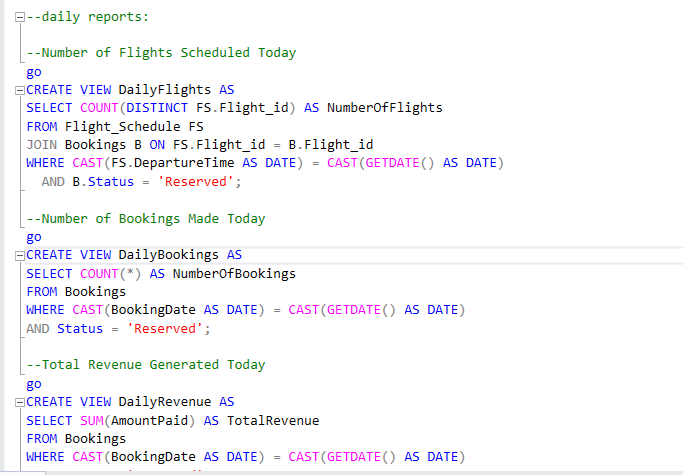


* 4th Deliverable (ERD Diagram)
* Group Members:
  + Abdullah Ahmad Arslan (FA22-BCS-042), CGPA: 3.75, Mail: [FA22-BCS-042@cuilahore.edu.pk,](mailto:FA22-BCS-042@cuilahore.edu.pk) Phone: 03334578883
  + Muhammad Awais (FA22-BCS-185), CGPA: 3.41, Mail: [FA22-BCS-185@cuilahore.edu.pk,](mailto:FA22-BCS-185@cuilahore.edu.pk) Phone: 03320144667
* **Subject:** Database Systems
* **Professor Name:** Dr. Abid Sohail

**DAILY REPORTS(using complex views from normalized scehma):**



A white background with colorful text

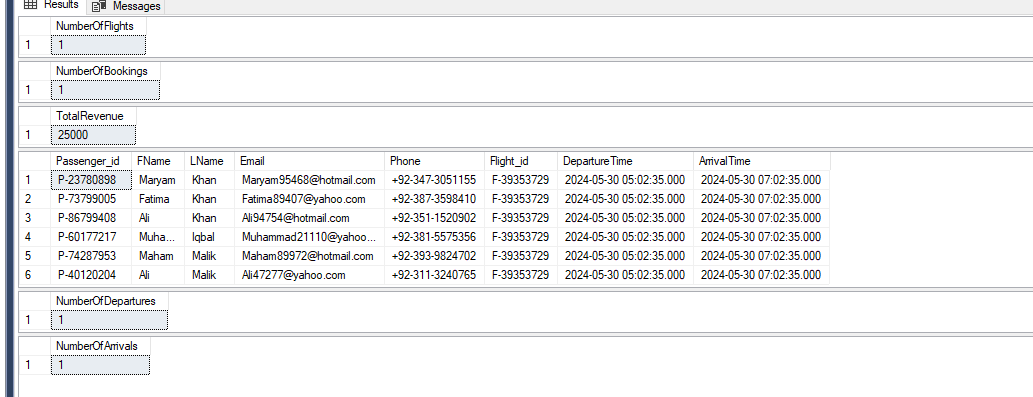
Description automatically generated

A screen shot of a computer

Description automatically generated

A close up of text

Description automatically generated



**Analytical reports(using materialized views)**

CREATE VIEW MonthlyRevenue AS

WITH UniqueBookings AS (

SELECT

DISTINCT Booking\_id,

Amount\_Paid,

Booking\_Date

FROM

dbo.DenormalizedIndexedView

WHERE

Booking\_Status = 'reserved'

)

SELECT

YEAR(Booking\_Date) AS Year,

MONTH(Booking\_Date) AS Month,

SUM(Amount\_Paid) AS TotalRevenue

FROM

UniqueBookings

GROUP BY

YEAR(Booking\_Date),

MONTH(Booking\_Date);

go

--Passenger Demographics Report:

CREATE VIEW PassengerDemographics AS

SELECT

Passenger\_Gender,

COUNT(DISTINCT Passenger\_id) AS NumberOfPassengers,

AVG(DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE())) AS AverageAge

FROM

dbo.DenormalizedIndexedView

GROUP BY

Passenger\_Gender;

go

--Flight Capacity Utilization Report(reserved ones only):

CREATE VIEW FlightCapacityUtilization AS

SELECT

Flight\_id,

Aircraft\_id,

Aircraft\_Capacity,

COUNT(DISTINCT Booking\_id) AS BookedSeats,

(COUNT(DISTINCT Booking\_id) \* 1.0 / Aircraft\_Capacity) \* 100 AS UtilizationPercentage

FROM

dbo.DenormalizedIndexedView

WHERE

Booking\_Status = 'reserved'

GROUP BY

Flight\_id,

Aircraft\_id,

Aircraft\_Capacity;

go

--Class Distribution Report:

CREATE VIEW ClassDistribution AS

SELECT

Ticket\_Class,

COUNT(DISTINCT Booking\_id) AS NumberOfTickets

FROM

dbo.DenormalizedIndexedView

WHERE

Booking\_Status = 'reserved'

