Data Management - Part A

Group 13

2020/11/13

Contents

1 Part A

1.1 Specifying the tables with DDL Queries

- Create tables for the database
- Set up data types and key constraints
- Assign AUTOINCREMENT to Primary Keys to generate unique numbers automatically

1.1.1 Guest table

- Information about each guest of all the hotels is tracked historically
- Hotel guest is anyone who used any service at a hotel (either accommodation or additional services)

```
-- Guests
CREATE TABLE 'Guest'(
  'GuestID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'FirstName' VARCHAR(30) NOT NULL,
  'MiddleName' VARCHAR(30),
  'LastName' VARCHAR(30) NOT NULL,
  'StreetNumber' INTEGER,
  'StreetName' VARCHAR(30) NOT NULL,
  'City' VARCHAR(30) NOT NULL,
  'State' VARCHAR(30),
  'PostalCode' VARCHAR(10) NOT NULL,
  'Country' VARCHAR(30) NOT NULL,
  'Email' VARCHAR(100),
  'HomePhone' INTEGER,
  'WorkPhone' INTEGER,
  'CellPhone' INTEGER
);
```

1.1.2 Hotel table

```
-- Hotel
CREATE TABLE 'Hotels'(
  'HotelID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'HotelName' VARCHAR(30) NOT NULL,
  'StreetNumber' INTEGER,
  'StreetName' VARCHAR(30),
  'City' VARCHAR(30) NOT NULL,
  'State' VARCHAR(30),
  'PostalCode' VARCHAR(10) NOT NULL,
  'Country' VARCHAR(30) NOT NULL,
  'WebAddress' VARCHAR(100),
  'PrimaryPhone' INTEGER
);
```

1.1.3 Channel Provider table

- Channel Providers charge a booking fee for each reservation

- This booking fee is fixed by channel provider and doesn't vary by hotel

```sql
-- Channel Provider
CREATE TABLE 'ChannelProvider'(
  'ChannelProviderID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'ChannelProviderName' VARCHAR(30) NOT NULL,
  'Website' VARCHAR(200),
  'BookingFeeRate' DOUBLE NOT NULL
);
```

1.1.4 Room table

- Room ID represents the room number
- Floor 1-5 are considered to be Low
- Floors > 5 are considered to be High
- Room Rates are fixed in this database, the seasonal costs are handled by another software

```sql
-- Room
CREATE TABLE 'Room'(
  'RoomID' VARCHAR(30),
  'HotelID' INTEGER NOT NULL,
  'Floor' INTEGER NOT NULL,
  'HighOrLow' VARCHAR(10),
  'NumberOfBeds' INTEGER NOT NULL,
  'SmokingAllowed' BOOLEAN,
  'RoomRate' INTEGER,
PRIMARY KEY ('RoomID', 'HotelID'),
FOREIGN KEY ('HotelID')
  REFERENCES Hotel('HotelID'),
);
```

1.1.5 Service Table

- The Service table stores accommodation as well as additional services
- Additional services include phone charging equipment rental, the use of hotel facilities, etc.
- The available hotel facilities vary based on the hotel
- At the hotel restaurant and bar reservations are mandatory, no walk-ins allowed
- Thus, all new guests should contact the hotel and leave their personal details first
- Costs of items on the restaurant's and bar's menu as well as equipment and other hotel facilities rentals. Another software provides these costs to our database

```sql
-- Service
CREATE TABLE 'Service'(
  'ServiceID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'ServiceName' VARCHAR(30) NOT NULL,
  'HotelID' INTEGER,
  FOREIGN KEY ('HotelID')
    REFERENCES Hotel('HotelID')
);
```

1.1.6 Reservation table

- A guest might make multiple reservations
- A guest receives a separate ReservationID for each additional room he/she books

```sql
-- Reservation
CREATE TABLE 'Reservation'(
  'ReservationID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'HotelID' INTEGER NOT NULL,
  'GuestID' INTEGER NOT NULL,
  'RoomID' VARCHAR(30) NOT NULL,
  'ChannelProviderID' INTEGER,
  'CreditCardNumber' INTEGER NOT NULL,
  'CardExpireYear' INTEGER NOT NULL,
  'CardExpireMonth' INTEGER NOT NULL,
  'DateFrom' DATE NOT NULL,
  'DateTo' DATE NOT NULL,
  'SmokingPreference' BOOLEAN DEFAULT FALSE,
  'NumberOfBeds' INTEGER DEFAULT 1,
  'NumberOfGuests' INTEGER DEFAULT 1,
  FOREIGN KEY ('GuestID')
    REFERENCES Guest('GuestID'),
  FOREIGN KEY ('RoomID','HotelID')
    REFERENCES Room('RoomID','HotelID'),
  FOREIGN KEY ('ChannelProviderID')
    REFERENCES ChannelProvider('ChannelProviderID'),
  CHECK(CreditCardNumber BETWEEN 0 AND 9999999999999999),
  CHECK(CardExpireYear BETWEEN 0 AND 99),
  CHECK(CardExpireMonth BETWEEN 1 AND 12)
);
```

