Project 2 for CSE581

Wei Liu

The revise/adjustments for project 1[1]:

1. Handle/correct all feedbacks from Dr.Palider.
    1). Employee report to linked to itself;
    2). Gate/terminal/airport names was added;
    3). Route linked to airports;
    4). Add junction table for AirplaneModels with EntertainmentsOptions (many to many relationship);
    5). Add look-up tables as needed.
2. Improvement.
    1). Drop several not-necessary look-up table.
    2). Arrange the places of tables, make sure no lines-crossing.
    3). Change some the nullability of the items in tables.
    4). Adjust the CustomerTransaction and Costs tables to be more logical.

The implementation of create/insertion date for all tables:

```
-- Create Table EmployeeType(Lookup table)
CREATE TABLE EmployeeType(
EmployeeTypeId      INTEGER             PRIMARY KEY  IDENTITY(1,1),
Text                VARCHAR(30)         NOT NULL
);

INSERT INTO EmployeeType(Text)
VALUES ('Manager'),
       ('Secretary'),
       ('Mechanic'),
       ('Pilot'),
       ('Attendant');

-- Create Table Employees[2][3]
CREATE TABLE Employees (
EmployeeId                  INTEGER             PRIMARY KEY  IDENTITY(1,1),
FirstName                   VARCHAR(50)         NOT NULL,
MiddleName                  VARCHAR(50),
LastName                    VARCHAR(50)         NOT NULL,
EmployeeType                INTEGER             NOT NULL     REFERENCES EmployeeType(EmployeeTypeId),
SocialSecurityNumber        VARCHAR(11)         NOT NULL
CHECK (SocialSecurityNumber LIKE '[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]'),
Phone                       VARCHAR(13)         NOT NULL,
```

[1] The revised Visio drawing was attached for reference.

[2] The Salary data type was set to DECIMAL(8,2) rather than MONEY type for better flexibility, also for improved accuracy without considering the cutoff error when multiplication/division was performed. Reference: stackover flow, MSDN.

[3] Phone was set to NOT NULL (it is necessary for contacting with the Employees in emergency).

```sql
Salary                      DECIMAL(8,2)        NOT NULL     CHECK(Salary >= 0),
DateOfBirth                 DATE                CHECK(DateOfBirth >= '1900-01-01' AND DateOfBirth <= GETDATE()),
ReportTo                    INTEGER             REFERENCES Employees (EmployeeId)
);

INSERT INTO Employees(FirstName, MiddleName, LastName, EmployeeType, SocialSecurityNumber, Phone, Salary, DateOfBirth, ReportTo)
VALUES ('Pablo', NULL,'Picasso', 1, '523-05-1256', '265-5123-5636', 150528.00, '1956-05-08', NULL),
       ('Vincent', 'van','Gogh', 2, '408-00-6528', '138-5263-0089', 105005.35, '1978-02-28', 1),
       ('Jasper', NULL,'Johns', 3, '526-25-6886', '693-5984-9205', 145235.88, '1983-02-28', 1),
       ('Claude', NULL,'Monet', 3, '845-23-9685', '115-3265-8659', 132000.18, '1990-01-16', 3),
       ('Jackson', NULL,'Pollock', 4, '415-26-8406', '189-9699-1515', 129000.06, '1972-11-11', 1),
       ('Andy', NULL,'Warhol', 4, '529-18-6863', '159-3211-0552', 115096.88, '1969-12-02', 5),
       ('Frida', NULL,'Kahlo', 5, '889-52-3697', '121-5343-0809', 11935.88, '1989-02-28', 1),
       ('Leonardo', 'da','Vinci', 5, '758-96-2015', '569-3152-0385', 105235.88, '1981-12-05', 7);

-- Create Table TransactionStatusType
CREATE TABLE TransactionStatusType(
StatusTypeId  CHAR                  PRIMARY KEY,
Text          VARCHAR(12)           NOT NULL
);

INSERT INTO TransactionStatusType(StatusTypeId, Text)
VALUES ('Y', 'Success'),
       ('N', 'Not Success');

-- Create Table ReservationStatus
CREATE TABLE ReservationStatus(
StatusId            INTEGER             PRIMARY KEY  IDENTITY(1,1),
Text                VARCHAR(10)         NOT NULL,
);

INSERT INTO ReservationStatus(Text)
VALUES ('Success'),
       ('Failed'),
       ('Pending'),
       ('Cancelled');

-- Create Table ReservationMethods
CREATE TABLE ReservationMethods(
ReservationMethodsId      INTEGER       PRIMARY KEY  IDENTITY(1,1),
Text                      VARCHAR(10)   NOT NULL,
);

INSERT INTO ReservationMethods(Text)
VALUES ('Phone'),
       ('Email'),
       ('Website');

-- Create Table Customers[4]
CREATE TABLE Customers(
CustomerId          INTEGER             PRIMARY KEY   IDENTITY(1,1),
FirstName           VARCHAR(50)         NOT NULL,
MiddleName          VARCHAR(50),
LastName            VARCHAR(50)         NOT NULL,
Email               VARCHAR(100),
DateOfBirth         DATE                NOT NULL     CHECK(DateOfBirth >= '1900-01-01' AND DateOfBirth <= GETDATE()),
FrequentFlyNumber   VARCHAR(6)
);

INSERT INTO Customers(FirstName,MiddleName,LastName,Email,DateOfBirth,FrequentFlyNumber)
VALUES ('Albert', NULL, 'Einstein', 'aeinstein@syr.edu', '1968-01-04', NULL),
       ('Isaac', NULL, 'Newton', 'isnewton@gmail.com', '1963-07-25', NULL),
       ('Galileo', NULL, 'Galilei', 'galileog@yahoo.com', '1988-01-13', NULL),
       ('Marie', NULL, 'Curie', 'mariec@hotmail.com', '1972-12-01', NULL),
       ('Charles', NULL, 'Darwin', 'cdarwin@yahoo.com', '1980-07-11', NULL),
       ('Alexander', 'Graham', 'Bell', 'abell@bell.com', '1961-09-09', NULL),
       ('Niels', NULL, 'Bohr', 'bohr85@gmail.com', '1975-06-11', NULL),
       ('Nikola', NULL, 'Tesla', 'ntesla@tesla.com', '1956-02-03', NULL),
       ('James', 'Clerk', 'Maxwell', 'maxwell@gmail.com', '1991-08-20', NULL),
       ('Michael', NULL, 'Faraday', 'mfaraday@syr.com', '1990-10-10', NULL);

-- Create Table PropulsionDetails(Lookup table)
CREATE TABLE PropulsionDetails(
PropulsionTypeId    INTEGER      PRIMARY KEY  IDENTITY(1,1),
Text                VARCHAR(20)  NOT NULL
);

INSERT INTO PropulsionDetails(Text)
VALUES ('Turbojet'),
       ('Turboprop'),
       ('Turbofan'),
       ('Ramjet'),
       ('Scramjet');

-- Create Table EntertainmentDetails (Lookup table)
CREATE TABLE EntertainmentDetails(
EntertainmentTypeId       INTEGER      PRIMARY KEY  IDENTITY(1,1),
Text                      VARCHAR(50)  NOT NULL
);
```