GROUP BY

Ticket\_Class;

go

--Age Group Distribution Report:

CREATE VIEW AgeGroupDistribution AS

SELECT

CASE

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) < 18 THEN 'Under 18'

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) BETWEEN 18 AND 25 THEN '18-25'

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) BETWEEN 26 AND 35 THEN '26-35'

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) BETWEEN 36 AND 50 THEN '36-50'

ELSE 'Above 50'

END AS AgeGroup,

COUNT(DISTINCT Passenger\_id) AS NumberOfPassengers

FROM

dbo.DenormalizedIndexedView

WHERE

Booking\_Status = 'reserved'

GROUP BY

CASE

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) < 18 THEN 'Under 18'

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) BETWEEN 18 AND 25 THEN '18-25'

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) BETWEEN 26 AND 35 THEN '26-35'

WHEN DATEDIFF(YEAR, Passenger\_DateOfBirth, GETDATE()) BETWEEN 36 AND 50 THEN '36-50'

ELSE 'Above 50'

END;

go

--Popular Payment Methods Report:

CREATE VIEW PopularPaymentMethods AS

SELECT

Payment\_Method,

COUNT(DISTINCT Booking\_id) AS NumberOfBookings

FROM

dbo.DenormalizedIndexedView

WHERE

Booking\_Status = 'reserved'

GROUP BY

Payment\_Method;

go

--Flights by Booking Status:

CREATE VIEW FlightsByBookingStatus AS

SELECT

Booking\_Status,

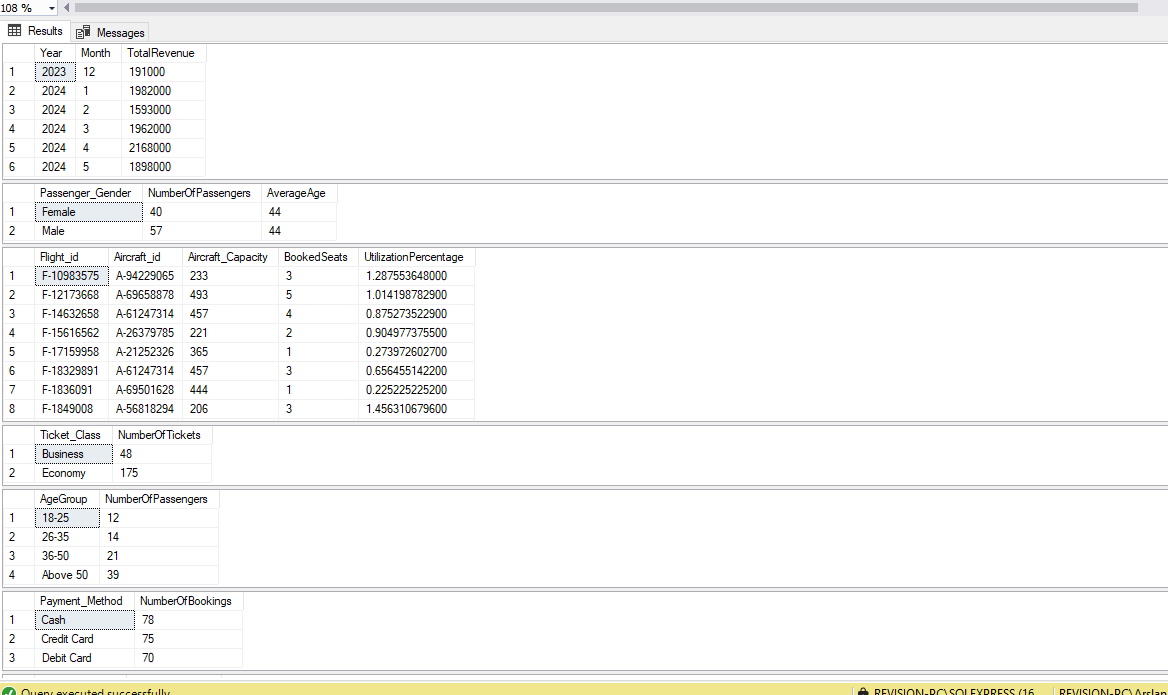
COUNT(DISTINCT Booking\_id) AS NumberOfFlights

FROM

dbo.DenormalizedIndexedView

GROUP BY

Booking\_Status;



**Reports(using Procedures):**

CREATE PROCEDURE GetPassengerBookingSummary

AS

BEGIN

SELECT

p.Passenger\_id,

p.FName,

p.LName,

COUNT(b.Bk\_id) AS TotalBookings,

SUM(b.AmountPaid) AS TotalAmountPaid

FROM

Passengers p

LEFT JOIN

Bookings b ON p.Passenger\_id = b.Passenger\_id

WHERE

b.Status = 'reserved'

GROUP BY

p.Passenger\_id,

p.FName,

p.LName

ORDER BY

TotalAmountPaid DESC;

