
1307     FROM
1308         Flight f
1309     INNER JOIN Ticket t ON f.FlightID = t.FlightID
1310     INNER JOIN Baggage b ON t.TicketID = b.TicketID
1311     INNER JOIN Reservation r ON t.PNR = r.PNR
1312     INNER JOIN Passenger p ON r.PassengerID = p.PassengerID
1313     WHERE
1314         b.Status = 'CheckedIn'
1315         AND (@FlightNumber IS NULL OR f.FlightNumber = @FlightNumber)
1316         AND (@Date IS NULL OR CAST(t.IssueDate AS DATE) = @Date));

```

Query executed successfully at 06:06:26

(localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 0 rows



```

1313     INNER JOIN Passenger p ON r.PassengerID = p.PassengerID
1314     WHERE
1315         b.Status = 'CheckedIn'
1316         AND (@FlightNumber IS NULL OR f.FlightNumber = @FlightNumber)
1317         AND (@Date IS NULL OR CAST(t.IssueDate AS DATE) = @Date);
1318
1319
1320     --USAGE EXAMPLE USING MORE DETAILED REPORT
1321     |-- Get all checked-in baggage details for flight AW101 on 25/04/2025
1322     SELECT * FROM fn_GetDetailedCheckedInBaggage('AW101', '2025-04-25');
1323
1324     -- Get all checked-in baggage for flight AW101 (all dates)
1325     SELECT * FROM fn_GetDetailedCheckedInBaggage('AW101', NULL);

```

FlightNumber	CheckinDate	PassengerID	PassengerName	BoardingNumber	BaggageID	BaggageWeightKG	BaggageFee	BaggageStatus	DepartureTime	Origin	Destination
AW101	2025-04-25	2001	Yuri Salo	EBCN0002	BG2001	27.67	126.70	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK
AW101	2025-04-25	2013	Ananya Patel	EBCN0014	BG2013	26.78	267.80	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK
AW101	2025-04-25	2021	Sofia Lopez	EBCN0022	BG2021	22.19	76.90	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK
AW101	2025-04-25	2033	Valentina Diaz	EBCN0034	BG2033	21.91	74.10	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK

Query executed successfully at 06:08:17

(localdb)\MSSQLLocalDB (15...) | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 4 rows

```

1313     INNER JOIN Passenger p ON r.PassengerID = p.PassengerID
1314     WHERE
1315         b.Status = 'CheckedIn'
1316         AND (@FlightNumber IS NULL OR f.FlightNumber = @FlightNumber)
1317         AND (@Date IS NULL OR CAST(t.IssueDate AS DATE) = @Date));
1318
1319
1320    --USAGE EXAMPLE USING MORE DETAILED REPORT
1321    |-- Get all checked-in baggage details for flight AW101 on 25/04/2025
1322    SELECT * FROM fn_GetDetailedCheckedInBaggage('AW101', '2025-04-25');
1323
1324    |-- Get all checked-in baggage for flight AW101 (all dates)
1325    SELECT * FROM fn_GetDetailedCheckedInBaggage('AW101', NULL);
1326
1327
1328

```

	FlightNumber	CheckinDate	PassengerID	PassengerName	EBoardingNumber	BaggageID	BaggageWeightKG	BaggageFee	BaggageStatus	DepartureTime	Origin	Destination
1	AW101	2025-04-25	2001	Yui Salo	EBN0002	BG2001	27.67	126.70	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK
2	AW101	2025-04-25	2013	Ananya Patel	EBN0014	BG2013	26.78	267.80	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK
3	AW101	2025-04-25	2021	Sofia Lopez	EBN0022	BG2021	22.19	76.90	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK
4	AW101	2025-04-25	2033	Valentina Diaz	EBN0034	BG2033	21.91	74.10	CheckedIn	2025-05-02 09:00:00.000	Lisbon	New York JFK

Query executed successfully at 06:09:02

This solution gives the ticketing system complete visibility into baggage check-in operations for any flight on any date, with the flexibility to view summary statistics or detailed passenger-level information.

3.7 Q8. Additional Database Objects for Enhanced Airport Ticketing System

I created these indexes to make the system faster and more efficient, especially for the kinds of searches and actions users will do most often—like looking up a passenger by email during login, finding flights based on time and route, or checking a passenger's reservations and baggage. By adding indexes to the columns most frequently used in searches, filters, and joins, I've helped the database respond more quickly and reduced the load during common tasks. It's all about making the experience smoother for both users and the system.