[4] DateOfBirth information is necessary for security season in air flights.

```sql
INSERT INTO EntertainmentDetails(Text)
VALUES ('Moving-map systems'),
       ('In-Flight music'),
       ('In-Flight Wi-Fi'),
       ('In-Flight USB Power'),
       ('In-Flight movies'),
       ('In-Flight games');

-- Create Table PlaneAvailabilityStatus (Lookup table)
CREATE TABLE PlaneAvailabilityStatus(
AvailabilityId      CHAR         PRIMARY KEY,
Text                VARCHAR(15)  NOT NULL
);


INSERT INTO PlaneAvailabilityStatus(AvailabilityId, Text)
VALUES ('Y', 'Available'),
       ('N', 'Not Available');


-- Create Table PlaneMaintenanceStatus (Lookup table)
CREATE TABLE PlaneMaintenanceStatus(
StatusId            INTEGER            PRIMARY KEY  IDENTITY(1,1),
Text                VARCHAR(30)        NOT NULL
);

INSERT INTO PlaneMaintenanceStatus(Text)
VALUES ('Completion on time'),
       ('Completion in advance'),
       ('Completion overdue'),
       ('Not completed');

-- Create Table MaintenanceLog (Lookup table)
CREATE TABLE MaintenanceLog(
MaintenanceLogId    INTEGER      PRIMARY KEY  IDENTITY(1,1),
Classification      CHAR         NOT NULL     CHECK(Classification IN ('A','B','C','D')),
Comments            VARCHAR(1000)
);

INSERT INTO MaintenanceLog(Classification, Comments)
VALUES ('D', 'Engine check and repaire'),
       ('A', 'Tire pressure check and rotation check'),
       ('B', 'Oil filter replaced'),
       ('C', 'Radar repaired'),
       ('B', 'Aircraft altimeter check and repaire'),
      ('A', 'Radio system check'),
      ('B', 'Oil supply system check'),
      ('C', 'Stabilizer maintenance');




-- Create Table PlaneMaintenances
CREATE TABLE PlaneMaintenances(
MaintenanceId       INTEGER      PRIMARY KEY  IDENTITY(1,1),
AirplaneId          INTEGER      NOT NULL     REFERENCES  Airplanes(AirplaneId),
MaintenanceLogId    INTEGER      NOT NULL     REFERENCES  MaintenanceLog(MaintenanceLogId),
StartTime           DATETIME     NOT NULL     CHECK(StartTime >= '1900-01-01' AND StartTime <= GETDATE()),
FinishTime          DATETIME     NOT NULL     CHECK(FinishTime >= '1900-01-01' AND FinishTime <= GETDATE()),
Status              INTEGER      NOT NULL     REFERENCES  PlaneMaintenanceStatus(StatusId)
);

INSERT INTO PlaneMaintenances(AirplaneId, MaintenanceLogId, StartTime, FinishTime, Status)
VALUES (1, 1, '2013-07-01T08:00:00', '2013-07-13T12:30:00', 1),
      (1, 2, '2013-07-03T11:35:00', '2013-07-04T21:05:00', 2),
      (2, 3, '2014-08-13T07:00:00', '2014-08-14T13:30:00', 3),
      (3, 4, '2015-01-25T20:30:00', '2015-01-26T10:15:00', 1),
      (4, 5, '2015-07-18T15:15:00', '2015-07-19T18:00:00', 1),
      (4, 6, '2016-01-03T09:30:00', '2016-01-03T15:35:00', 3),
      (5, 7, '2016-02-28T07:00:00', '2016-03-01T17:40:00', 1);

-- Create Table MaintenanceRecords
CREATE TABLE MaintenanceRecords(
EmployeeId          INTEGER      REFERENCES  Employees(EmployeeId),
MaintenanceId       INTEGER      REFERENCES  PlaneMaintenances(MaintenanceId),
PRIMARY KEY (EmployeeId, MaintenanceId)
);

INSERT INTO MaintenanceRecords(EmployeeId, MaintenanceId)
VALUES (3, 1),
      (3, 2),
      (3, 3),
      (3, 5),
      (4, 1),
      (4, 2),
      (4, 4),
      (4, 6);

-- Create Table AirplaneModels
CREATE TABLE AirplaneModels(
AirplaneModelId           INTEGER            PRIMARY KEY  IDENTITY(1,1),
```

```sql
AirplaneModelName          VARCHAR(20)         NOT NULL,
PropulsionType             INTEGER             NOT NULL     REFERENCES PropulsionDetails(PropulsionTypeId),
NumberOfPilots             INTEGER             NOT NULL     CHECK(NumberOfPilots >= 1),
NumberOfAttendants         INTEGER             NOT NULL     CHECK(NumberOfAttendants >= 1),
FlyRange                   INTEGER             NOT NULL     CHECK(FlyRange > 0),
TotalSeats                 INTEGER             NOT NULL     CHECK(TotalSeats >= 1),
);

INSERT INTO AirplaneModels(AirplaneModelName, PropulsionType, NumberOfPilots, NumberOfAttendants, FlyRange, TotalSeats)
VALUES       ('Airbus a380', 1, 2, 3, 5360, 100),
             ('Airbus a319', 2, 2, 2, 4350, 80),
             ('Boeing 787', 1, 1, 2, 5610, 95),
             ('Boeing 737', 2, 2, 3, 5510, 100),
             ('Bombardier Q200', 3, 1, 2, 3615, 30),
             ('Bombardier CRJ200', 2, 2, 2, 2360, 20),
             ('Ilyushin il-76', 3, 2, 3, 3360, 80),
             ('Comac C919', 3, 2, 2, 4360, 70);

--Create table PlaneEntertainmentOptions
CREATE TABLE PlaneEntertainmentOptions(
AirplaneModelId            INTEGER      REFERENCES   AirplaneModels(AirplaneModelId),
EntertainmentTypeID        INTEGER      REFERENCES   EntertainmentDetails(EntertainmentTypeId),
PRIMARY KEY (AirplaneModelId, EntertainmentTypeID)
);

INSERT INTO PlaneEntertainmentOptions(AirplaneModelId, EntertainmentTypeID)
VALUES (1, 1),
       (1, 2),
       (1, 5),
       (2, 1),
       (2, 3),
       (3, 1),
       (3, 4),
       (4, 5),
       (4, 3),
       (5, 2),
       (6, 1),
       (7, 1),
       (8, 2),
       (9, 3);

-- Create Table Airports
CREATE TABLE Airports(
AirportId            VARCHAR(3)          PRIMARY KEY,
AirportName          VARCHAR(300)        NOT NULL,
HangarCapacity       INTEGER             NOT NULL        CHECK(HangarCapacity >= 0)
```

```
);

INSERT INTO Airports(AirportId, AirportName, HangarCapacity)
VALUES ('ORD', 'Chicago O'' Hare International Airport', 10),
       ('SYR', 'Syracuse Hancock International Airport', 3),
       ('JFK', 'John F. Kennedy International Airport', 12),
       ('SAN', 'San Diego International Airport', 8),
       ('SEA', 'Seattle-Tacoma International Airport', 6),
       ('COS', 'Colorado Springs Airport', 2),
       ('IAH', 'Houston George Bush Intercontinental Airport', 10),
       ('MIA', 'Miami International Airport', 12),
       ('ATL', 'Hartsfield-Jackson Atlanta International Airport', 15),
       ('HNL', 'Honolulu International Airport', 0);


-- Create Table Airplanes
CREATE TABLE Airplanes(
AirplaneId          INTEGER      PRIMARY KEY         IDENTITY(1,1),
AirplaneModel       INTEGER      NOT NULL            REFERENCES AirplaneModels(AirplaneModelId),
Availability        CHAR         NOT NULL            REFERENCES PlaneAvailabilityStatus(AvailabilityId)
CHECK (Availability IN ('Y', 'N')),
CurrentLocation     VARCHAR(3)   REFERENCES Airports(AirportId),
BuiltDate           DATE         NOT NULL            CHECK(BuiltDate >= '1900-01-01' AND BuiltDate <= GETDATE()),
);

INSERT INTO Airplanes(AirplaneModel, Availability, CurrentLocation, BuiltDate)
VALUES (1, 'N', NULL, '2007-09-09'),
       (2, 'Y', 'ORD', '2009-09-19'),
       (2, 'N', NULL, '2011-12-05'),
       (3, 'N', 'JFK', '2006-01-14'),
