DATA MANAGEMENT
PROJECT ASSIGNMENT - 2

Team Name: SKYGauger

NAME	EMAIL	SBU ID
Atharv Raotole	atharv.raotole@stonybrook.edu	116480860
Parth Chavan	parth.chavan@stonybrook.edu	116558055
Dhananjay Sharma	dhananjay.sharma@stonybrook.edu	116740841

1) A dump (listing) of all the tables with the initial Demo Data loaded into them.

Creating the database

```
CREATE DATABASE DAPROJECT;
```

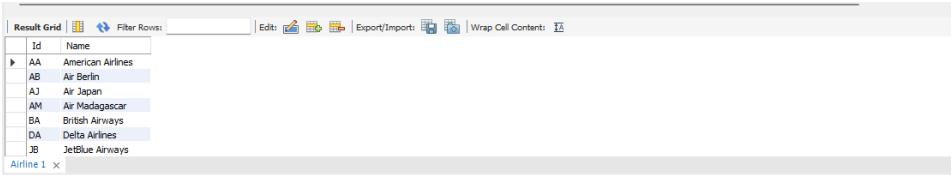
1) **Airline Table**

```
CREATE TABLE Airline (
  Id CHAR(2),
  Name VARCHAR(100) NOT NULL,
  PRIMARY KEY (Id)
);
```

Inserting data into Airline table

```
INSERT INTO Airline (Id, Name) VALUES
('AB', 'Air Berlin'),
('AJ', 'Air Japan'),
('AM', 'Air Madagascar'),
('AA', 'American Airlines'),
('BA', 'British Airways'),
('DA', 'Delta Airlines'),
('JB', 'JetBlue Airways'),
('LH', 'Lufthansa'),
('SW', 'Southwest Airlines'),
('UA', 'United Airlines');

SELECT * FROM Airline;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name
AA	American Airlines
AB	Air Berlin
AJ	Air Japan
AM	Air Madagascar
BA	British Airways
DA	Delta Airlines
JB	JetBlue Airways

Airline 1

2) **Airport Table**

```
CREATE TABLE Airport (
  Id CHAR(3),
  Name VARCHAR(100) NOT NULL,
  City VARCHAR(50) NOT NULL,
  Country VARCHAR(50) NOT NULL,
  PRIMARY KEY (Id)
);
```

Inserting data into Airport table

```
INSERT INTO Airport (Id, Name, City, Country) VALUES
('TXL', 'Berlin Tegel', 'Berlin', 'Germany'),
('ORD', 'Chicago O\'Hare International', 'Chicago', 'United States of America'),
('ATL', 'Hartsfield-Jackson Atlanta Int', 'Atlanta', 'United States of America'),
('TNR', 'Ivato International', 'Antananarivo', 'Madagascar'),
('LAX', 'Los Angeles International','Los Angeles', 'California'),
('JFK', 'John F. Kennedy International', 'New York', 'United States of America');

SELECT * FROM Airport;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Id	Name	City	Country
ATL	Hartsfield-Jackson Atlanta Int	Atlanta	United States of America
JFK	John F. Kennedy International	New York	United States of America
ORD	Chicago O'Hare International	Chicago	United States of America
TNR	Ivato International	Antananarivo	Madagascar
TXL	Berlin Tegel	Berlin	Germany
NULL	NULL	NULL	NULL

Airport 2

3) **Flight Table**

```
CREATE TABLE Flight (
  AirlineID CHAR(2),
  FlightNo INTEGER NOT NULL,
  NoOfSeats INTEGER NOT NULL,
  DaysOperating CHAR(7) NOT NULL,
  MinLengthOfStay INTEGER NOT NULL,
  MaxLengthOfStay INTEGER NOT NULL,
  isDelayed TINYINT(1), -- Changed BOOLEAN to TINYINT(1)
  PRIMARY KEY (AirlineID, FlightNo),
  FOREIGN KEY (AirlineID) REFERENCES Airline(Id),
  CHECK (NoOfSeats > 0),
  CHECK (MinLengthOfStay >= 0),
```

```
    CHECK (MaxLengthOfStay > MinLengthOfStay));
```

Inserting data into Flight table

```
INSERT INTO Flights (AirlineID, FlightNo, NoOfSeats, DaysOperating, MinLengthOfStay,
MaxLengthOfStay, isDelayed) VALUES
('AA', 101, 150, '1234567', 1, 14, 1),
('JB', 202, 180, '135', 2, 10, 0),
('DA', 303, 200, '246', 3, 15, 1),
('UA', 404, 120, '123', 1, 7, 0);
```

```
SELECT * FROM Flight;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

AirlineID	FlightNo	NoOfSeats	DaysOperating	MinLengthOfStay	MaxLengthOfStay	isDelayed
AA	101	150	1234567	1	14	1
DA	303	200	246	3	15	1
JB	202	180	135	2	10	0
UA	404	120	123	1	7	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

4) Leg Table

```
CREATE TABLE Leg (
    AirlineID CHAR(2),
    FlightNo INTEGER NOT NULL,
    LegNo INTEGER NOT NULL,
    DepAirportID CHAR(3) NOT NULL,
    ArrAirportID CHAR(3) NOT NULL,
    ArrTime DATETIME NOT NULL,
    DepTime DATETIME NOT NULL,
    PRIMARY KEY (AirlineID, FlightNo, LegNo),
    FOREIGN KEY (AirlineID, FlightNo) REFERENCES Flight(AirlineID, FlightNo),
    FOREIGN KEY (DepAirportID) REFERENCES Airport(Id),
    FOREIGN KEY (ArrAirportID) REFERENCES Airport(Id),
    CHECK (LegNo > 0)
);
```

Inserting data into Leg table