```

1328
1329    --8
1330
1331    CREATE INDEX IX_Passenger_Email ON Passenger>Email);
1332    /*Improve lookups by email, e.g., for login or duplicate checks
1333
1334    |--Flight
1335    CREATE INDEX IX_Flight_Departure ON Flight(DepartureTime);
1336    CREATE INDEX IX_Flight_OriginDestination ON Flight(Origin, Destination);
1337    /*Useful for searches by time, origin, and destination
1338
1339    |--Reservation
1340    CREATE INDEX IX_Reservation_PassengerID ON Reservation(PassengerID);
1341    CREATE INDEX IX_Reservation_Status ON Reservation(Status);
1342    /*Helps when listing or filtering bookings by passenger or status
1343
1344    |--Ticket
1345    CREATE INDEX IX_Ticket_PNR ON Ticket(PNR);
1346    CREATE INDEX IX_Ticket_FlightID ON Ticket(FlightID);
1347    CREATE INDEX IX_Ticket_IssuedBy ON Ticket(IssuedBy);
1348    /*Speeds up joins with reservations, flights, and employees
1349
1350    |--AdditionalServices
1351    CREATE INDEX IX_AdditionalServices_TicketID ON AdditionalServices(TicketID);
1352    /*Speeds up joins between Ticket and AdditionalServices, especially when retrieving services like baggage or upgraded me
1353
1354    |--Baggage
1355    CREATE INDEX IX_Baggage_PassengerID ON Baggage(PassengerID);
1356    CREATE INDEX IX_Baggage_TicketID ON Baggage(TicketID);
1357    /*PassengerID index helps when listing or verifying baggage per passenger (e.g., during check-in).
1358    /*TicketID index improves joins with Ticket table or when calculating baggage fees related to a specific ticket.
1359
1360    |--Seat
1361    CREATE INDEX IX_Seat_FlightID ON Seat(FlightID);
1362    CREATE INDEX IX_Seat_Status ON Seat(
1363    /*Useful for checking available/reserved seats

```

Ln: 1098 Ch: 1 MIXED CRLF

New Connection Opened | (localdb)\MSSQLLocalDB (15.... | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 0 rows

T-SQL Message

```

1352    /*Speeds up joins between Ticket and AdditionalServices, especially when retrieving services like baggage or upgraded me
1353
1354    |--Baggage
1355    CREATE INDEX IX_Baggage_PassengerID ON Baggage(PassengerID);
1356    CREATE INDEX IX_Baggage_TicketID ON Baggage(TicketID);
1357    /*PassengerID index helps when listing or verifying baggage per passenger (e.g., during check-in).
1358    /*TicketID index improves joins with Ticket table or when calculating baggage fees related to a specific ticket.
1359
1360    |--Seat
1361    CREATE INDEX IX_Seat_FlightID ON Seat(FlightID);
1362    CREATE INDEX IX_Seat_Status ON Seat(
1363    /*Useful for checking available/reserved seats

```

Ln: 1098 Ch: 1 MIXED CRLF

New Connection Opened | (localdb)\MSSQLLocalDB (15.... | ISDADS\sup548 (73) | AirWaveExpressTicketin... | 00:00:00 | 0 rows

T-SQL Message

4. Additional Suggestions

To make the system truly ready for everyday use, a few extra touches would go a long way like hashing passwords for safety, using parameterized queries to guard against SQL

injection, and adding clear error messages to help with troubleshooting. Making sure commonly searched fields are indexed and using transactions when needed can also help things run more smoothly. Simple checks, like catching duplicate emails or validating roles, help keep the data clean and user-friendly.

5. Final Thoughts

All in all, this database does a solid job of meeting the needs of an airport ticketing system. It's built with care, keeps things secure and fast, and follows best practices to make sure it runs reliably now and scales well in the future.