END;

exec GetPassengerBookingSummary

--Flight Schedule and Crew Assignment

CREATE PROCEDURE GetFlightScheduleAndCrew

AS

BEGIN

SELECT

fs.Flight\_id,

fs.DepartureTime,

fs.ArrivalTime,

a.Aircraft\_Name,

ad.FName AS AdminFirstName,

ad.LName AS AdminLastName,

cs.FName AS CrewFirstName,

cs.LName AS CrewLastName,

cs.Designation

FROM

Flight\_Schedule fs

JOIN

Aircrafts a ON fs.Ac\_id = a.Ac\_id

JOIN

Admins ad ON fs.Admin\_id = ad.Admin\_id

JOIN

StaffAssignments sa ON fs.Flight\_id = sa.Flight\_id

JOIN

Crew\_Staff cs ON sa.Staff\_id = cs.Staff\_id

WHERE

fs.DepartureTime IS NOT NULL AND fs.ArrivalTime IS NOT NULL

ORDER BY

fs.DepartureTime;

END;

exec GetFlightScheduleAndCrew

--Revenue Breakdown by Passenger Class

CREATE PROCEDURE GetRevenueBreakdownByPassengerClass

AS

BEGIN

SELECT

t.Class,

COUNT(b.Bk\_id) AS TotalBookings,

SUM(b.AmountPaid) AS TotalRevenue

FROM

Bookings b

JOIN

Tickets t ON b.Tkt\_id = t.Tkt\_id

WHERE

b.Status = 'reserved'

GROUP BY

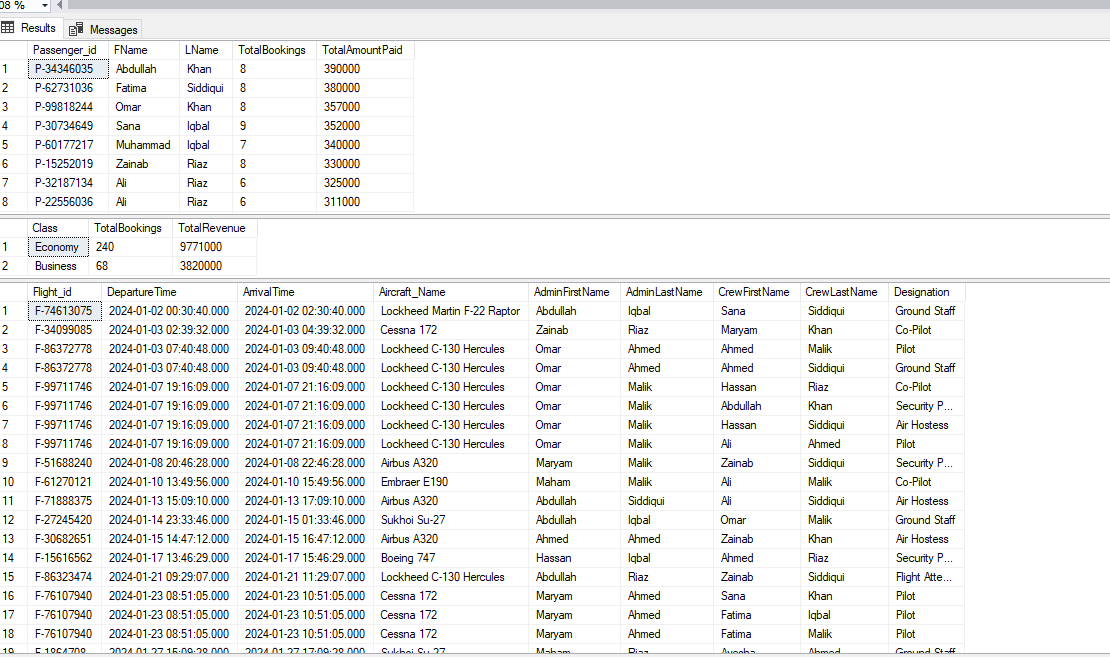
t.Class

ORDER BY

TotalRevenue DESC;

END;

EXEC GetRevenueBreakdownByPassengerClass



**CRUD PROCEDURES:**

--CRUD Procedures

-- CRUD procedures for Passengers table

CREATE PROCEDURE InsertPassenger

@Passenger\_id VARCHAR(10),

@FName VARCHAR(100),

@LName VARCHAR(100),

@Email VARCHAR(100),

@Phone VARCHAR(20),

@Address VARCHAR(255),

@DateOfBirth DATE,

@Gender VARCHAR(10),

@Cnic VARCHAR(50),

@passport\_no VARCHAR(10)

AS

BEGIN

INSERT INTO Passengers (Passenger\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender, Cnic, passport\_no)

VALUES (@Passenger\_id, @FName, @LName, @Email, @Phone, @Address, @DateOfBirth, @Gender, @Cnic, @passport\_no);

END;

GO

CREATE PROCEDURE UpdatePassenger

@Passenger\_id VARCHAR(10),

@FName VARCHAR(100),

@LName VARCHAR(100),

@Email VARCHAR(100),

@Phone VARCHAR(20),

@Address VARCHAR(255),

@DateOfBirth DATE,

@Gender VARCHAR(10),

@Cnic VARCHAR(50),

@passport\_no VARCHAR(10)

AS

BEGIN

UPDATE Passengers

SET FName = @FName,

LName = @LName,

Email = @Email,

Phone = @Phone,

Address = @Address,

DateOfBirth = @DateOfBirth,

Gender = @Gender,

Cnic = @Cnic,

passport\_no = @passport\_no

WHERE Passenger\_id = @Passenger\_id;