       (3, 'Y', 'COS', '2010-10-23'),
       (5, 'N', NULL, '2014-01-13'),
       (6, 'N', 'JFK','1999-05-08');


-- Create Table AirportLocation
CREATE TABLE AirportLocation(
AirportId           VARCHAR(3)         PRIMARY KEY  REFERENCES  Airports(AirportId),
City                VARCHAR(50)        NOT NULL,
State               VARCHAR(50)        NOT NULL
);

INSERT INTO AirportLocation(AirportId, City, State)
VALUES        ('ORD', 'Chicago', 'Illinois'),
              ('SYR', 'Syracuse', 'New york'),
              ('JFK', 'New York City', 'New york'),
             ('SAN', 'San Diego', 'California'),
             ('SEA', 'Seattle', 'Washington'),
             ('COS', 'Colorado Springs', 'Colorado'),
             ('IAH', 'Houston', 'Texas'),
             ('MIA', 'Miami', 'Florida'),
             ('ATL', 'Atlanta', 'Georgia'),
             ('HNL', 'Honolulu', 'Hawaii');


-- Create Table Terminals
CREATE TABLE Terminals(
TerminalId          INTEGER             PRIMARY KEY  IDENTITY(1,1),
AirportId           VARCHAR(3)          NOT NULL     REFERENCES  Airports(AirportId),
TerminalName        VARCHAR(10)         NOT NULL,
);

INSERT INTO Terminals(AirportId, TerminalName)
VALUES       ('ORD', '1'),
             ('ORD', '2'),
             ('SYR', '2'),
             ('SYR', '1'),
             ('SAN', '2'),
             ('JFK', '1'),
             ('SEA', 'A'),
             ('SEA', 'C'),
             ('COS', '1'),
             ('IAH', 'B'),
             ('MIA', 'J'),
             ('ATL', 'E'),
             ('HNL', '1');

-- Create Table Gates
CREATE TABLE Gates(
GatesId             INTEGER             PRIMARY KEY  IDENTITY(1,1),
TerminalId          INTEGER             NOT NULL     REFERENCES Terminals(TerminalId),
GateName            VARCHAR(10)
);

INSERT INTO Gates(TerminalId, GateName)
VALUES       (1, '1'),
             (2, '4'),
             (2, '3'),
             (3, '4'),
             (3, '3'),
             (4, '1'),
             (5, '1'),
            (6, '5'),
            (7, '8'),
            (8, '9'),
            (9, '11'),
            (10, '1'),
            (11, '2'),
            (12, '6'),
            (13, '1');


-- Create Table AirportFees
CREATE TABLE AirportFees(
AirportId           VARCHAR(3)          PRIMARY KEY  REFERENCES  Airports(AirportId),
StateTaxesFees      DECIMAL(12,2)       NOT NULL     CHECK (StateTaxesFees >= 0),
CityTaxesFees       DECIMAL(12,2)       NOT NULL     CHECK (CityTaxesFees >= 0),
OtherFees           DECIMAL(12,2)       CHECK (OtherFees >= 0),
);

INSERT INTO AirportFees(AirportId, StateTaxesFees, CityTaxesFees, OtherFees)
VALUES ('ORD', 5630.25, 2636.15, NULL),
       ('SYR', 7221.52, 3618.50, 225.30),
       ('JFK', 10526.85, 6152.90, NULL),
       ('HNL', 9638.20, 5984.22, 651.20),
       ('MIA', 8942.63, 7895.62, 256.30),
       ('IAH', 12056.31, 9606.35, NULL),
       ('SAN', 16521.63, 96843.52, 1003.20),
       ('SEA', 13964.29, 8863.00, NULL),
       ('COS', 6652.12, 3652.19, 886.52),
       ('ATL', 9596.35, 9852.19, NULL);



--Create Table AirportsHandleAirplanes
CREATE TABLE AirportsHandleAirplanes(
Airport      VARCHAR(3)          REFERENCES   Airports(AirportId),
Airplane     INTEGER             REFERENCES   Airplanes(AirplaneId),
PRIMARY KEY  (Airport, Airplane)
);

INSERT INTO AirportsHandleAirplanes(Airport, Airplane)
VALUES ('JFK', 1),
       ('JFK', 2),
       ('JFK', 4),
       ('JFK', 6),
       ('MIA', 4),
       ('MIA', 1),
```

```sql
       ('SYR', 2),
       ('SYR', 1),
       ('ORD', 1),
       ('ORD', 2),
       ('ORD', 3),
       ('COS', 5),
       ('IAH', 5),
       ('ATL', 1),
       ('ATL', 2),
       ('SEA', 1),
       ('SEA', 2),
       ('HNL', 3);

-- Create Table SeatClass(Lookup table)
CREATE TABLE SeatClass(
SeatClassId         INTEGER             PRIMARY KEY     IDENTITY(1,1),
Text                VARCHAR(30)         NOT NULL
);

INSERT INTO SeatClass(Text)
VALUES ('First Class'),
       ('Business Class'),
       ('Economy Class');


-- Create Table Seats
CREATE TABLE Seats(
SeatsId             INTEGER      PRIMARY KEY  IDENTITY(1,1),
AirplaneId          INTEGER      NOT NULL     REFERENCES  Airplanes(AirplaneId),
SeatClassId         INTEGER      NOT NULL     REFERENCES  SeatClass(SeatClassId),
RowNumber           INTEGER      NOT NULL     CHECK(RowNumber >= 1)
);

INSERT INTO Seats(AirplaneId, SeatClassId, RowNumber)
VALUES (2, 3, 1),
       (1, 3, 1),
       (2, 3, 1),
       (2, 3, 1),
       (3, 1, 2),
       (4, 3, 2),
       (5, 3, 3),
       (5, 3, 3),
       (5, 2, 3),
       (2, 3, 3);
```

```sql
-- Create Table FlightRoutes
CREATE TABLE FlightRoutes(
RouteId                   INTEGER      PRIMARY KEY  IDENTITY(1,1),
DepartureAirportId        VARCHAR(3)   NOT NULL     REFERENCES  Airports(AirportId),
ArrivalAirportId          VARCHAR(3)   NOT NULL     REFERENCES  Airports(AirportId),
FlightDistance            INTEGER      CHECK(FlightDistance > 0),
FlightDuration            INTEGER      CHECK(FlightDuration > 0)
);

INSERT INTO FlightRoutes(DepartureAirportId, ArrivalAirportId, FlightDistance, FlightDuration)
VALUES ('SYR', 'ORD', 667, 150),
       ('ORD', 'HNL', 2425, 545),
       ('ATL', 'SEA', 2002, 450),
       ('COS', 'IAH', 735, 165),
       ('JFK', 'MIA', 1312, 450),
       ('ORD', 'SYR', 667, 150),
       ('HNL', 'ORD', 2425, 545),
       ('SEA', 'ATL', 2002, 450),
       ('IAH', 'COS', 735, 165),
       ('MIA', 'JFK', 1312, 450);

-- Create Table FlightSchedules
CREATE TABLE FlightSchedules(
FlightId            INTEGER             PRIMARY KEY  IDENTITY(1,1),
FlightRoute         INTEGER             NOT NULL     REFERENCES FlightRoutes(RouteId),
AirplaneId          INTEGER             NOT NULL     REFERENCES Airplanes(AirplaneId),
FlightNumber        VARCHAR(6)          NOT NULL
);

INSERT INTO FlightSchedules(FlightRoute, AirplaneId, FlightNumber)
VALUES (5, 4, 'AB2153'),
       (1, 2, 'AB1536'),
       (4, 5, 'AB3620'),
       (3, 1, 'AB1320'),
       (1, 2, 'AB1536'),
       (2, 3, 'AB2019'),
       (4, 5, 'AB3620'),
       (1, 2, 'AB1536');



-- Create Table Reservations
CREATE TABLE Reservations(
ReservationId       INTEGER      PRIMARY KEY  IDENTITY(1,1),
CustomerId          INTEGER      NOT NULL     REFERENCES Customers(CustomerId),
FlightId            INTEGER      NOT NULL     REFERENCES FlightSchedules(FlightId),
ReservationStatus   INTEGER      NOT NULL     REFERENCES ReservationStatus(StatusId),
ReservationMethod   INTEGER      REFERENCES   ReservationMethods(ReservationMethodsId),
DateOfReservation   DATETIME     NOT NULL     CHECK(DateOfReservation >= '1900-01-01' AND DateOfReservation <= GETDATE())
);