```
INSERT INTO Leg (AirlineID, FlightNo, LegNo, DepAirportID, ArrAirportID, ArrTime,
DepTime) VALUES
```

```
('AA', 111, 1, 'JFK', 'LAX', '2024-01-05 09:00:00', '2024-01-05 11:00:00');

SELECT * FROM Leg;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

AirlineID	FlightNo	LegNo	DepAirportID	ArrAirportID	ArrTime	DepTime
AA	111	1	JFK	LAX	2024-01-05 09:00:00	2024-01-05 11:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

5) **Employee Table**

```
CREATE TABLE Employee (
  SSN INTEGER,
  Name VARCHAR(100) NOT NULL,
  IsManager BOOLEAN,
  StartDate DATE NOT NULL,
  HourlyRate DECIMAL(10, 2),
  PRIMARY KEY (SSN)
);
```

Inserting data into Employee table

```
INSERT INTO Employee (SSN, Name, IsManager, StartDate, HourlyRate) VALUES
(123456789, 'John Doe', TRUE, '2024-01-01', 30.00),
(987654321, 'Jane Smith', FALSE, '2024-02-01', 25.00),
(192837465, 'Alice Brown', TRUE, '2024-03-01', 40.00);

SELECT * FROM Employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

SSN	Name	IsManager	StartDate	HourlyRate
123456789	John Doe	1	2024-01-01	30.00
192837465	Alice Brown	1	2024-03-01	40.00
987654321	Jane Smith	0	2024-02-01	25.00
NULL	NULL	NULL	NULL	NULL

Employee 5

6) **Customer Table**

```
CREATE TABLE Customers (
  AccountNo INTEGER,
  Name VARCHAR(100) NOT NULL,
  Address VARCHAR(255),
  Phone VARCHAR(15),
  Email VARCHAR(100),
  PRIMARY KEY (AccountNo)
);
```

Inserting data into Customer table

```
INSERT INTO Customers (AccountNo, Name, Address, Phone, Email) VALUES
(1, 'Company A', '123 Main St, CityA', '123-456-7890', 'companya@gmail.com'),
(2, 'Company B', '456 Elm St, CityB', '234-567-8901', 'companyb@gmail.com'),
(3, 'Company C', '789 Oak St, CityC', '345-678-9012', 'companyc@gmail.com');

SELECT * FROM Customers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

AccountNo	Name	Address	Phone	Email
1	Company A	123 Main St, CityA	123-456-7890	companya@gmail.com
2	Company B	456 Elm St, CityB	234-567-8901	companyb@gmail.com
3	Company C	789 Oak St, CityC	345-678-9012	companyc@gmail.com
NULL	NULL	NULL	NULL	NULL

7) **Reservation table**

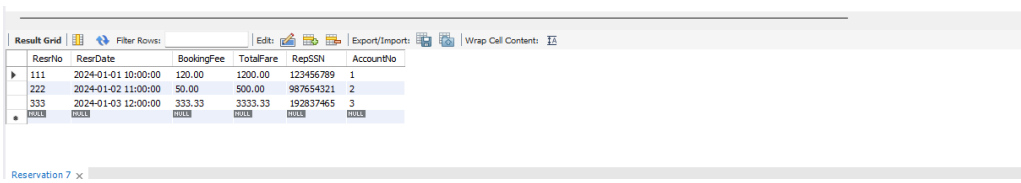
```
CREATE TABLE Reservation (
  ResrNo INTEGER,
  ResrDate DATETIME NOT NULL,
  BookingFee NUMERIC(10,2) NOT NULL,
  TotalFare NUMERIC(10,2) NOT NULL,
  RepSSN INTEGER,
  AccountNo INTEGER NOT NULL,
  PRIMARY KEY (ResrNo),
  FOREIGN KEY (RepSSN) REFERENCES Employee(SSN),
  FOREIGN KEY (AccountNo) REFERENCES Customer(AccountNo),
```

```
  CHECK (ResrNo > 0),
  CHECK (BookingFee >= 0),
  CHECK (TotalFare > BookingFee)
);
```

Inserting data in Reservation table

```
INSERT INTO Reservation (ResrNo, ResrDate, BookingFee, TotalFare, RepSSN, AccountNo)
VALUES
(111, '2024-01-01 10:00:00', 120.00, 1200.00, 123456789, 1),
(222, '2024-01-02 11:00:00', 50.00, 500.00, 987654321, 2),
(333, '2024-01-03 12:00:00', 333.33, 3333.33, 192837465, 3);

SELECT * FROM Reservation;
```



ResrNo	ResrDate	BookingFee	TotalFare	RepSSN	AccountNo
111	2024-01-01 10:00:00	120.00	1200.00	123456789	1
222	2024-01-02 11:00:00	50.00	500.00	987654321	2
333	2024-01-03 12:00:00	333.33	3333.33	192837465	3
NULL	NULL	NULL	NULL	NULL	NULL