END;

GO

CREATE PROCEDURE DeletePassenger

@Passenger\_id VARCHAR(10)

AS

BEGIN

DELETE FROM Passengers WHERE Passenger\_id = @Passenger\_id;

END;

GO

CREATE PROCEDURE SelectAllPassengers

AS

BEGIN

SELECT \* FROM Passengers;

END;

GO

-- Procedure to check a specific passenger record

CREATE PROCEDURE CheckPassenger

@Passenger\_id VARCHAR(10)

AS

BEGIN

SELECT \* FROM Passengers WHERE Passenger\_id = @Passenger\_id;

END;

GO

-- CRUD procedures for Aircrafts table

CREATE PROCEDURE InsertAircraft

@Ac\_id VARCHAR(10),

@Aircraft\_Name VARCHAR(100),

@Model VARCHAR(100),

@Capacity INT

AS

BEGIN

INSERT INTO Aircrafts (Ac\_id, Aircraft\_Name, Model, Capacity)

VALUES (@Ac\_id, @Aircraft\_Name, @Model, @Capacity);

END;

GO

CREATE PROCEDURE UpdateAircraft

@Ac\_id VARCHAR(10),

@Aircraft\_Name VARCHAR(100),

@Model VARCHAR(100),

@Capacity INT

AS

BEGIN

UPDATE Aircrafts

SET Aircraft\_Name = @Aircraft\_Name,

Model = @Model,

Capacity = @Capacity

WHERE Ac\_id = @Ac\_id;

END;

GO

CREATE PROCEDURE DeleteAircraft

@Ac\_id VARCHAR(10)

AS

BEGIN

DELETE FROM Aircrafts WHERE Ac\_id = @Ac\_id;

END;

GO

CREATE PROCEDURE SelectAllAircrafts

AS

BEGIN

SELECT \* FROM Aircrafts;

END;

GO

-- Procedure to check a specific aircraft record

CREATE PROCEDURE CheckAircraft

@Ac\_id VARCHAR(10)

AS

BEGIN

SELECT \* FROM Aircrafts WHERE Ac\_id = @Ac\_id;

END;

GO

-- CRUD procedures for Tickets table

CREATE PROCEDURE InsertTicket

@Tkt\_id VARCHAR(50),

@Class VARCHAR(50),

@Ticket\_from VARCHAR(100),

@Ticket\_to VARCHAR(100),

@Price FLOAT

AS

BEGIN

INSERT INTO Tickets (Tkt\_id, Class, Ticket\_from, Ticket\_to, Price)

VALUES (@Tkt\_id, @Class, @Ticket\_from, @Ticket\_to, @Price);

END;

GO

CREATE PROCEDURE UpdateTicket

@Tkt\_id VARCHAR(50),

@Class VARCHAR(50),

@Ticket\_from VARCHAR(100),

@Ticket\_to VARCHAR(100),

@Price FLOAT

AS

BEGIN

UPDATE Tickets

SET Class = @Class,

Ticket\_from = @Ticket\_from,

Ticket\_to = @Ticket\_to,

Price = @Price

WHERE Tkt\_id = @Tkt\_id;

END;

GO

CREATE PROCEDURE DeleteTicket

@Tkt\_id VARCHAR(50)

AS

BEGIN

DELETE FROM Tickets WHERE Tkt\_id = @Tkt\_id;

END;

GO

CREATE PROCEDURE SelectAllTickets

AS

BEGIN

SELECT \* FROM Tickets;

END;

GO

-- Procedure to check a specific ticket record

CREATE PROCEDURE CheckTicket

@Tkt\_id VARCHAR(50)

AS

BEGIN

SELECT \* FROM Tickets WHERE Tkt\_id = @Tkt\_id;

END;

GO

-- CRUD procedures for Admins table

CREATE PROCEDURE InsertAdmin

@Admin\_id VARCHAR(250),

@FName VARCHAR(100),

@LName VARCHAR(100),

@Email VARCHAR(100),

@Phone VARCHAR(20),

@Address VARCHAR(255),

@DateOfBirth DATE,

@Gender VARCHAR(10)

AS

BEGIN

INSERT INTO Admins (Admin\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender)

VALUES (@Admin\_id, @FName, @LName, @Email, @Phone, @Address, @DateOfBirth, @Gender);

END;

GO

CREATE PROCEDURE UpdateAdmin

@Admin\_id VARCHAR(250),

@FName VARCHAR(100),

@LName VARCHAR(100),

@Email VARCHAR(100),

@Phone VARCHAR(20),

@Address VARCHAR(255),

@DateOfBirth DATE,

@Gender VARCHAR(10)

AS

BEGIN

UPDATE Admins

SET FName = @FName,

LName = @LName,

Email = @Email,

Phone = @Phone,

Address = @Address,

DateOfBirth = @DateOfBirth,

Gender = @Gender

WHERE Admin\_id = @Admin\_id;

END;

GO

CREATE PROCEDURE DeleteAdmin

@Admin\_id VARCHAR(250)

AS

BEGIN

DELETE FROM Admins WHERE Admin\_id = @Admin\_id;