1.1.7 Past Reservation table

- Reservation is removed from the Reservation table after it has been used
- It is considered used when either a guests check in or cancels his/her reservation
- Information on these reservations is stored in the Past Reservation table

```sql
-- Past Reservations
CREATE TABLE 'PastReservation'(
  'PastReservationID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'HotelID' INTEGER NOT NULL,
  'GuestID' INTEGER NOT NULL,
  'RoomID' VARCHAR(30) NOT NULL,
  'DateFrom' DATE NOT NULL,
  'DateTo' DATE NOT NULL,
  'PastReservationStatus' VARCHAR(20),
  'ChannelProviderID' INTEGER,
  FOREIGN KEY ('GuestID')
    REFERENCES Guest('GuestID')
);
```

1.1.8 Room Status table

- Tracks the rooms status across all hotels
- The information in this table is used to calculate NumberOfAvailableRooms for the Room Available table

```sql
-- Room Status
CREATE TABLE 'RoomStatus'(
  'RoomStatusID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'RoomStatus' BOOLEAN DEFAULT FALSE,
  'RoomID' VARCHAR(30),
  'HotelID' INTEGER,
  'rsDate' DATE NOT NULL,
FOREIGN KEY ('RoomID', 'HotelID')
  REFERENCES Room('RoomID', 'HotelID')
);
```

1.1.9 Room Available table

- Each hotel provider requires at least two rooms to be available for booking at any time
- We are tracking number of rooms available at each hotel daily since the current date and onward
- Available rooms are all rooms which are not reserved during a specific period of time
- Another software processes the information in the given table and sends it to the channel providers
- On a provider's website, when a user is searching for a room during a specific period and if there are less than 2 rooms available for booking at a particular hotel, this hotel is not displayed in the search results on the provider's website.

```sql
-- Room Available
CREATE TABLE 'RoomAvailable'(
  'ChannelProviderID' INTEGER,
  'HotelID' INTEGER,
  'NumberOfAvailableRooms' INTEGER DEFAULT 0,
  'DateFrom' DATE,
  'DateTo' DATE,
  PRIMARY KEY ('ChannelProviderID', 'HotelID'),
  FOREIGN KEY ('ChannelProviderID')
    REFERENCES ChannelProvider('ChannelProviderID'),
  FOREIGN KEY ('HotelID')
    REFERENCES Hotel('HotelID')
);
```

1.1.10 Invoice table

- An invoice is considered paid if outstanding amount is £0.00
- PaidFlag shows whether an invoice was paid in full (PaidFlag = TRUE/FALSE)
- DateFrom indicates the date when the first charge was made for a particular stay
- InvoiceCheckOutTime is assigned when all charges are paid in full by a guest and after he/she checks out
- Amount Outstanding is updated immediately right after any payment or charge is issued

```sql
-- Invoice
CREATE TABLE 'Invoice'(
  'InvoiceID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'GuestID' INTEGER NOT NULL,
  'AmountOutstanding' INTEGER,
  'PaidFlag' BOOLEAN DEFAULT FALSE,
  'DateFrom' DATETIME NOT NULL,
  'InvoiceCheckOutTime' DATETIME,
  FOREIGN KEY ('GuestID')
    REFERENCES Guest('GuestID')
);
```

1.1.11 Charge table

- A guest can book a room and use additional services provided by a hotel
- Accommodation fees as well as additional costs are stored as Charges
- Charges are reflected by each guest during his/her particular stay

```sql
-- Charge
CREATE TABLE 'Charge'(
  'ChargeID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'GuestID' INTEGER NOT NULL,
  'ServiceID' INTEGER,
  'InvoiceID' INTEGER NOT NULL,
  'ChargeAmount' INTEGER NOT NULL,
  'ChargeTime' DATETIME,
  FOREIGN KEY ('GuestID')
    REFERENCES Guest('GuestID'),
  FOREIGN KEY ('ServiceID')
    REFERENCES Service('ServiceID'),
  FOREIGN KEY ('InvoiceID')
    REFERENCES Invoice('InvoiceID')
);
```