8) **Includes table**

```
CREATE TABLE Includes (
  ResrNo INTEGER,
  AirlineID CHAR(2),
  FlightNo INTEGER,
  LegNo INTEGER,
  Date DATE NOT NULL,
  PRIMARY KEY (ResrNo, AirlineID, FlightNo, LegNo),
  FOREIGN KEY (ResrNo) REFERENCES Reservation(ResrNo),
  FOREIGN KEY (AirlineID, FlightNo, LegNo) REFERENCES Leg(AirlineID, FlightNo, LegNo)
);
```

Inserting data in Includes table

```
INSERT INTO Includes (ResrNo, AirlineID, FlightNo, LegNo, Date) VALUES
(111, 'AA', 111, 1, '2024-01-05'),
(222, 'JB', 222, 1, '2024-01-10'), -- Changed FlightNo to match the Flight table
(333, 'AM', 1337, 1, '2024-01-13');
```

```
SELECT * FROM Includes;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ResrNo	AirlineID	FlightNo	LegNo	Date
111	AA	111	1	2024-01-05
NULL	NULL	NULL	NULL	NULL

9) BidHist table

```
CREATE TABLE BidHist (
  BidID INTEGER AUTO_INCREMENT PRIMARY KEY,
  ResrNo INTEGER,
  LegNo INTEGER,
  Bid DECIMAL(10, 2),
  Accepted BOOLEAN,
  BidTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (ResrNo) REFERENCES Reservation(ResrNo),
  FOREIGN KEY (LegNo) REFERENCES Leg(LegNo)
);
```

Inserting data in BidHist table

```
INSERT INTO BidHist (ResrNo, LegNo, Bid, Accepted)
VALUES (111, 1, 400.00, TRUE);
INSERT INTO BidHist (ResrNo, LegNo, Bid, Accepted)
VALUES (111, 1, 300.00, TRUE);
INSERT INTO BidHist (ResrNo, LegNo, Bid, Accepted)
VALUES (111, 1, 350.00, FALSE);
INSERT INTO BidHist (ResrNo, LegNo, Bid, Accepted)
VALUES (111, 1, 325.00, TRUE);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

BidID	ResrNo	LegNo	Bid	Accepted	BidTime
1	111	1	400.00	1	2024-11-06 16:14:48
2	111	1	300.00	1	2024-11-06 16:14:53
3	111	1	350.00	0	2024-11-06 16:14:58
4	111	1	325.00	1	2024-11-06 16:15:02
NULL	NULL	NULL	NULL	NULL	NULL

BidHist 8 ×

TRANSACTIONS IN SECTION 3:

1) Manager-Level Transactions

Add, Edit, and Delete Information for an Employee

a) Add Employee

Definition: Add a new employee to the database.

Parameters: SSN (INTEGER), IsManager (BOOLEAN), StartDate (DATE), HourlyRate (NUMERIC)

SQL Statement:

```
BEGIN TRANSACTION;
INSERT INTO Employee (SSN, IsManager, StartDate, HourlyRate)
VALUES (?, ?, ?, ?);
COMMIT;
```

Execution Example:

```
-- Execute with: SSN=1234567890, IsManager=TRUE, StartDate='2024-01-01', HourlyRate=30.00
BEGIN TRANSACTION;
INSERT INTO Employee (SSN, IsManager, StartDate, HourlyRate)
VALUES (1234567890, TRUE, '2024-01-01', 30.00);
COMMIT;
-- Output: Employee added successfully.
```

Result Grid | Filter Rows: | Edit: | Export/Impo

SSN	Name	IsManager	StartDate	HourlyRate
123456789	John Doe	1	2024-01-01	30.00
192837465	Alice Brown	1	2024-03-01	40.00
987654321	Jane Smith	0	2024-02-01	25.00
1234567890	Smith Green	1	2024-01-01	30.00
NULL	NULL	NULL	NULL	NULL

b) Edit Employee

Definition: Update an employee's hourly rate.

Parameters: SSN (INTEGER), HourlyRate (NUMERIC)

SQL Statement:

```
BEGIN TRANSACTION;
UPDATE Employee
SET HourlyRate = ?
WHERE SSN = ?;
COMMIT;
```

Execution Example:

```
-- Execute with: SSN=1234567890, HourlyRate=35.00
BEGIN TRANSACTION;
UPDATE Employee
SET HourlyRate = 35.00
WHERE SSN = 1234567890;
COMMIT;
-- Output: Employee updated successfully.
```

Result Grid | Filter Rows: | Edit: | Export/Imp

SSN	Name	IsManager	StartDate	HourlyRate
123456789	John Doe	1	2024-01-01	30.00
192837465	Alice Brown	1	2024-03-01	40.00
987654321	192837465	0	2024-02-01	25.00
1234567890	Smith Green	1	2024-01-01	35.00
NULL	NULL	NULL	NULL	NULL

c) Delete Employee

Definition: Remove an employee from the database.

Parameters: SSN (INTEGER)

SQL Statement:

```
BEGIN TRANSACTION;
DELETE FROM Employee
WHERE SSN = ?;
COMMIT;
```

Execution Example:

```
-- Execute with: SSN=1234567890
BEGIN TRANSACTION;
```

```
DELETE FROM Employee
WHERE SSN = 1234567890;
COMMIT;
-- Output: Employee deleted successfully.
```

Result Grid | Filter Rows: | Edit: | Export

SSN	Name	IsManager	StartDate	HourlyRate
123456789	John Doe	1	2024-01-01	30.00
192837465	Alice Brown	1	2024-03-01	40.00
987654321	Jane Smith	0	2024-02-01	25.00
NULL	NULL	NULL	NULL	NULL

d) Obtain a Sales Report for a Particular Month

Definition: Generate a sales report for a specified month and year.