END;

GO

CREATE PROCEDURE SelectAllAdmins

AS

BEGIN

SELECT \* FROM Admins;

END;

GO

-- Procedure to check a specific admin record

CREATE PROCEDURE CheckAdmin

@Admin\_id VARCHAR(250)

AS

BEGIN

SELECT \* FROM Admins WHERE Admin\_id = @Admin\_id;

END;

GO

-- CRUD procedures for Crew\_Staff table

CREATE PROCEDURE InsertCrewStaff

@Staff\_id VARCHAR(10),

@FName VARCHAR(100),

@LName VARCHAR(100),

@Designation VARCHAR(100),

@DateOfBirth DATE,

@Gender VARCHAR(10),

@Email VARCHAR(100),

@Phone VARCHAR(20),

@Address VARCHAR(255)

AS

BEGIN

INSERT INTO Crew\_Staff (Staff\_id, FName, LName, Designation, DateOfBirth, Gender, Email, Phone, Address)

VALUES (@Staff\_id, @FName, @LName, @Designation, @DateOfBirth, @Gender, @Email, @Phone, @Address);

END;

GO

CREATE PROCEDURE UpdateCrewStaff

@Staff\_id VARCHAR(10),

@FName VARCHAR(100),

@LName VARCHAR(100),

@Designation VARCHAR(100),

@DateOfBirth DATE,

@Gender VARCHAR(10),

@Email VARCHAR(100),

@Phone VARCHAR(20),

@Address VARCHAR(255)

AS

BEGIN

UPDATE Crew\_Staff

SET FName = @FName,

LName = @LName,

Designation = @Designation,

DateOfBirth = @DateOfBirth,

Gender = @Gender,

Email = @Email,

Phone = @Phone,

Address = @Address

WHERE Staff\_id = @Staff\_id;

END;

GO

CREATE PROCEDURE DeleteCrewStaff

@Staff\_id VARCHAR(10)

AS

BEGIN

DELETE FROM Crew\_Staff WHERE Staff\_id = @Staff\_id;

END;

GO

CREATE PROCEDURE SelectAllCrewStaff

AS

BEGIN

SELECT \* FROM Crew\_Staff;

END;

GO

-- Procedure to check a specific crew staff record

CREATE PROCEDURE CheckCrewStaff

@Staff\_id VARCHAR(10)

AS

BEGIN

SELECT \* FROM Crew\_Staff WHERE Staff\_id = @Staff\_id;

END;

GO

-- CRUD procedures for Flight\_Schedule table

CREATE PROCEDURE InsertFlightSchedule

@Flight\_id VARCHAR(250),

@DepartureTime DATETIME,

@ArrivalTime DATETIME,

@Ac\_id VARCHAR(10),

@Admin\_id VARCHAR(250)

AS

BEGIN

INSERT INTO Flight\_Schedule (Flight\_id, DepartureTime, ArrivalTime, Ac\_id, Admin\_id)

VALUES (@Flight\_id, @DepartureTime, @ArrivalTime, @Ac\_id, @Admin\_id);

END;

GO

CREATE PROCEDURE UpdateFlightSchedule

@Flight\_id VARCHAR(250),

@DepartureTime DATETIME,

@ArrivalTime DATETIME,

@Ac\_id VARCHAR(10),

@Admin\_id VARCHAR(250)

AS

BEGIN

UPDATE Flight\_Schedule

SET DepartureTime = @DepartureTime,

ArrivalTime = @ArrivalTime,

Ac\_id = @Ac\_id,

Admin\_id = @Admin\_id

WHERE Flight\_id = @Flight\_id;

END;

GO

CREATE PROCEDURE DeleteFlightSchedule

@Flight\_id VARCHAR(250)

AS

BEGIN

DELETE FROM Flight\_Schedule WHERE Flight\_id = @Flight\_id;

END;

GO

CREATE PROCEDURE SelectAllFlightSchedules

AS

BEGIN

SELECT \* FROM Flight\_Schedule;

END;

GO

-- Procedure to check a specific flight schedule record

CREATE PROCEDURE CheckFlightSchedule

@Flight\_id VARCHAR(250)

AS

BEGIN

SELECT \* FROM Flight\_Schedule WHERE Flight\_id = @Flight\_id;

END;

GO

-- CRUD procedures for StaffAssignments table

CREATE PROCEDURE InsertStaffAssignment

@Assg\_id VARCHAR(50),

@Flight\_id VARCHAR(250),

@Staff\_id VARCHAR(10),

@AssignmentDate DATETIME

AS

BEGIN

INSERT INTO StaffAssignments (Assg\_id, Flight\_id, Staff\_id, AssignmentDate)

VALUES (@Assg\_id, @Flight\_id, @Staff\_id, @AssignmentDate);

END;

GO

CREATE PROCEDURE UpdateStaffAssignment

@Assg\_id VARCHAR(50),

@Flight\_id VARCHAR(250),

@Staff\_id VARCHAR(10),

@AssignmentDate DATETIME

AS

BEGIN

UPDATE StaffAssignments

SET Flight\_id = @Flight\_id,

Staff\_id = @Staff\_id,

AssignmentDate = @AssignmentDate

WHERE Assg\_id = @Assg\_id;