-- Some tickets were reserved while others were purchased without reservation, some reservation was not successful, some
reservation was cancelled…
INSERT INTO Reservations(CustomerId, FlightId, ReservationStatus, ReservationMethod, DateOfReservation)
VALUES (1, 1, 1, 1,'2014-01-12T23:01:12'),
       (10, 1, 2, 3,'2014-05-05T09:22:49'),
       (5, 2, 4, 2,'2014-08-17T07:10:55'),
       (5, 2, 1, 2,'2014-08-17T23:01:45'),
       (4, 3, 1, 1,'2015-05-01T09:36:40'),
       (1, 1, 4, 3,'2015-10-10T22:13:05'),
       (8, 5, 1, 2,'2016-01-05T16:08:20'),
       (1, 5, 1, 2,'2016-01-11T08:11:06'),
       (1, 6, 1, 3,'2016-01-11T08:12:59'),
       (9, 8, 1, 1,'2016-06-21T12:04:50');

-- Create Table FlightTimes
CREATE TABLE FlightTimes(
FlightId                  INTEGER      PRIMARY KEY  REFERENCES FlightSchedules(FlightId),
ScheduledDepartureTime    DATETIME     NOT NULL     CHECK(ScheduledDepartureTime >= '1900-01-01' AND ScheduledDepartureTime <=
GETDATE()),
ScheduledArrivalTime      DATETIME     NOT NULL     CHECK(ScheduledArrivalTime >= '1900-01-01' AND ScheduledArrivalTime <=
GETDATE()),
ProjectedDepartureTime    DATETIME     CHECK(ProjectedDepartureTime >= '1900-01-01' AND ProjectedDepartureTime <= GETDATE()),
ProjectedArrivalTime      DATETIME     CHECK(ProjectedArrivalTime >= '1900-01-01' AND ProjectedArrivalTime <= GETDATE()),
ActualDepartureTime       DATETIME     CHECK(ActualDepartureTime >= '1900-01-01' AND ActualDepartureTime <= GETDATE()),
ActualArrivalTime         DATETIME     CHECK(ActualArrivalTime >= '1900-01-01' AND ActualArrivalTime <= GETDATE()),
);


INSERT INTO FlightTimes(FlightId, ScheduledDepartureTime, ScheduledArrivalTime, ProjectedDepartureTime, ProjectedArrivalTime,
ActualDepartureTime, ActualArrivalTime)
VALUES      (1, '2014-02-26T11:15:00', '2014-02-26T16:10:00', '2014-02-26T11:15:00', '2014-02-26T16:10:00', '2014-02-
26T11:15:00', '2014-02-26T16:20:00'),
            (2, '2014-08-25T07:30:00', '2014-08-25T10:00:00', '2014-08-25T07:30:00', '2014-08-25T10:00:00', '2014-08-
25T07:40:00', '2014-08-25T10:00:00'),
            (3, '2015-06-05T14:20:00', '2015-06-05T16:55:00', '2015-06-05T14:20:00', '2015-06-05T16:55:00', '2015-06-
05T14:30:00', '2015-06-05T17:15:00'),
            (4, '2015-07-05T20:10:00', '2015-07-06T03:40:00', '2015-07-05T20:30:00', '2015-07-05T04:00:00', '2015-07-
05T21:00:00', '2015-07-06T04:40:00'),
            (5, '2016-01-26T07:30:00', '2016-01-26T10:00:00', '2016-01-26T07:30:00', '2016-01-26T10:00:00', '2016-01-
26T07:30:00', '2016-01-26T10:00:00'),
             (6, '2016-04-02T23:15:00', '2016-04-03T05:00:00', '2016-04-02T23:15:00', '2016-04-03T05:00:00', '2016-04-
02T23:15:00', '2016-04-03T05:10:00'),
             (7, '2016-05-11T05:40:00', '2016-05-11T08:25:00', '2016-05-11T05:40:00', '2016-05-11T08:25:00', '2016-05-
11T05:40:00', '2016-05-11T08:25:00'),
             (8, '2016-07-05T07:30:00', '2016-07-05T10:00:00', '2016-07-05T07:30:00', '2016-07-05T10:00:00', NULL, NULL);

--Create Table FlightGates
CREATE TABLE FlightGates(
FlightId      INTEGER      PRIMARY KEY        REFERENCES FlightSchedules(FlightId),
ScheduleDepartureGate      INTEGER            NOT NULL     REFERENCES Gates(GatesId),
ScheduleArrivalGate        INTEGER            NOT NULL     REFERENCES Gates(GatesId),
ActualDepartureGate        INTEGER            REFERENCES Gates(GatesId),
ActualArrivalGate          INTEGER            REFERENCES Gates(GatesId)
);

INSERT INTO FlightGates(FlightId, ScheduleDepartureGate, ScheduleArrivalGate, ActualDepartureGate, ActualArrivalGate)
VALUES       (1, 8, 13, 8, 13),
             (2, 4, 1, 4, 2),
             (3, 11, 12, 11, 12),
             (4, 14, 9, 14, 10),
             (5, 6, 1, 5, 1),
             (6, 2, 15, 2, 15),
             (7, 11, 12, 11, 12),
             (8, 4, 3, 4, 3);

-- Create Table Tickets
CREATE TABLE Tickets(
TicketNumber INTEGER      PRIMARY KEY  IDENTITY(1,1),
CustomerId   INTEGER      NOT NULL     REFERENCES Customers(CustomerId),
FlightId     INTEGER      NOT NULL     REFERENCES FlightSchedules(FlightId),
SeatsId      INTEGER      NOT NULL     REFERENCES Seats(SeatsId)
);

INSERT INTO Tickets(CustomerId, FlightId, SeatsId)
VALUES (1, 1, 7),
       (5, 2, 2),
       (3, 3, 8),
       (4, 3, 10),
       (2, 4, 1),
       (8, 5, 2),
       (9, 5, 3),
       (1, 5, 4),
       (1, 6, 6),
       (3, 7, 10),
       (9, 8, 3);
```

```sql
--Create Table CustomerTransaction
CREATE TABLE CustomerTransaction(
TransactionId       INTEGER       PRIMARY KEY   IDENTITY(1,1),
TicketNumber        INTEGER       NOT NULL      REFERENCES    Tickets(TicketNumber),
TransactionTime     DATETIME      NOT NULL      CHECK(TransactionTime >= '1900-01-01' AND TransactionTime <= GETDATE()),
TransactionStatus   CHAR          NOT NULL      CHECK (TransactionStatus IN ('Y', 'N'))
REFERENCES    TransactionStatusType(StatusTypeId),
Comments            VARCHAR(100)
);

INSERT INTO CustomerTransaction(TicketNumber, TransactionTime, TransactionStatus, Comments)
VALUES (1, '2014-01-12T23:12:05', 'Y', NULL),
       (2, '2014-08-18T11:33:12', 'N', 'Bank declined'),
       (2, '2014-08-18T13:52:13', 'Y', NULL),
       (3, '2015-06-05T08:22:06', 'Y', NULL),
       (4, '2015-05-01T11:25:47', 'Y', NULL),
       (5, '2015-07-05T14:55:01', 'Y', NULL),
       (6, '2016-01-05T16:11:53', 'Y', NULL),
       (7, '2016-01-25T23:25:23', 'Y', NULL),
       (8, '2016-01-11T08:11:47', 'Y', NULL),
       (9, '2016-01-11T08:13:10', 'Y', NULL),
       (10, '2016-05-11T01:25:01', 'Y', NULL),
       (11, '2016-06-21T17:00:05', 'Y', NULL);

--Create Table Costs
CREATE TABLE Costs(
TransactionId       INTEGER            PRIMARY KEY          REFERENCES CustomerTransaction(TransactionId),
TicketPrice         DECIMAL(16,2)      NOT NULL             CHECK (TicketPrice >= 0),
MileageUsed         INTEGER            NOT NULL             CHECK (MileageUsed >= 0),
Taxes               DECIMAL(16,2)      CHECK (Taxes >= 0),
ServiceFees         DECIMAL(16,2)      CHECK (ServiceFees >= 0),
Discount            DECIMAL(16,2)      CHECK (Discount >= 0 ),
);

INSERT INTO Costs(TransactionId, TicketPrice, MileageUsed, Taxes, ServiceFees, Discount)
VALUES (1, 458.00, 0, 36.64, 15.00, 0.00),
       (2, 298.00, 0, 23.84, 20.00, 14.90),
       (3, 0, 670, 23.84, 20.00, 14.90),
       (4, 869.00, 0, 69.52, 10.00, 43.45),
       (5, 368.00, 0, 29.44, 0.00, 0.00),
       (6, 686.00, 0, 54.88, 15.00, 34.3),
       (7, 275.00, 0, 22.00, 50.00, 13.75),
       (8, 115.00, 675, 33.20, 15.00, 0.00),
       (9, 295.00, 0, 23.60, 0.00, 14.75),
       (10, 996.00, 1125, 127.68, 15.00, 0.00),
       (11, 639.00, 0, 51.12, 0.00, 31.95),
(12, 273.00, 0, 21.84, 15.00, 13.65);
```