Parameters: Month (INTEGER), Year (INTEGER)

SQL Statement:

```
SELECT ResrNo, SUM(TotalFare) AS TotalSales
FROM Reservation
WHERE MONTH(ResrDate) = ? AND YEAR(ResrDate) = ?
GROUP BY ResrNo;
```

Execution Example:

```
-- Execute with: Month=1, Year=2024
SELECT ResrNo, SUM(TotalFare) AS TotalSales
FROM Reservation
WHERE MONTH(ResrDate) = 1 AND YEAR(ResrDate) = 2024
GROUP BY ResrNo;
-- Output: List of reservations with total sales for January.
```

Result Grid | Filter Rows:

ResrNo	TotalSales
111	1200.00
222	500.00
333	3333.33

e) Produce a comprehensive listing of all flights

Definition: Generate a detailed listing of all flights in the database, including flight details such as FlightNumber, Departure, Arrival, DepartureTime, ArrivalTime, Status, etc.

Parameters: Not Required

SQL Statement:

```
SELECT f.FlightNo, l.DepAirportID, l.ArrAirportID, l.DepTime, l.ArrTime
FROM Flight f JOIN Leg l ON f.FlightNo = l.FlightNo;
COMMIT;
-- Output: detailed listing of all flights in the database
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FlightNo	DepAirportID	ArrAirportID	DepTime	ArrTime
111	JFK	LAX	2024-01-05 11:00:00	2024-01-05 09:00:00

f) Produce a list of reservations by flight number or by customer name

Definition: produce a list of reservations by flight number or by customer name using the provided database structure, we need to join the Reservation, Includes, Customer, and Leg tables.

Parameters: FlightNo , CustomerName

SQL Statement:

```
SELECT r.ResrNo, r.ResrDate, r.TotalFare, c.Name AS CustomerName, i.AirlineID, i.FlightNo,
l.DepAirportID, l.ArrAirportID, l.DepTime, l.ArrTime
FROM Reservation r
JOIN Customer c ON r.AccountNo = c.AccountNo
JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Leg l ON i.AirlineID = l.AirlineID AND i.FlightNo = l.FlightNo AND i.LegNo = l.LegNo
WHERE i.FlightNo = ? OR c.Name = ?;
```

Execution Example:

```
-- Execute with: FlightNo=111, Name='Company A'
SELECT r.ResrNo, r.ResrDate, r.TotalFare, c.Name AS CustomerName, i.AirlineID, i.FlightNo,
l.DepAirportID, l.ArrAirportID, l.DepTime, l.ArrTime
```

```
FROM Reservation r
JOIN Customer c ON r.AccountNo = c.AccountNo
JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Leg l ON i.AirlineID = l.AirlineID AND i.FlightNo = l.FlightNo AND i.LegNo = l.LegNo
WHERE i.FlightNo = 111 OR c.Name = 'Company A';
-- Output: a list of reservations by flight number = 111  or by customer name = 'Company A'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ResrNo	ResrDate	TotalFare	CustomerName	AirlineID	FlightNo	DepAirportID	ArrAirportID	DepTime	ArrTime
111	2024-01-01 10:00:00	1200.00	Company A	AA	111	JFK	LAX	2024-01-05 11:00:00	2024-01-05 09:00:00

g) Produce a summary listing of revenue generated by a particular flight, destination city, or customer

Definition: produce a summary listing of revenue generated by a particular flight, destination city, or customer

1. **Revenue Generated by a Particular Flight**

   Parameters: FlightNo

   SQL Statement:

```
SELECT i.AirlineID, i.FlightNo, SUM(r.TotalFare) AS TotalRevenue
FROM Reservation r JOIN Includes i ON r.ResrNo = i.ResrNo
GROUP BY i.AirlineID, i.FlightNo
HAVING i.FlightNo = ?;
```

   Execution Example:

```
-- Execute with: FlightNo=111
SELECT i.AirlineID, i.FlightNo, SUM(r.TotalFare) AS TotalRevenue
FROM Reservation r JOIN Includes i ON r.ResrNo = i.ResrNo
GROUP BY i.AirlineID, i.FlightNo
HAVING i.FlightNo = 111;
-- Output: produce a summary listing of revenue generated by a particular flight = 111
```

Result Grid | Filter Rows:

AirlineID	FlightNo	TotalRevenue
AA	111	1200.00

2. **Revenue Generated by a Destination City**
   Parameters: FlightNo

   SQL Statement:

```
SELECT a.City AS DestinationCity, SUM(r.TotalFare) AS TotalRevenue
FROM Reservation r JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Leg l ON i.AirlineID = l.AirlineID AND i.FlightNo = l.FlightNo
AND i.LegNo = l.LegNo
JOIN Airport a ON l.ArrAirportID = a.Id
GROUP BY a.City
HAVING a.City = ?;
```

