**Section 1:**  
**DDL**  
CREATE TABLE Flights (

FlightID INT PRIMARY KEY,

DepartureTime DATETIME NOT NULL,

ArrivalTime DATETIME NOT NULL,

[Status] VARCHAR(9) CHECK ([Status] in ('Departing', 'Delayed', 'Arrived', 'Cancelled')),

OriginAirportID INT,

DestinationAirportID INT,

AirlineID INT

CONSTRAINT FK\_Flights\_Airports\_Origin FOREIGN KEY(OriginAirportID) REFERENCES Airports(AirportID),

CONSTRAINT FK\_Flights\_Airports\_Destination FOREIGN KEY(DestinationAirportID) REFERENCES Airports(AirportID),

CONSTRAINT FK\_Flights\_Airlines FOREIGN KEY(AirlineID) REFERENCES Airlines(AirlineID)

)

CREATE TABLE Tickets (

TicketID INT PRIMARY KEY,

Price DECIMAL(8,2) NOT NULL,

Class VARCHAR(6) CHECK (Class in ('First', 'Second', 'Third')),

Seat VARCHAR(5) NOT NULL,

CustomerID INT,

FlightID INT

CONSTRAINT FK\_Tickets\_Customers FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID),

CONSTRAINT FK\_Tickets\_Flights FOREIGN KEY(FlightID) REFERENCES Flights(FlightID),

)

**Section 2:  
DML - 01. Data Insertion**

INSERT INTO Flights (FlightID, DepartureTime, ArrivalTime, [Status], OriginAirportID, DestinationAirportID,

AirlineID)

VALUES (1, '2016-10-13 06:00 AM', '2016-10-13 10:00 AM', 'Delayed', 1, 4, 1),

(2, '2016-10-12 12:00 PM', '2016-10-12 12:01 PM', 'Departing',1, 3, 2),

(3, '2016-10-14 03:00 PM', '2016-10-20 04:00 AM', 'Delayed', 4, 2, 4),

(4, '2016-10-12 01:24 PM', '2016-10-12 4:31 PM', 'Departing', 3, 1, 3),

(5, '2016-10-12 08:11 AM', '2016-10-12 11:22 PM', 'Departing', 4, 1, 1),

(6, '1995-06-21 12:30 PM', '1995-06-22 08:30 PM', 'Arrived', 2, 3, 5),

(7, '2016-10-12 11:34 PM', '2016-10-13 03:00 AM', 'Departing', 2, 4, 2),

(8, '2016-11-11 01:00 PM', '2016-11-12 10:00 PM', 'Delayed', 4, 3, 1),

(9, '2015-10-01 12:00 PM', '2015-12-01 01:00 AM', 'Arrived', 1, 2, 1),

(10, '2016-10-12 07:30 PM', '2016-10-13 12:30 PM', 'Departing', 2, 1, 7)

INSERT INTO Tickets (TicketID, Price, Class, Seat, CustomerID, FlightID)

VALUES (1, 3000.00, 'First', '233-A', 3, 8),

(2, 1799.90, 'Second', '123-D', 1, 1),

(3, 1200.50, 'Second', '12-Z', 2, 5),

(4, 410.68, 'Third', '45-Q', 2, 8),

(5, 560.00, 'Third', '201-R', 4, 6 ),

(6, 2100.00, 'Second', '13-T', 1, 9 ),

(7, 5500.00, 'First', '98-O', 2, 7)  
  
  
 **DML - 02. Update Flights**  
  
UPDATE Flights

SET AirlineID = 1

WHERE [Status] = 'Arrived'

**DML - 03. Update Tickets**

UPDATE [dbo].[Tickets]

SET Price = 1.5\*Price

WHERE FlightID in

(SELECT f.FlightID FROM Flights AS f

INNER JOIN Airlines AS a

ON f.AirlineID = a.AirlineID  
WHERE a.Rating = (Select Max(a2.Rating) FROM Airlines AS a2))  
  
  
  