END;

GO

CREATE PROCEDURE DeleteStaffAssignment

@Assg\_id VARCHAR(50)

AS

BEGIN

DELETE FROM StaffAssignments WHERE Assg\_id = @Assg\_id;

END;

GO

CREATE PROCEDURE SelectAllStaffAssignments

AS

BEGIN

SELECT \* FROM StaffAssignments;

END;

GO

-- Procedure to check a specific staff assignment record

CREATE PROCEDURE CheckStaffAssignment

@Assg\_id VARCHAR(50)

AS

BEGIN

SELECT \* FROM StaffAssignments WHERE Assg\_id = @Assg\_id;

END;

GO

-- CRUD procedures for Bookings table

CREATE PROCEDURE InsertBooking

@Bk\_id VARCHAR(20),

@Passenger\_id VARCHAR(10),

@Tkt\_id VARCHAR(50),

@PaymentMethod VARCHAR(50),

@AmountPaid FLOAT,

@Flight\_id VARCHAR(250),

@Seat\_No VARCHAR(10),

@BookingDate DATETIME,

@Status VARCHAR(50)

AS

BEGIN

INSERT INTO Bookings (Bk\_id, Passenger\_id, Tkt\_id, PaymentMethod, AmountPaid, Flight\_id, Seat\_No, BookingDate, Status)

VALUES (@Bk\_id, @Passenger\_id, @Tkt\_id, @PaymentMethod, @AmountPaid, @Flight\_id, @Seat\_No, @BookingDate, @Status);

END;

GO

CREATE PROCEDURE UpdateBooking

@Bk\_id VARCHAR(20),

@Passenger\_id VARCHAR(10),

@Tkt\_id VARCHAR(50),

@PaymentMethod VARCHAR(50),

@AmountPaid FLOAT,

@Flight\_id VARCHAR(250),

@Seat\_No VARCHAR(10),

@BookingDate DATETIME,

@Status VARCHAR(50)

AS

BEGIN

UPDATE Bookings

SET Passenger\_id = @Passenger\_id,

Tkt\_id = @Tkt\_id,

PaymentMethod = @PaymentMethod,

AmountPaid = @AmountPaid,

Flight\_id = @Flight\_id,

Seat\_No = @Seat\_No,

BookingDate = @BookingDate,

Status = @Status

WHERE Bk\_id = @Bk\_id;

END;

GO

CREATE PROCEDURE DeleteBooking

@Bk\_id VARCHAR(20)

AS

BEGIN

DELETE FROM Bookings WHERE Bk\_id = @Bk\_id;

END;

GO

CREATE PROCEDURE SelectAllBookings

AS

BEGIN

SELECT \* FROM Bookings;

END;

GO

-- Procedure to check a specific booking record

CREATE PROCEDURE CheckBooking

@Bk\_id VARCHAR(20)

AS

BEGIN

SELECT \* FROM Bookings WHERE Bk\_id = @Bk\_id;

END;

GO

**HISTORY TABLES:**

--log tables:

-- Create Passengers\_Log table

CREATE TABLE Passengers\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Passenger\_id VARCHAR(10),

FName VARCHAR(100),

LName VARCHAR(100),

Email VARCHAR(100),

Phone VARCHAR(20),

Address VARCHAR(255),

DateOfBirth DATE,

Gender VARCHAR(10),

Cnic VARCHAR(50),

passport\_no VARCHAR(10)

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_passenger\_inserts

ON Passengers

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Passengers\_Log (Operation, Operation\_Timestamp, UserName, Passenger\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender, Cnic, passport\_no)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Passenger\_id,

FName,

LName,

Email,

Phone,

Address,

DateOfBirth,

Gender,

Cnic,

passport\_no

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_passenger\_updates

ON Passengers

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Passengers\_Log (Operation, Operation\_Timestamp, UserName, Passenger\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender, Cnic, passport\_no)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Passenger\_id,

inserted.FName,

inserted.LName,

inserted.Email,

inserted.Phone,

inserted.Address,

inserted.DateOfBirth,

inserted.Gender,

inserted.Cnic,

inserted.passport\_no

FROM

inserted

JOIN

deleted

ON

inserted.Passenger\_id = deleted.Passenger\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_passenger\_deletes

ON Passengers

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Passengers\_Log (Operation, Operation\_Timestamp, UserName, Passenger\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender, Cnic, passport\_no)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Passenger\_id,

FName,

LName,

Email,

Phone,

Address,

DateOfBirth,

Gender,

Cnic,

passport\_no

FROM

deleted;

END;

GO

-- Create Aircrafts\_Log table

CREATE TABLE Aircrafts\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Ac\_id VARCHAR(10),

Aircraft\_Name VARCHAR(100),

Model VARCHAR(100),

Capacity INT

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_aircraft\_inserts

ON Aircrafts

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Aircrafts\_Log (Operation, Operation\_Timestamp, UserName, Ac\_id, Aircraft\_Name, Model, Capacity)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Ac\_id,