VIEW 1

```
--find out the flight history of the Customers, show the CustomerId and also their full names
CREATE VIEW CustomersFlightHistory AS
SELECT c.FirstName + ' ' + c.LastName AS [Customer Name], temp.CustomerId, temp.FlightNumber, temp.FlightTimes
       FROM
              (SELECT c.CustomerId, f.FlightNumber, COUNT(*) AS FlightTimes
                     FROM Customers c INNER JOIN Tickets t
                     ON c.CustomerId = t.CustomerId
                     INNER JOIN FlightSchedules f
                     ON f.FlightId = t.FlightId
                     GROUP BY c.CustomerId, f.FlightNumber) temp INNER JOIN Customers c
                     ON temp.CustomerId = c.CustomerId
```

VIEW 2

```
--find out all the FlightId and their departure/arrival cities
CREATE VIEW FlightBetweenCities   AS
       SELECT F.FlightId, AL1.City AS [Departure City], AL2.City AS [Arrival City]
              FROM FlightSchedules F INNER JOIN FlightRoutes R
                     ON F.FlightRoute = R.RouteId
                     INNER JOIN Airports A1
                     ON A1.AirportId = R.DepartureAirportId
                     INNER JOIN Airports A2
                     ON A2.AirportId = R.ArrivalAirportId
                     INNER JOIN AirportLocation AL1
                     ON AL1.AirportId = A1.AirportId
                     INNER JOIN AirportLocation AL2
                     ON AL2.AirportId = A2.AirportId
```