   Execution Example:

```
-- Execute with: city='Los Angeles'
SELECT a.City AS DestinationCity, SUM(r.TotalFare) AS TotalRevenue
FROM Reservation r JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Leg l ON i.AirlineID = l.AirlineID AND i.FlightNo = l.FlightNo
AND i.LegNo = l.LegNo
JOIN Airport a ON l.ArrAirportID = a.Id
GROUP BY a.City
HAVING a.City = 'Los Angeles';
-- Output: produce a summary listing of revenue generated by a particular destination city = 'Los Angeles'
```

Result Grid | Filter Rows:

DestinationCity	TotalRevenue
Los Angeles	1200.00

3. **Revenue Generated by a Customer**
   Parameters: Customer Name

   SQL Statement:

```
SELECT c.Name AS CustomerName, SUM(r.TotalFare) AS TotalRevenue
FROM Reservation r JOIN Customer c ON r.AccountNo = c.AccountNo
GROUP BY c.Name
HAVING c.Name = ?;
```

Execution Example:

```
-- Execute with: Name='Company A'
SELECT c.Name AS CustomerName, SUM(r.TotalFare) AS TotalRevenue
FROM Reservation r JOIN Customer c ON r.AccountNo = c.AccountNo
GROUP BY c.Name
HAVING c.Name = 'Company A';
-- Output: produce a summary listing of revenue generated by a particular Customer = 'Company A'
```

Result Grid | Filter Rows:

CustomerName	TotalRevenue
Company A	1200.00

h) Determine which customer representative generated most total revenue

Definition: determine which customer representative generated the most total revenue

Parameters: Not Required

SQL Statement:

```
SELECT e.Name AS RepresentativeName, e.SSN, SUM(r.TotalFare) AS TotalRevenue
FROM Employee e
JOIN Reservation r ON e.SSN = r.RepSSN
WHERE e.IsManager = FALSE
GROUP BY e.Name, e.SSN
ORDER BY TotalRevenue DESC
LIMIT 1;
-- Output: customer representative generated the most total revenue
```

Result Grid | Filter Rows: | Export: | Wrap C

RepresentativeName	SSN	TotalRevenue
Jane Smith	987654321	500.00

i) Determine which customer generated most total revenue

Definition: Determine which customer generated most total revenue

Parameters: Not Required

SQL Statement:

```
SELECT c.Name AS CustomerName, c.AccountNo, SUM(r.TotalFare) AS TotalRevenue
FROM Customer c
JOIN Reservation r ON c.AccountNo = r.AccountNo
GROUP BY c.Name, c.AccountNo
ORDER BY TotalRevenue DESC
 LIMIT 1;
-- Output: customer  generated the most total revenue
```

Result Grid | Filter Rows: | Expo

CustomerName	AccountNo	TotalRevenue
Company C	3	3333.33

j) Produce a list of most active flights

Definition: produce a list of the most active flights, which could mean the flights that have the highest number of reservations.

Parameters: Not Required

SQL Statement:

```
SELECT i.AirlineID, i.FlightNo, COUNT(i.ResrNo) AS ReservationCount
FROM Includes i JOIN Reservation r ON i.ResrNo = r.ResrNo
GROUP BY i.AirlineID, i.FlightNo
ORDER BY ReservationCount DESC;

-- Output: list of most active flights
```

Result Grid | Filter Rows: | Exp

AirlineID	FlightNo	ReservationCount
AA	111	1

k) Produce a list of all customers who have seats reserved on a given flight

Definition: produce a list of all customers who have seats reserved on a given flight

Parameters: AirLineId, FlightNo

SQL Statement:

```
SELECT c.Name AS CustomerName, r.ResrNo AS ReservationNumber, f.AirlineID, f.FlightNo
FROM Reservation r
```

```
JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Customer c ON r.AccountNo = c.AccountNo
JOIN Flight f ON i.AirlineID = f.AirlineID AND i.FlightNo = f.FlightNo
WHERE f.AirlineID = ? AND f.FlightNo = ?;
```

Execution Example:

```
-- Execute with: AirlineID = 'AA' and FlightNo = 111
SELECT c.Name AS CustomerName, r.ResrNo AS ReservationNumber, f.AirlineID, f.FlightNo
FROM Reservation r
JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Customer c ON r.AccountNo = c.AccountNo
JOIN Flight f ON i.AirlineID = f.AirlineID AND i.FlightNo = f.FlightNo
WHERE f.AirlineID = 'AA' AND f.FlightNo = 111;
-- Output: list of all customers who have seats reserved on a given flight
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerName	ReservationNumber	AirlineID	FlightNo
Company A	111	AA	111

l) Produce a list of all flights for a given airport

Definition: To produce a list of all flights for a given airport, we need to join the Leg and Flight tables

Parameters: DepAirportID

SQL Statement:

```
SELECT f.AirlineID, f.FlightNo, l.DepAirportID, l.ArrAirportID, l.DepTime, l.ArrTime
FROM Leg l
JOIN Flight f ON l.AirlineID = f.AirlineID AND l.FlightNo = f.FlightNo
WHERE l.DepAirportID = ?';
```

Execution Example:

```
-- Execute with: DepAirportID = 'JFK'
SELECT f.AirlineID, f.FlightNo, l.DepAirportID, l.ArrAirportID, l.DepTime, l.ArrTime
FROM Leg l
JOIN Flight f ON l.AirlineID = f.AirlineID AND l.FlightNo = f.FlightNo
WHERE l.DepAirportID = 'JFK';
```

-- Output: list of all flights for a given airport

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AirlineID	FlightNo	DepAirportID	ArrAirportID	DepTime	ArrTime
AA	111	JFK	LAX	2024-01-05 11:00:00	2024-01-05 09:00:00

m) Produce a list of all flights whose arrival and departure times are on-time/delayed

Definition:

This query generates a report of all flights, indicating whether each flight is on-time or delayed. It leverages the isDelayed field in the Flights table to classify flights by their status.

Parameter: AirlineID, FlightNo, isDelayed

SQL Statement:

```
SELECT
  AirlineID,
  FlightNo,
  NoOfSeats,
  DaysOperating,
  MinLengthOfStay,
  MaxLengthOfStay,
  CASE
    WHEN isDelayed = 1 THEN 'Delayed'
    ELSE 'On-Time'
  END AS Status
FROM
  Flights
ORDER BY
  AirlineID, FlightNo;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AirlineID	FlightNo	NoOfSeats	DaysOperating	MinLengthOfStay	MaxLengthOfStay	Status
AA	101	150	1234567	1	14	Delayed
DA	303	200	246	3	15	Delayed
JB	202	180	135	2	10	On-Time
UA	404	120	123	1	7	On-Time

2) Customer-Representative-Level Transactions

a) Record a Reservation

Definition: Record a new reservation in the system.

Parameters: ResrNo (INTEGER), ResrDate (DATETIME), BookingFee (NUMERIC), TotalFare (NUMERIC), RepSSN (INTEGER), AccountNo (INTEGER)

SQL Statement:

```
BEGIN TRANSACTION;
INSERT INTO Reservation (ResrNo, ResrDate, BookingFee, TotalFare, RepSSN, AccountNo)
VALUES (?, ?, ?, ?, ?, ?);
COMMIT;
```

Execution Example:

```
-- Execute with: ResrNo=444, ResrDate='2024-01-15', BookingFee=50.00, TotalFare=500.00,
RepSSN=987654321, AccountNo=2
BEGIN TRANSACTION;
INSERT INTO Reservation (ResrNo, ResrDate, BookingFee, TotalFare, RepSSN, AccountNo)
VALUES (444, '2024-01-15', 50.00, 500.00, 987654321, 2);
COMMIT;
-- Output: Reservation recorded successfully.
```

ResrNo	ResrDate	BookingFee	TotalFare	RepSSN	AccountNo
111	2024-01-01 10:00:00	120.00	1200.00	123456789	1
222	2024-01-02 11:00:00	50.00	500.00	987654321	2
333	2024-01-03 12:00:00	333.33	3333.33	192837465	3
444	2024-01-15 00:00:00	50.00	500.00	987654321	2
NULL	NULL	NULL	NULL	NULL	NULL

b) Add, Edit and Delete information for a customer

1. **Add Information for a Customer**

   Parameters: AccountNo, Name

   SQL Statement:

```
INSERT INTO Customer (AccountNo, Name)
VALUES (?,?);
```

Execution Example:

```
-- Execute with: AccountNo = 4, Name = Company D
BEGIN TRANSACTION;
INSERT INTO Customer (AccountNo, Name)
VALUES (4, 'Company D');
-- Output: Customer Information recorded successfully.
```

Result Grid | Filter Rows:

AccountNo	Name
1	Company A
2	Company B
3	Company C
4	Company D
NULL	NULL

2. Edit Information for a Customer

Parameters: AccountNo, Name

SQL Statement:

```
UPDATE Customer
SET Name = ?
WHERE AccountNo = ?;
```

Execution Example:

```
-- Execute with: AccountNo = 4, Name = Company E
BEGIN TRANSACTION;
UPDATE Customer
SET Name = 'Company E'
WHERE AccountNo = 4;
-- Output: Customer Information Upadated successfully.
```

Result Grid | Filter Rows:

AccountNo	Name
1	Company A
2	Company B
3	Company C
4	Company E
NULL	NULL

3. Delete Information for a Customer

Parameters: AccountNo

SQL Statement:

```
DELETE FROM Customer
WHERE AccountNo = ?;
```

Execution Example:

```
-- Execute with: AccountNo = 4
BEGIN TRANSACTION;
DELETE FROM Customer
WHERE AccountNo = 4;
-- Output: Customer Information deleted successfully.
```

Result Grid | Filter Rows:

AccountNo	Name
1	Company A
2	Company B
3	Company C
NULL	NULL

c) Produce customer mailing lists

Parameters: Name, Address, Phone, Email

SQL Statement:

```
SELECT
    Name AS CustomerName,
    Address AS MailingAddress,
    Phone AS ContactNumber,
    Email AS EmailAddress
FROM
    Customers
ORDER BY
    Name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerName	MailingAddress	ContactNumber	EmailAddress
Company A	123 Main St, CityA	123-456-7890	companya@gmail.com
Company B	456 Elm St, CityB	234-567-8901	companyb@gmail.com
Company C	789 Oak St, CityC	345-678-9012	companyc@gmail.com

d) Produce a list of flight suggestions for a given customer (based on that customer's past reservations)

Parameters: AccountNo

SQL Statement:

```
SELECT DISTINCT f.AirlineID, f.FlightNo, a.Name AS AirlineName, l.DepAirportID AS
DepartureAirport, l.ArrAirportID AS ArrivalAirport
FROM Reservation r
JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Flight f ON i.AirlineID = f.AirlineID AND i.FlightNo = f.FlightNo
JOIN Leg l ON f.AirlineID = l.AirlineID AND f.FlightNo = l.FlightNo
JOIN Airline a ON f.AirlineID = a.Id
WHERE r.AccountNo = ?
ORDER BY f.AirlineID, f.FlightNo;
```

Execution Example:

```
-- Execute with: AccountNo = 1
BEGIN TRANSACTION;
SELECT DISTINCT f.AirlineID, f.FlightNo, a.Name AS AirlineName, l.DepAirportID AS
DepartureAirport, l.ArrAirportID AS ArrivalAirport
FROM Reservation r
JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Flight f ON i.AirlineID = f.AirlineID AND i.FlightNo = f.FlightNo
JOIN Leg l ON f.AirlineID = l.AirlineID AND f.FlightNo = l.FlightNo
JOIN Airline a ON f.AirlineID = a.Id
WHERE r.AccountNo = 1
ORDER BY f.AirlineID, f.FlightNo;
-- Output: list of flight suggestions for a given customer
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AirlineID	FlightNo	AirlineName	DepartureAirport	ArrivalAirport
AA	111	American Airlines	JFK	LAX

3) Customer-Level Transactions

a) Cancel an Existing Reservation

Definition: Cancel a reservation by its number.

Parameters: ResrNo (INTEGER)

SQL Statement:

```
BEGIN TRANSACTION;
DELETE FROM Reservation
WHERE ResrNo = ?;
COMMIT;
```

Execution Example:

```
-- Execute with: ResrNo=444
BEGIN TRANSACTION;
DELETE FROM Reservation
WHERE ResrNo = 444;
COMMIT;
-- Output: Reservation canceled successfully.
```

b) A customer's current reservations

Definition: retrieve a customer's current reservations from the database.

Parameters: AccountNo

SQL Statement:

```
BEGIN TRANSACTION;
SELECT ResrNo, ResrDate, TotalFare, BookingFee, RepSSN, AccountNo
FROM Reservation
WHERE AccountNo = ?;
```

```
COMMIT;
```

Execution Example:

```
-- Execute with: AccountNo = 1
SELECT ResrNo, ResrDate, TotalFare, BookingFee, RepSSN, AccountNo
FROM Reservation
WHERE AccountNo = 1;
-- Output: customer's current reservations
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ResrNo	ResrDate	TotalFare	BookingFee	RepSSN	AccountNo
111	2024-01-01 10:00:00	1200.00	120.00	123456789	1
NULL	NULL	NULL	NULL	NULL	NULL

c) Travel itinerary for a given reservation

Definition: to retrieve a **travel itinerary** for a given reservation, we need to combine information from the Reservation, Includes, Leg, Flight, and Airport tables

Parameters: ResrNo

SQL Statement:

```
BEGIN TRANSACTION;
SELECT r.ResrNo, r.ResrDate, i.Date AS FlightDate, f.AirlineID, f.FlightNo, a1.Name AS DepartureAirport, a2.Name AS ArrivalAirport, l.DepTime, l.ArrTime
FROM Reservation r JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Leg l ON i.AirlineID = l.AirlineID AND i.FlightNo = l.FlightNo AND i.LegNo = l.LegNo
JOIN Flight f ON i.AirlineID = f.AirlineID AND i.FlightNo = f.FlightNo
JOIN Airport a1 ON l.DepAirportID = a1.Id JOIN Airport a2 ON l.ArrAirportID = a2.Id
WHERE r.ResrNo = ?;
```

Execution Example:

```
-- Execute with: ResrNo = 111
SELECT r.ResrNo, r.ResrDate, i.Date AS FlightDate, f.AirlineID, f.FlightNo, a1.Name AS DepartureAirport, a2.Name AS ArrivalAirport, l.DepTime, l.ArrTime
```

```
FROM Reservation r JOIN Includes i ON r.ResrNo = i.ResrNo
JOIN Leg l ON i.AirlineID = l.AirlineID AND i.FlightNo = l.FlightNo AND i.LegNo = l.LegNo
JOIN Flight f ON i.AirlineID = f.AirlineID AND i.FlightNo = f.FlightNo
JOIN Airport a1 ON l.DepAirportID = a1.Id JOIN Airport a2 ON l.ArrAirportID = a2.Id
WHERE r.ResrNo = 111;
-- Output: Travel itinerary for a given reservation
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ResrNo	ResrDate	FlightDate	AirlineID	FlightNo	DepartureAirport	ArrivalAirport	DepTime	ArrTime
111	2024-01-01 10:00:00	2024-01-05	AA	111	John F. Kennedy International	Los Angeles International	2024-01-05 11:00:00	2024-01-05 09:00:00

d) A customer's current bid on a given reverse auction

Definition: To retrieve a customer's current bid on a given reverse auction, we can reference the **BidHist** table where each customer's bid is stored.