Aircraft\_Name,

Model,

Capacity

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_aircraft\_updates

ON Aircrafts

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Aircrafts\_Log (Operation, Operation\_Timestamp, UserName, Ac\_id, Aircraft\_Name, Model, Capacity)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Ac\_id,

inserted.Aircraft\_Name,

inserted.Model,

inserted.Capacity

FROM

inserted

JOIN

deleted

ON

inserted.Ac\_id = deleted.Ac\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_aircraft\_deletes

ON Aircrafts

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Aircrafts\_Log (Operation, Operation\_Timestamp, UserName, Ac\_id, Aircraft\_Name, Model, Capacity)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Ac\_id,

Aircraft\_Name,

Model,

Capacity

FROM

deleted;

END;

GO

-- Create Tickets\_Log table

CREATE TABLE Tickets\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Tkt\_id VARCHAR(50),

Class VARCHAR(50),

Ticket\_from VARCHAR(100),

Ticket\_to VARCHAR(100),

Price FLOAT

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_ticket\_inserts

ON Tickets

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Tickets\_Log (Operation, Operation\_Timestamp, UserName, Tkt\_id, Class, Ticket\_from, Ticket\_to, Price)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Tkt\_id,

Class,

Ticket\_from,

Ticket\_to,

Price

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_ticket\_updates

ON Tickets

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Tickets\_Log (Operation, Operation\_Timestamp, UserName, Tkt\_id, Class, Ticket\_from, Ticket\_to, Price)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Tkt\_id,

inserted.Class,

inserted.Ticket\_from,

inserted.Ticket\_to,

inserted.Price

FROM

inserted

JOIN

deleted

ON

inserted.Tkt\_id = deleted.Tkt\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_ticket\_deletes

ON Tickets

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Tickets\_Log (Operation, Operation\_Timestamp, UserName, Tkt\_id, Class, Ticket\_from, Ticket\_to, Price)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Tkt\_id,

Class,

Ticket\_from,

Ticket\_to,

Price

FROM

deleted;

END;

GO

-- Create Admins\_Log table

CREATE TABLE Admins\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Admin\_id VARCHAR(250),

FName VARCHAR(100),

LName VARCHAR(100),

Email VARCHAR(100),

Phone VARCHAR(20),

Address VARCHAR(255),

DateOfBirth DATE,

Gender VARCHAR(10)

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_admin\_inserts

ON Admins

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Admins\_Log (Operation, Operation\_Timestamp, UserName, Admin\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Admin\_id,

FName,

LName,

Email,

Phone,

Address,

DateOfBirth,

Gender

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_admin\_updates

ON Admins

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Admins\_Log (Operation, Operation\_Timestamp, UserName, Admin\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Admin\_id,

inserted.FName,

inserted.LName,

inserted.Email,

inserted.Phone,

inserted.Address,

inserted.DateOfBirth,

inserted.Gender

FROM

inserted

JOIN

deleted

ON

inserted.Admin\_id = deleted.Admin\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_admin\_deletes

ON Admins

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Admins\_Log (Operation, Operation\_Timestamp, UserName, Admin\_id, FName, LName, Email, Phone, Address, DateOfBirth, Gender)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Admin\_id,

FName,

LName,

Email,

Phone,

Address,

DateOfBirth,

Gender

FROM

deleted;

END;

GO

-- Create Crew\_Staff\_Log table

CREATE TABLE Crew\_Staff\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Staff\_id VARCHAR(10),

FName VARCHAR(100),

LName VARCHAR(100),

Designation VARCHAR(100),

DateOfBirth DATE,

Gender VARCHAR(10),

Email VARCHAR(100),

Phone VARCHAR(20),

Address VARCHAR(255)

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_crew\_staff\_inserts

ON Crew\_Staff

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Crew\_Staff\_Log (Operation, Operation\_Timestamp, UserName, Staff\_id, FName, LName, Designation, DateOfBirth, Gender, Email, Phone, Address)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Staff\_id,

FName,

LName,

Designation,

DateOfBirth,

Gender,

Email,

Phone,

Address

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_crew\_staff\_updates

ON Crew\_Staff

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Crew\_Staff\_Log (Operation, Operation\_Timestamp, UserName, Staff\_id, FName, LName, Designation, DateOfBirth, Gender, Email, Phone, Address)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Staff\_id,

inserted.FName,

inserted.LName,

inserted.Designation,

inserted.DateOfBirth,

inserted.Gender,

inserted.Email,

inserted.Phone,

inserted.Address

FROM

inserted

JOIN

deleted

ON

inserted.Staff\_id = deleted.Staff\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_crew\_staff\_deletes

ON Crew\_Staff

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Crew\_Staff\_Log (Operation, Operation\_Timestamp, UserName, Staff\_id, FName, LName, Designation, DateOfBirth, Gender, Email, Phone, Address)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Staff\_id,

FName,

LName,

Designation,

DateOfBirth,

Gender,

Email,

Phone,

Address

FROM

deleted;

END;

GO

-- Create Flight\_Schedule\_Log table

CREATE TABLE Flight\_Schedule\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Flight\_id VARCHAR(250),

DepartureTime DATETIME,

ArrivalTime DATETIME,

Ac\_id VARCHAR(10),

Admin\_id VARCHAR(250)