VIEW 3

```
--find out all the maintenance works done by the Mechanics, the work on which plane, the plane model, maintenance details and
start/finish time
CREATE VIEW EmployeesMaitenanceAirPlane AS
       SELECT E.FirstName + ' ' + E.LastName AS [Employee Name], A.AirplaneId, AM. AirplaneModelName, ML.Comments, PM.StartTime,
PM.FinishTime
              FROM Employees E INNER JOIN MaintenanceRecords M
                     ON E.EmployeeId = M.EmployeeId
                     INNER JOIN PlaneMaintenances PM
                     ON PM.MaintenanceId = M.MaintenanceId
                     INNER JOIN MaintenanceLog ML
                     ON ML.MaintenanceLogId = PM.MaintenanceLogId
                     INNER JOIN Airplanes A
                     ON A.AirplaneId = PM.AirplaneId
                     INNER JOIN AirplaneModels AM
                     ON AM.AirplaneModelId = A.AirplaneModel
```

VIEW 4

```
--find the cusomers who has placed reservations more than 2 times in the years 2014-2016, also show their Email address so we can contact
them for giving rewards
CREATE VIEW FindValuableCustomers AS
       SELECT c.FirstName + ' ' + c.LastName AS [Customer Name], c.Email, temp.[Total Order Times]
              FROM
                     (SELECT r.CustomerId, COUNT(*) AS [Total Order Times]
                            FROM Customers c, Reservations r
                            WHERE c.CustomerId = r.CustomerId
                                   AND r.ReservationStatus = '1'
                                   AND  YEAR(r.DateOfReservation) IN ('2014', '2015', '2016')
                            GROUP BY r. CustomerId
                            HAVING COUNT(*) >= 2) Temp INNER JOIN Customers c
                            ON Temp.CustomerId = c.CustomerId
```

PROCEDURE 1

```sql
--when input a CustomerId, this procedure will calculate the total money that customer spent
CREATE PROCEDURE myInquriyProcedure1 (@id AS INT) AS
BEGIN
	IF (NOT EXISTS(SELECT 1 FROM Customers WHERE CustomerId = @id))
		BEGIN
		--The input id was not found in customers
			PRINT 'Error: Customer not found, please check again.'
			RETURN  -1
		END
	DECLARE @totalCost   DECIMAL(16,2)
	DECLARE @customerId INT
	DECLARE @ticketPrice DECIMAL (16,2)
	DECLARE @taxes DECIMAL (16,2)
	DECLARE @discount DECIMAL (16,2)
	DECLARE @serviceFee DECIMAL (16,2)
	SET @totalCost = 0
	SET @ticketPrice = 0
	SET @taxes = 0
	SET @discount = 0
	SET @serviceFee = 0

	--Use CURSOR for traversing the table
	DECLARE myCursor CURSOR FOR
	SELECT c.CustomerId, cs.TicketPrice, cs.ServiceFees, cs.Taxes, cs.Discount
		FROM Customers c INNER JOIN Tickets t
			ON c.CustomerId = t.CustomerId
			INNER JOIN CustomerTransaction ct
			ON ct.TicketNumber = t.TicketNumber
			INNER JOIN Costs cs
			ON cs.TransactionId = ct.TransactionId
				AND ct.TransactionStatus = 'Y'
	OPEN myCursor
	FETCH NEXT FROM myCursor INTO @customerId, @ticketPrice, @serviceFee, @taxes, @discount
	WHILE @@FETCH_STATUS = 0
		BEGIN
			IF (@customerId = @id)
				BEGIN
					SET @totalCost = @totalCost + @ticketPrice + @serviceFee + @taxes -@discount
				END
			FETCH NEXT FROM myCursor INTO @customerId, @ticketPrice, @serviceFee, @taxes, @discount
		END
	CLOSE myCursor
	DEALLOCATE myCursor
	RETURN @totalCost
END
```