**DML - 04. Table Creation**CREATE TABLE CustomerReviews(

ReviewID INT PRIMARY KEY,

ReviewContent VARCHAR(255) NOT NULL,

ReviewGrade INT CHECK(ReviewGrade>= 0 OR ReviewGrade <=10),

AirlineID INT,

CustomerID INT

CONSTRAINT FK\_CustomerReviews\_Airlines FOREIGN KEY(AirlineID) REFERENCES Airlines(AirlineID),

CONSTRAINT FK\_CustomerReviews\_Customers FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID)

)

CREATE TABLE CustomerBankAccounts(

AccountID INT PRIMARY KEY,

AccountNumber VARCHAR(10) NOT NULL UNIQUE,

Balance DECIMAL(10,2) NOT NULL,

CustomerID INT,

CONSTRAINT FK\_CustomerBankAccounts\_Customers FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID)

)  
  
**DML - 05. Fillin New Tables**  
  
INSERT INTO CustomerReviews (ReviewID, ReviewContent, ReviewGrade, AirlineID, CustomerID)

VALUES(1, 'Me is very happy. Me likey this airline. Me good.', 10, 1, 1),

(2, 'Ja, Ja, Ja... Ja, Gut, Gut, Ja Gut! Sehr Gut!', 10, 1, 4),

(3, 'Meh...', 5, 4, 3),

(4, 'Well Ive seen better, but Ive certainly seen a lot worse...', 7, 3, 5)

INSERT INTO CustomerBankAccounts (AccountID, AccountNumber, Balance, CustomerID )

VALUES(1,'123456790', 2569.23, 1),

(2, '18ABC23672', 14004568.23, 2),

(3, 'F0RG0100N3', 19345.20, 5)   
  
**Section 3:   
Querying - 01. Extract All Tickets**SELECT t.TicketID, t.Price, t.Class, t.Seat FROM Tickets AS t

ORDER BY t.TicketID  
 **Querying - 02. Extract All Customers**

SELECT c.CustomerID, Concat(c.FirstName, ' ', c.LastName) AS FullName, c.Gender FROM Customers AS c

ORDER BY FullName, c.CustomerID

**Querying - 03. Extract Delayed Flights**

SELECT f.FlightID, f.DepartureTime, f.ArrivalTime FROM Flights AS f

WHERE f.Status = 'Delayed'

ORDER BY f.FlightID

**Querying - 04. Top 5 Airlines**

SELECT TOP 5 a.AirlineID, a.AirlineName, a.Nationality, a.Rating FROM Airlines AS a

WHERE a.AirlineID IN (SELECT f.AirlineID FROM Flights AS f)

ORDER BY a.Rating DESC, a.AirlineID  
  
  
**Querying - 05. All Tickets Below 5000**

SELECT t.TicketID, a.AirportName AS Destination, CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName FROM Tickets

AS t

INNER JOIN Flights AS f

ON t.FlightID = f.FlightID

INNER JOIN Airports AS a

ON a.AirportID = f.DestinationAirportID

INNER JOIN Customers AS c

ON t.CustomerID = c.CustomerID

WHERE t.Price < 5000

AND t.Class = 'First'

ORDER BY TicketID

## Querying - 06. Customers From Home

SELECT c.CustomerID, CONCAT(c.FirstName, ' ', c.LastName) AS FullName, towns.TownName AS HomeTown   
 FROM Customers AS c

INNER JOIN Tickets AS t

ON t.CustomerID = c.CustomerID

INNER JOIN Flights AS f

ON t.FlightID = f.FlightID

INNER JOIN Airports AS a

ON f.OriginAirportID = a.AirportID

INNER JOIN Towns AS towns

ON a.TownID = towns.TownID

WHERE a.TownID = c.HomeTownID

AND f.[Status] = 'Departing'

ORDER BY c.CustomerID ASC

## Querying - 07. Customers who will fly

SELECT c.CustomerID, CONCAT(c.FirstName, ' ', c.LastName) AS FullName,

((SELECT YEAR(CAST ('2016' AS DATE)) - YEAR(c.DateOfBirth) AS Age)) AS Age

FROM Customers AS c

INNER JOIN Tickets AS t

ON c.CustomerID = t.CustomerID

INNER JOIN Flights AS f

ON t.FlightID = f.FlightID

WHERE f.[Status] = 'Departing'