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_flight\_schedule\_inserts

ON Flight\_Schedule

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Flight\_Schedule\_Log (Operation, Operation\_Timestamp, UserName, Flight\_id, DepartureTime, ArrivalTime, Ac\_id, Admin\_id)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Flight\_id,

DepartureTime,

ArrivalTime,

Ac\_id,

Admin\_id

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_flight\_schedule\_updates

ON Flight\_Schedule

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Flight\_Schedule\_Log (Operation, Operation\_Timestamp, UserName, Flight\_id, DepartureTime, ArrivalTime, Ac\_id, Admin\_id)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Flight\_id,

inserted.DepartureTime,

inserted.ArrivalTime,

inserted.Ac\_id,

inserted.Admin\_id

FROM

inserted

JOIN

deleted

ON

inserted.Flight\_id = deleted.Flight\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_flight\_schedule\_deletes

ON Flight\_Schedule

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Flight\_Schedule\_Log (Operation, Operation\_Timestamp, UserName, Flight\_id, DepartureTime, ArrivalTime, Ac\_id, Admin\_id)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Flight\_id,

DepartureTime,

ArrivalTime,

Ac\_id,

Admin\_id

FROM

deleted;

END;

GO

-- Create StaffAssignments\_Log table

GO

CREATE TABLE StaffAssignments\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Assg\_id VARCHAR(50),

Flight\_id VARCHAR(250),

Staff\_id VARCHAR(10),

AssignmentDate DATETIME

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_staff\_assignments\_inserts

ON StaffAssignments

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StaffAssignments\_Log (Operation, Operation\_Timestamp, UserName, Assg\_id, Flight\_id, Staff\_id,AssignmentDate)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Assg\_id,

Flight\_id,

Staff\_id,

AssignmentDate

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_staff\_assignments\_updates

ON StaffAssignments

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StaffAssignments\_Log (Operation, Operation\_Timestamp, UserName, Assg\_id, Flight\_id, Staff\_id,AssignmentDate)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Assg\_id,

inserted.Flight\_id,

inserted.Staff\_id,

inserted.AssignmentDate

FROM

inserted

JOIN

deleted

ON

inserted.Assg\_id = deleted.Assg\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_staff\_assignments\_deletes

ON StaffAssignments

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StaffAssignments\_Log (Operation, Operation\_Timestamp, UserName, Assg\_id, Flight\_id, Staff\_id,AssignmentDate)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Assg\_id,

Flight\_id,

Staff\_id,

AssignmentDate

FROM

deleted;

END;

GO

-- Create Bookings\_Log table

CREATE TABLE Bookings\_Log (

Log\_id INT IDENTITY(1,1) PRIMARY KEY,

Operation VARCHAR(10),

Operation\_Timestamp DATETIME DEFAULT GETDATE(),

UserName VARCHAR(100),

Bk\_id VARCHAR(20),

Passenger\_id VARCHAR(10),

Tkt\_id VARCHAR(50),

PaymentMethod VARCHAR(50),

AmountPaid FLOAT,

Flight\_id VARCHAR(250),

Seat\_No VARCHAR(10),

BookingDate DATETIME,

Status VARCHAR(50)

);

GO

-- Create trigger for INSERT

CREATE TRIGGER log\_bookings\_inserts

ON Bookings

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Bookings\_Log (Operation, Operation\_Timestamp, UserName, Bk\_id, Passenger\_id, Tkt\_id, PaymentMethod, AmountPaid, Flight\_id, Seat\_No, BookingDate, Status)

SELECT

'INSERT',

GETDATE(),

SYSTEM\_USER,

Bk\_id,

Passenger\_id,

Tkt\_id,

PaymentMethod,

AmountPaid,

Flight\_id,

Seat\_No,

BookingDate,

Status

FROM

inserted;

END;

GO

-- Create trigger for UPDATE

CREATE TRIGGER log\_bookings\_updates

ON Bookings

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Bookings\_Log (Operation, Operation\_Timestamp, UserName, Bk\_id, Passenger\_id, Tkt\_id, PaymentMethod, AmountPaid, Flight\_id, Seat\_No, BookingDate, Status)

SELECT

'UPDATE',

GETDATE(),

SYSTEM\_USER,

inserted.Bk\_id,

inserted.Passenger\_id,

inserted.Tkt\_id,

inserted.PaymentMethod,

inserted.AmountPaid,

inserted.Flight\_id,

inserted.Seat\_No,

inserted.BookingDate,

inserted.Status

FROM

inserted

JOIN

deleted

ON

inserted.Bk\_id = deleted.Bk\_id;

END;

GO

-- Create trigger for DELETE

CREATE TRIGGER log\_bookings\_deletes

ON Bookings

AFTER DELETE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Bookings\_Log (Operation, Operation\_Timestamp, UserName, Bk\_id, Passenger\_id, Tkt\_id, PaymentMethod, AmountPaid, Flight\_id, Seat\_No, BookingDate, Status)

SELECT

'DELETE',

GETDATE(),

SYSTEM\_USER,

Bk\_id,

Passenger\_id,

Tkt\_id,

PaymentMethod,

AmountPaid,

Flight\_id,

Seat\_No,

BookingDate,

Status

FROM

deleted;

END;

GO