FUNCTION 2

```
--The function will take a year as input parameter, then return a table contains the info of:
--total income of this year and total income from customers who booked the tickets/directly purchased the tickets
CREATE FUNCTION dbo.myFunction2 (@paramYear INT)
        RETURNS @t TABLE
        (
                TimeOfYear       INT PRIMARY KEY,
                TotalIncome DECIMAL(16,2)  NULL,
                IncomeWithReservation DECIMAL (16,2) NULL,
                IncomeWithoutReservation DECIMAL(16,2) NULL

        )
        AS
        BEGIN
                DECLARE
                        @TotalIncome DECIMAL(16,2),
                        @IncomeWithReservation DECIMAL (16,2),
                        @IncomeWithoutReservation DECIMAL (16,2);


                SET @IncomeWithReservation = (SELECT SUM (c.TicketPrice + c.ServiceFees + c.Taxes - c.Discount)
                                                FROM CustomerTransaction ct INNER JOIN Tickets t
                                                ON ct.TicketNumber = t.TicketNumber
                                                INNER JOIN Costs c
                                                ON c.TransactionId = ct.TransactionId
                                                WHERE ct.TransactionStatus = 'Y'
                                                        AND YEAR (ct.TransactionTime) = @paramYear
                                                        AND (t.TicketNumber IN (SELECT TicketNumber
                                                                                FROM(SELECT Customers.CustomerId, FlightSchedules.FlightId
                                                                                        FROM Customers INNER JOIN Reservations
                                                                                                ON Customers.CustomerId = Reservations.CustomerId
                                                                                                INNER JOIN FlightSchedules
                                                                                                ON FlightSchedules.FlightId = Reservations.FlightId
                                                                                                WHERE Reservations.ReservationStatus = 1) X INNER JOIN Tickets
                                                                                                ON Tickets.CustomerId = X.CustomerId
                                                                                                AND Tickets.FlightId = X.FlightId)))

                SET @TotalIncome = (SELECT SUM (c.TicketPrice + c.ServiceFees + c.Taxes - c.Discount)
                                                FROM CustomerTransaction ct INNER JOIN Costs c
                                                        ON c.TransactionId = ct.TransactionId
                                                        WHERE ct.TransactionStatus = 'Y'
                                                                AND YEAR (ct.TransactionTime) = @paramYear)

                SET @IncomeWithoutReservation = @TotalIncome - @IncomeWithReservation
                BEGIN
                        INSERT @t
                        SELECT  @paramYear, @TotalIncome, @IncomeWithReservation, @IncomeWithoutReservation;
                END
                RETURN;
        END
        GO
```

PROCEDURE 3

```sql
--Based on the Customers' flight history, update the frequencyFlighNumber in Customers table, and output appropriate message
CREATE PROCEDURE myInquriyProcedure3 AS
       BEGIN
              DECLARE @customerId INT
              DECLARE @temp VARCHAR(6)
              DECLARE @flightNumber VARCHAR(6)
              DECLARE @customerName VARCHAR
              DECLARE mCursor CURSOR FOR
              SELECT CustomerId from CustomersFlightHistory
              OPEN mCursor
              FETCH NEXT FROM mCursor INTO @customerId
              WHILE @@FETCH_STATUS = 0
                     BEGIN
                            --use the view we created in a procedure to save extra codes
                            SET @flightNumber = (SELECT TOP 1 FlightNumber FROM CustomersFlightHistory
                                   WHERE CustomerId = @customerId)
                            SET @temp = (SELECT TOP 1 Customers.FrequentFlyNumber FROM Customers
                                          WHERE Customers.CustomerId = @customerId)
                            IF ((@temp IS NULL) OR (NOT @temp = @flightNumber))
                                   BEGIN
                                          IF(@temp IS NULL)
                                                 BEGIN
                                                        --the FrequentFlyNumber will be inserted
                                                        PRINT 'FrequentFlyNumber was inserted for customer (customerId = ' +
                                                        STR(@customerId) +')'
                                                 END
                                          ELSE
                                                 BEGIN
                                                        --the FrequentFlyNumber will be updated
                                                        PRINT 'FrequentFlyNumber was updated from ' + @temp + ' to ' + @flightNumber
                                                 END
                                          UPDATE Customers
                                                 SET FrequentFlyNumber = @flightNumber
                                                 WHERE CustomerId = @customerId

                                   END
                            FETCH NEXT FROM mCursor INTO @customerId
                     END
              CLOSE mCursor
              DEALLOCATE mCursor
              RETURN
       END
```