1.1.12 Payment table

- A guest may pay his/her bill at any given time during a particular stay
- A guest might make multiple payments of any amounts
- A wide variety of payment types are accepted (credit card. check, cash, etc)
- Different payment types are stored as PaymentType

```sql
-- Payment
CREATE TABLE 'Payment'(
  'PaymentID' INTEGER PRIMARY KEY AUTOINCREMENT,
  'GuestID' INTEGER NOT NULL,
  'HotelID' INTEGER,
  'InvoiceID' INTEGER,
  'PayAmount' INTEGER NOT NULL,
  'PaymentType' VARCHAR(30),
  'PayTime' DATETIME,
  FOREIGN KEY ('GuestID')
```

```
    REFERENCES Guest('GuestID'),
  FOREIGN KEY ('HotelID')
    REFERENCES Hotel('HotelID'),
  FOREIGN KEY ('InvoiceID')
    REFERENCES Invoice('InvoiceID')
);
```

1.2 A set of scenarios with SQL queries to describe the information flow in the database

1.2.1 Room Availability

```
-- Show a list of rooms available in specific the hotel during Nov 25-27, 2020
SELECT
      RoomID AS Available_RoomID
    , Floor
    , NumberOfBeds
    , SmokingPolicy
FROM Room
WHERE HotelID = 5

EXCEPT

SELECT
      res.RoomID
    , Room.Floor
    , Room.NumberOfBeds
    , Room.SmokingPolicy
FROM Reservation AS res
INNER JOIN Room USING(RoomID, HotelID)
WHERE HotelID = 5
AND ((res.DateFrom >= '2020-11-25' AND res.DateFrom < '2020-11-27')
  OR (res.DateTo > '2020-11-25' AND res.DateTo <= '2020-11-27')
  OR (res.DateFrom < '2020-11-25' AND res.DateTo > '2020-11-27'));
```

1.2.2 Reservations

```
-- The check-in process
INSERT INTO PastReservation
(HotelID, GuestID, RoomID, DateFrom, DateTo, PastReservationStatus, ChannelProviderID)
SELECT HotelID, GuestID, RoomID, DateFrom, DateTo, "", ChannelProviderID
FROM Reservation
WHERE ReservationID = 29;

DELETE FROM Reservation
WHERE ReservationID = 29;

UPDATE PastReservation
SET PastReservationStatus = "Checked-in"
```

```sql
WHERE PastReservationStatus = "";

-- The cancellation process. Use ReservationID to specify a reservation to be canceled
INSERT INTO PastReservation
(HotelID, GuestID, RoomID, DateFrom, DateTo, PastReservationStatus, ChannelProviderID)
SELECT HotelID, GuestID, RoomID, DateFrom, DateTo, "", ChannelProviderID
FROM Reservation
WHERE ReservationID = 27;

DELETE FROM Reservation
WHERE ReservationID = 27;

UPDATE PastReservation
SET PastReservationStatus = "Canceled"
WHERE PastReservationStatus = "";
```

1.2.3 Checking in

```sql
-- Generate the invoice for this stay while checking in using GuestID and ReservationID
INSERT INTO Invoice
(GuestID, AmountOutstanding, PaidFlag, DateFrom, InvoiceCheckOutTime)
SELECT GuestID, SUM(Each_Room_Fee) AS Total_Room_Fee, FALSE, DateFrom, NULL
FROM (
  SELECT
      res.GuestID
    , ((JULIANDAY(res.DateTo)-JULIANDAY(res.DateFrom))*room.RoomRate) AS RoomFee
    , res.DateFrom
  FROM Reservation AS res
  INNER JOIN Room AS room USING(RoomID, HotelID)
  WHERE res.ReservationID = 29
  )
GROUP BY GuestID;

-- Charge accommodation fees for this stay while checking in with InvoiceID
INSERT INTO Charge (GuestID, ServiceID, InvoiceID, ChargeAmount, ChargeTime)
SELECT GuestID, ServiceID, InvoiceID, AmountOutstanding, DATETIME('now')
FROM Invoice, Service
WHERE InvoiceID = 17
  AND ServiceName = "Accommodation"
  AND HotelID = 5;
```