GROUP BY c.CustomerID, CONCAT(c.FirstName, ' ', c.LastName), (YEAR(CAST ('2016' AS DATE)) - YEAR(c.DateOfBirth))

ORDER BY (YEAR(CAST ('2016' AS DATE)) - YEAR(c.DateOfBirth)) ASC, c.CustomerID ASC

## Querying - 08. Top 3 Customers Delayed

SELECT TOP 3 c.CustomerID, CONCAT(c.FirstName, ' ', c.LastName) AS FullName, t.Price AS TicketPrice, a.AirportName

AS Destination FROM Customers AS c

INNER JOIN Tickets AS t

ON c.CustomerID = t.CustomerID

INNER JOIN Flights AS f

ON t.FlightID = f.FlightID

INNER JOIN Airports AS a

ON a.AirportID = f.DestinationAirportID

WHERE f.[Status] = 'Delayed'

ORDER BY t.Price DESC, c.CustomerID ASC

**Querying - 09. Last 5 Departing Flights**

SELECT \*

FROM (

SELECT TOP 5 Flights.FlightID, Flights.DepartureTime, Flights.ArrivalTime, Origin.AirportName AS Origin, Destination.AirportName AS Destination FROM Flights

INNER JOIN Airports Origin

ON (Flights.OriginAirportID = Origin.AirportID)

INNER JOIN Airports Destination

ON (Flights.DestinationAirportID = Destination.AirportID)

WHERE STATUS = 'Departing'

ORDER BY Flights.DepartureTime DESC) LastFive

ORDER BY DepartureTime ASC, FlightID ASC

**Querying - 10. Customers Below 21**

SELECT \* FROM

(SELECT DISTINCT Customers.CustomerID, CONCAT(FirstName, ' ', LastName) AS FullName, DATEDIFF(YYYY, DateOfBirth, '2016') AS Age FROM Customers

INNER JOIN Tickets

ON Customers.CustomerID = Tickets.CustomerID

INNER JOIN Flights

ON Tickets.FlightID = Flights.FlightID

WHERE Flights.Status = 'Arrived') AS FlyingUnderaged

WHERE Age < 21

ORDER BY Age DESC, CustomerID ASC

**Querying - 11. AIrports and Passengers**

SELECT f.OriginAirportID AS AirportID, a.AirportName, COUNT(t.TicketID) AS Passengers FROM Flights AS f

INNER JOIN Tickets AS t

ON t.FlightID = f.FlightID

INNER JOIN Airports AS a

ON a.AirportID = f.OriginAirportID

WHERE f.[Status] = 'Departing'

GROUP BY f.OriginAirportID, a.AirportName

ORDER BY f.OriginAirportID

**Section 4:**

**Programmibility - 02. Ticket Purchase**

CREATE PROCEDURE usp\_PurchaseTicket

@CustomerID INT,

@FlightID INT,

@TicketPrice DECIMAL(8, 2),

@Class VARCHAR(6),

@Seat VARCHAR(5)

AS

BEGIN

DECLARE @CustomerBalance DECIMAL(10, 2);

DECLARE @TicketID INT;

BEGIN TRANSACTION

SELECT @TicketID = (SELECT TOP 1 TicketID FROM Tickets ORDER BY TicketID DESC) + 1;

IF @TicketID IS NULL

BEGIN

SELECT @TicketID = 1

END

SELECT @CustomerBalance = (SELECT Balance FROM CustomerBankAccounts

WHERE CustomerID = @CustomerID)

INSERT INTO Tickets(TicketID, Price, Class, Seat, CustomerID, FlightID)

VALUES(@TicketID, @TicketPrice, @Class, @Seat, @CustomerID, @FlightID)

UPDATE CustomerBankAccounts

SET Balance = @CustomerBalance - @TicketPrice

WHERE CustomerID = @CustomerID

IF @CustomerBalance IS NULL OR @CustomerBalance < @TicketPrice

BEGIN

ROLLBACK

RAISERROR('Insufficient bank account balance for ticket purchase.', 16, 1)

RETURN

END

COMMIT

END