PROCEDURE 4

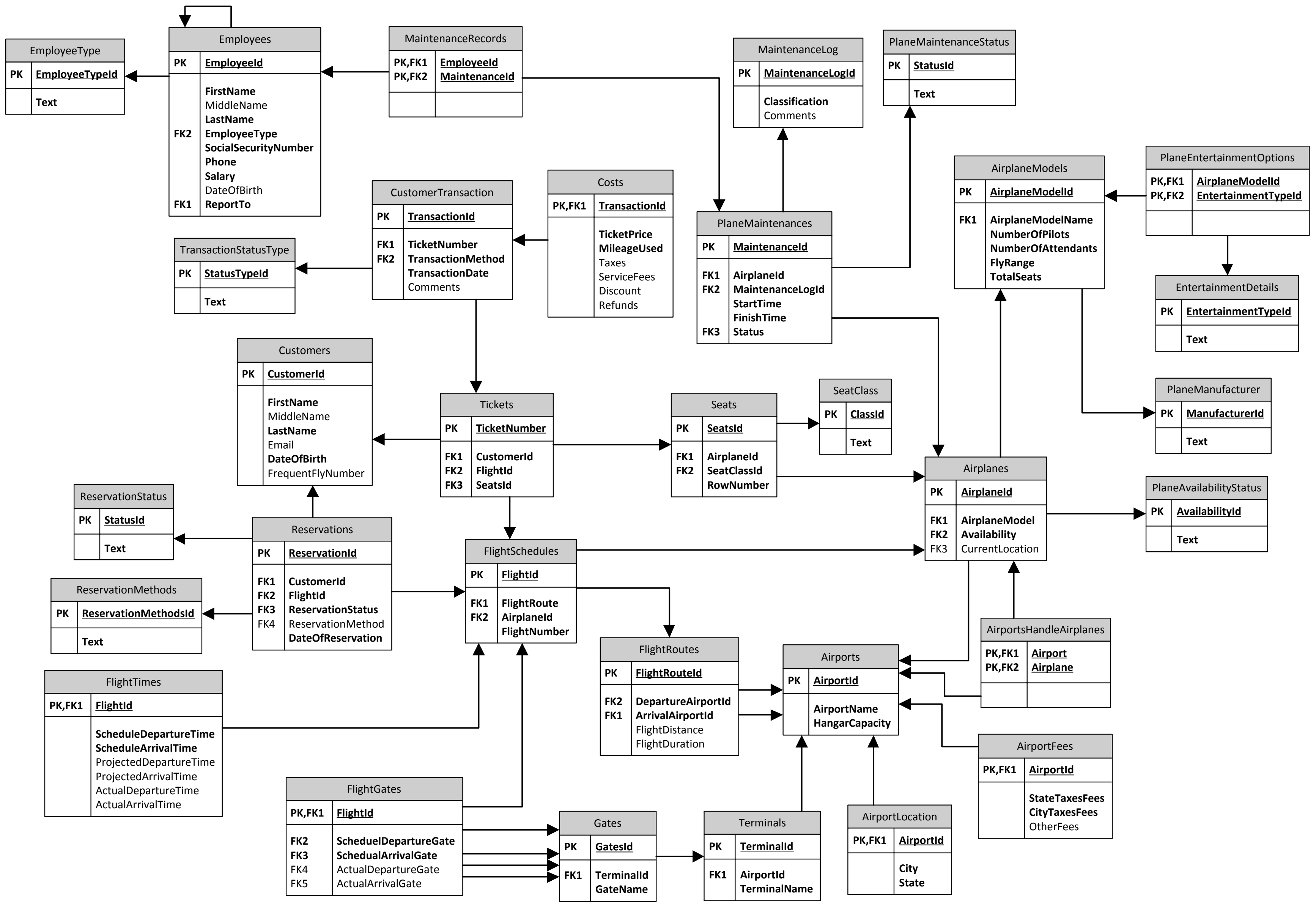
```sql
--This procedure will take a year as input parameter, calculate and return the flight punctuality rates for both Departures and Arrivals
```

```
CREATE PROCEDURE myInquriyProcedure4 (@year INT, @onScheduleDepartureRate DECIMAL(5,2) OUTPUT,  @onScheduleArrivalRate  DECIMAL(5,2)
OUTPUT )
       AS
       BEGIN
              DECLARE @totalFightsTimes   INT
              DECLARE @lateDepartureTimes INT
              DECLARE @lateArrivalTimes   INT

              SET @lateArrivalTimes = (SELECT count(*)
                                         FROM FlightTimes f
                                             WHERE f.ActualArrivalTime > f.ScheduledArrivalTime
                                                   AND YEAR(ActualArrivalTime) = @year)

              SET @lateDepartureTimes = (SELECT count(*)
                                            FROM FlightTimes f
                                                WHERE f.ActualDepartureTime > f.ScheduledDepartureTime
                                                      AND YEAR(ActualDepartureTime) = @year)
              SET @totalFightsTimes = (SELECT COUNT(*)
                                          FROM FlightTimes f
                                                WHERE YEAR(ActualDepartureTime) = @year)
              PRINT CAST (@totalFightsTimes AS VARCHAR (10))
              --error prevention using TRY/CATCH
              BEGIN TRY
                     SET @onScheduleDepartureRate = @lateDepartureTimes/ @totalFightsTimes
                     SET @onScheduleArrivalRate = @lateArrivalTimes/ @totalFightsTimes
              END TRY

              BEGIN CATCH
                     SET @onScheduleDepartureRate = -1
                     SET @onScheduleArrivalRate = -1
              END CATCH
              PRINT CAST (@onScheduleDepartureRate AS VARCHAR (10))
       END;
       RETURN
       GO
```


EmployeeType
PK EmployeeTypeId
Text
Employees
PK EmployeeId
FirstName
MiddleName
LastName
FK2 EmployeeType
SocialSecurityNumber
Phone
Salary
DateOfBirth
FK1 ReportTo
MaintenanceRecords
PK,FK1 EmployeeId
PK,FK2 MaintenanceId
MaintenanceLog
PK MaintenanceLogId
Classification
Comments
PlaneMaintenanceStatus
PK StatusId
Text
CustomerTransaction
PK TransactionId
FK1 TicketNumber
FK2 TransactionMethod
TransactionDate
Comments
Costs
PK,FK1 TransactionId
TicketPrice
MileageUsed
Taxes
ServiceFees
Discount
Refunds
TransactionStatusType
PK StatusTypeId
Text
PlaneMaintenances
PK MaintenanceId
FK1 AirplaneId
FK2 MaintenanceLogId
StartTime
FinishTime
FK3 Status
AirplaneModels
PK AirplaneModelId
FK1 AirplaneModelName
NumberOfPilots
NumberOfAttendants
FlyRange
TotalSeats
PlaneEntertainmentOptions
PK,FK1 AirplaneModelId
PK,FK2 EntertainmentTypeId
EntertainmentDetails
PK EntertainmentTypeId
Text
Customers
PK CustomerId
FirstName
MiddleName
LastName
Email
DateOfBirth
FrequentFlyNumber
Tickets
PK TicketNumber
FK1 CustomerId
FK2 FlightId
FK3 SeatsId
Seats
PK SeatsId
FK1 AirplaneId
FK2 SeatClassId
RowNumber
SeatClass
PK ClassId
Text
PlaneManufacturer
PK ManufacturerId
Text
Airplanes
PK AirplaneId
FK1 AirplaneModel
FK2 Availability
FK3 CurrentLocation
PlaneAvailabilityStatus
PK AvailabilityId
Text
ReservationStatus
PK StatusId
Text
Reservations
PK ReservationId
FK1 CustomerId
FK2 FlightId
FK3 ReservationStatus
FK4 ReservationMethod
DateOfReservation
FlightSchedules
PK FlightId
FK1 FlightRoute
FK2 AirplaneId
FlightNumber
ReservationMethods
PK ReservationMethodsId
Text
AirportsHandleAirplanes
PK,FK1 Airport
PK,FK2 Airplane
FlightRoutes
PK FlightRouteId
FK2 DepartureAirportId
FK1 ArrivalAirportId
FlightDistance
FlightDuration
Airports
PK AirportId
AirportName
HangarCapacity
FlightTimes
PK,FK1 FlightId
ScheduleDepartureTime
ScheduleArrivalTime
ProjectedDepartureTime
ProjectedArrivalTime
ActualDepartureTime
ActualArrivalTime
AirportFees
PK,FK1 AirportId
StateTaxesFees
CityTaxesFees
OtherFees
FlightGates
PK,FK1 FlightId
FK2 ScheduelDepartureGate
FK3 SchedualArrivalGate
FK4 ActualDepartureGate
FK5 ActualArrivalGate
Gates
PK GatesId
FK1 TerminalId
GateName
Terminals
PK TerminalId
FK1 AirportId
TerminalName
AirportLocation
PK,FK1 AirportId
City
State