Parameters: ResrNo, LegNo

SQL Statement:

```
SELECT ResrNo, LegNo, Bid, Accepted, BidTime
FROM BidHist
WHERE ResrNo = ? AND LegNo = ?
ORDER BY BidTime DESC
LIMIT 1;
```

Execution Example:

```
-- Execute with: ResrNo = 111, LegNo = 1
SELECT ResrNo, LegNo, Bid, Accepted, BidTime
FROM BidHist
WHERE ResrNo = 111 AND LegNo = 1
ORDER BY BidTime DESC
LIMIT 1;
-- Output: customer's current bid on a given reverse auction
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ResrNo	LegNo	Bid	Accepted	BidTime
111	1	400.00	1	2024-11-06 03:45:40

e) The bid history for a given reverse auction

Definition: To retrieve a customer's bid History on a given reverse auction, we can reference the **BidHist** table where each customer's bid is stored.

Parameters: ResrNo, LegNo

SQL Statement:

```
SELECT ResrNo, LegNo, Bid, Accepted, BidTime
FROM BidHistory
WHERE ResrNo =? AND LegNo =?
ORDER BY BidTime DESC;
```

Execution Example:

```
-- Execute with: ResrNo = 111, LegNo = 1
SELECT ResrNo, LegNo, Bid, Accepted, BidTime
FROM BidHistory
WHERE ResrNo = 111 AND LegNo = 1
ORDER BY BidTime DESC;
-- Output: customer's  bid history on a given reverse auction
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ResrNo	LegNo	Bid	Accepted	BidTime
111	1	400.00	1	2024-11-06 03:45:40
111	1	300.00	1	2024-11-06 03:45:40
111	1	350.00	0	2024-11-06 03:45:40
111	1	325.00	1	2024-11-06 03:45:40

f) A history of all current and past reservations a customer has made

Definition: To retrieve the history of all current and past reservations made by a customer, we can query the Reservation table and join it with the Customer table to filter the results based on a specific customer (identified by their AccountNo). This will show all reservations made by the customer, whether current or past.

Parameters: AccountNo

SQL Statement:

```
SELECT R.ResrNo, R.ResrDate, R.TotalFare, R.BookingFee
FROM Reservation R JOIN Customer C ON R.AccountNo = C.AccountNo
WHERE C.AccountNo = ?
ORDER BY R.ResrDate DESC;
```

Execution Example:

```
-- Execute with: AccountNo = 1
SELECT R.ResrNo, R.ResrDate, R.TotalFare, R.BookingFee
FROM Reservation R JOIN Customer C ON R.AccountNo = C.AccountNo
WHERE C.AccountNo = 1
ORDER BY R.ResrDate DESC;
-- Output: history of all current and past reservations a customer
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ResrNo	ResrDate	TotalFare	BookingFee
111	2024-01-01 10:00:00	1200.00	120.00

111

g) Best-Seller list of flights

Definition: To produce a "Best-Seller list of flights," we would likely be looking for the flights with the highest total revenue generated. This can be derived by summing up the total fare for each flight, based on the reservations that have been made for that flight.

Parameters: Not Required

SQL Statement:

```
SELECT L.AirlineID, L.FlightNo, SUM(R.TotalFare) AS TotalRevenue
FROM Reservation R
JOIN Includes I ON R.ResrNo = I.ResrNo
JOIN Leg L ON I.AirlineID = L.AirlineID AND I.FlightNo = L.FlightNo AND I.LegNo = L.LegNo
GROUP BY L.AirlineID, L.FlightNo
ORDER BY TotalRevenue DESC;
-- Output: Best-Seller list of flights
```

Result Grid | Filter Rows: | Ex

AirlineID	FlightNo	TotalRevenue
AA	111	1200.00

h) Personalized flight suggestion list

Definition: To create a **personalized flight suggestion list** for a customer, we would typically base the suggestions on their past reservations or preferences. This can be achieved by finding patterns in their past flight bookings, such as frequent destinations, preferred airlines, or flight times.

Parameters: AccountNo

SQL Statement:

```
SELECT DISTINCT F.AirlineID, F.FlightNo, A.Name AS AirlineName, COUNT(R.ResrNo) AS
NumberOfBookings
FROM Reservation R
JOIN Includes I ON R.ResrNo = I.ResrNo
JOIN Flight F ON I.AirlineID = F.AirlineID AND I.FlightNo = F.FlightNo
JOIN Airline A ON F.AirlineID = A.Id
WHERE R.AccountNo = ?
GROUP BY F.AirlineID, F.FlightNo, A.Name
ORDER BY NumberOfBookings DESC;
```

Execution Example:

```
-- Execute with: AccountNo = 1
SELECT DISTINCT F.AirlineID, F.FlightNo, A.Name AS AirlineName, COUNT(R.ResrNo) AS
NumberOfBookings
FROM Reservation R
JOIN Includes I ON R.ResrNo = I.ResrNo
JOIN Flight F ON I.AirlineID = F.AirlineID AND I.FlightNo = F.FlightNo
JOIN Airline A ON F.AirlineID = A.Id
WHERE R.AccountNo = 1
GROUP BY F.AirlineID, F.FlightNo, A.Name
ORDER BY NumberOfBookings DESC;
-- Output: Personalized flight suggestion list
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

	AirlineID	FlightNo	AirlineName	NumberOfBookings
▶	AA	111	American Airlines	1

CONCLUSION:

This assignment creates a complete travel reservation system, handling important tasks for managers, representatives, and customers. The SQL transactions allow for managing employees, reservations, and customer information smoothly. With demo data and clear functions, it supports easy use and efficient service for all users.