1.2.4 During the stay in a hotel

```sql
-- Paying the accommodation fees
INSERT INTO Payment (GuestID, HotelID, InvoiceID, PayAmount, PaymentType, PayTime)
VALUES (1, 5, 17, 4500, "Cash", DATETIME('now'));

UPDATE Invoice
SET
    AmountOutstanding = (
        SELECT AmountOutstanding - PayAmount
        FROM Invoice INNER JOIN Payment USING(InvoiceID)
        WHERE InvoiceID = 17 AND PayTime = DATETIME('now'))
WHERE InvoiceID = 17;


-- New charge at the restaurant
INSERT INTO Charge (GuestID, ServiceID, InvoiceID, ChargeAmount, ChargeTime)
SELECT 1, ServiceID, 17, 50, DATETIME('now')
FROM Service
WHERE HotelID = 5 AND ServiceName = "Restaurant";

UPDATE Invoice
SET
    AmountOutstanding = (
        SELECT AmountOutstanding + ChargeAmount
        FROM Invoice INNER JOIN Charge USING(InvoiceID)
        WHERE InvoiceID = 17 AND ChargeTime = DATETIME('now'))
WHERE InvoiceID = 17;
```

1.2.5 Check Out

```sql
-- Check if Amount Outstanding of invoice #17 is £0.00
SELECT
    g.FirstName
  , g.LastName
  , i.AmountOutstanding
  , i.PaidFlag
  , i.DateFrom
  , i.InvoiceCheckOutTime
FROM Invoice AS i
INNER JOIN Guest AS g
ON i.GuestID = g.GuestID
WHERE InvoiceID = 17;


-- If yes, set the PaidFlag of invoice to TRUE and check out the invoice
UPDATE Invoice
SET InvoiceCheckOutTime = DATE('now'), PaidFlag = TRUE
WHERE InvoiceID = 17;


-- Otherwise, request the guest to pay the outstanding amount and check out the invoice
INSERT INTO Payment (GuestID, HotelID, InvoiceID, PayAmount, PaymentType, PayTime)
VALUES (1, 5, 17, 3190, "Credit Card", DATETIME('now'));

UPDATE Invoice
SET AmountOutstanding = (
        SELECT AmountOutstanding - PayAmount
        FROM Invoice INNER JOIN Payment USING(InvoiceID)
        WHERE InvoiceID = 17 AND PayTime = DATETIME('now'))
WHERE InvoiceID = 17;
```

```

UPDATE Invoice
SET InvoiceCheckOutTime = DATE('now'), PaidFlag = TRUE
WHERE InvoiceID = 17;


-- Check the invoice which lists all events
SELECT
      'Charge' AS EventType
    , ServiceName AS Event
    , ChargeAmount AS Amount
    , ChargeTime AS EventTime
FROM Charge INNER JOIN Service USING (ServiceID)
WHERE InvoiceID = 17

UNION

SELECT
      'Payment'
    , PaymentType
    , -PayAmount
    , PayTime
FROM Payment
WHERE InvoiceID = 17

UNION

SELECT
      ''
    , 'Total Balance'
    , AmountOutstanding
    , DATETIME('now')
FROM Invoice
WHERE InvoiceID = 17
ORDER BY EventTime;
```

1.3 A set of scenarios with SQL queries that satisfy the business goals

1.3.1 The total spend by customer for a particular stay

- The amount spent by customer during each stay is reflected in an invoice
- A guest might have several invoices that correspond to each stay
- An invoice is released when the amount outstanding is paid in full AND after a guest checks out
- Display the total amount spend by customer whose GuestID = 1 and InvoiceID = 1

```
SELECT
      c.GuestID
    , c.InvoiceID
    , SUM(c.ChargeAmount) AS TotalAmount
    , i.PaidFlag
    , i.InvoiceCheckOutTime AS InvoicePrinted
FROM Charge AS c
LEFT JOIN Invoice AS i
```

```sql
ON c.InvoiceID=i.InvoiceID
WHERE i.PaidFlag=1
  AND i.InvoiceCheckOutTime IS NOT NULL
  AND c.GuestID = 1
  AND c.InvoiceID = 1
GROUP BY c.GuestID, c.InvoiceID, i.PaidFlag
```

1.3.2 The most valuable customers

- The most valuable customers are defined as the highest paying customers in a specific period of time
- Payments by customer and date information are stored in the PAYMENT table
- Pull the top 10 most valuable customers by the specified period

```sql
-- in the last two months (excluding the current month)
SELECT
    GuestID
  , FirstName
  , LastName
  , SUM(PayAmount) AS TotalValue
  , DATE('now','start of month', '-2 month') AS DateFrom
  , DATE('now', 'start of month', '-1 day') AS DateTo
FROM Payment
NATURAL JOIN Guest
WHERE PayTime BETWEEN DATE('now','start of month', '-2 month')
                  AND DATE('now', 'start of month', '-1 day')
GROUP BY GuestID, FirstName, LastName, DateFrom, DateTo
ORDER BY TotalValue DESC
LIMIT 10;


-- in the past year
SELECT
    GuestID
  , FirstName
  , LastName
  , SUM(PayAmount) AS TotalValue
  , DATE('now','start of year', '-1 year') AS DateFrom
  , DATE('now', 'start of year', '-1 day') AS DateTo
FROM Payment
NATURAL JOIN Guest
WHERE PayTime BETWEEN DATE('now','start of year', '-1 year')
                  AND DATE('now', 'start of year', '-1 day')
GROUP BY GuestID, FirstName, LastName, DateFrom, DateTo
ORDER BY TotalValue DESC
LIMIT 10;


-- from the beginning of the records (excluding the current date)
SELECT
    p.GuestID
  , g.FirstName
  , g.LastName
  , SUM(p.PayAmount) AS TotalValue
  , DATE('now', '-1 day') AS DateTo
FROM Payment AS p
NATURAL JOIN Guest AS g
WHERE p.PayTime < DATE('now', '-1 day')
GROUP BY p.GuestID, g.FirstName, g.LastName, DateTo
ORDER BY TotalValue DESC
LIMIT 10;
```

1.3.3 Which are the top countries where our customers come from

- Pull top 10 countries by the total number of guests

```sql
SELECT
    Country
  , COUNT(GuestID) AS NumberOfGuests
FROM Guest
GROUP BY Country
ORDER BY COUNT(GuestID) DESC
LIMIT 10
```

1.3.4 How much did the hotel pay in referral fees for each of the platforms

- Each channel provider charges a booking fee for each reservation
- A set percentage is paid on each reservation made through each platform
- The booking fee is payable at the end of each month
- It is charged only when a reservation has been used (not canceled)

```sql
-- Create a view that combines Channel Provider and Room Bookings information
CREATE VIEW IF NOT EXISTS PastReservationByChannel
AS
SELECT
          prc.*
        , r.RoomRate
FROM
    (SELECT
          cp.ChannelProviderName
        , cp.BookingFeeRate
        , pr.HotelID
        , pr.RoomID
        , JulianDay(pr.DateTo)-Julianday(pr.DateFrom) AS DaysBooked
        , pr.PastReservationStatus
    FROM PastReservation AS pr
    LEFT JOIN ChannelProvider AS cp
    ON pr.ChannelProviderID = cp.ChannelProviderID
    WHERE pr.PastReservationStatus IS 'Checked-in'
    AND pr.DateTo < DATE('now', 'start of month', '-1 day')) AS prc
LEFT JOIN Room AS r
ON prc.HotelID = r.HotelID
AND prc.RoomID = r.RoomID

```

```sql
-- Calculate referral fees by hotel for each channel provider
SELECT
      HotelID
    , ChannelProviderName
    , SUM(BookingFeeRate*RoomRate*DaysBooked) AS ReferralFee
FROM PastReservationByChannelProvider
GROUP BY HotelID, ChannelProviderName
```

1.3.5 The utilization rate for each hotel in the last 12 months

- Calculate the average utilization of room bookings in each hotel
- The number of days of stays for a room represents the number of billable days
- Therefore, the average billable days of each hotel is the mean of the number of billable days for all rooms in that hotel
- In the last 12 month excluding the current month

```sql
SELECT
    t2.HotelID
  , AVG(t2.TotalBillableDays) AS AvgBillableDays
FROM
    ( SELECT
          t1.HotelID
        , t1.RoomID
        , SUM(t1.billable_day) AS TotalBillableDays
    FROM
        ( SELECT
              h.HotelID
            , pr.RoomID
            , CASE WHEN pr.DateFrom < DATE('now', 'start of month', '-12 month')
                   THEN JulianDay(pr.DateTo)-Julianday(DATE('now', 'start of month', '-12 month'))
                   ELSE JulianDay(pr.DateTo)-Julianday(pr.DateFrom)
                   END billable_day
        FROM PastReservation AS pr
        NATURAL JOIN Hotel AS h
        WHERE DATE(pr.DateTo, '-1 day') >= DATE('now', 'start of month', '-12 month')
              AND pr.PastReservationStatus = 'Checked-in' ) AS t1
    GROUP BY t1.HotelID, t1.RoomID ) AS t2
GROUP BY t2.HotelID
```

1.3.6 Customer Value in terms of total spent for each customer before the current booking

- The current booking means that a guest's invoice has not been released yet
- So, InvoiceCheckOutTime is NULL for the current bookings
- Calculate LTV by each customer based on total payments with invoice closed

```sql
SELECT
    i.GuestID
  , g.FirstName
  , g.LastName
  , SUM(p.PayAmount) AS TotalSpend
FROM Invoice  AS i
NATURAL JOIN Payment AS p
NATURAL JOIN Guest AS g
WHERE i.InvoiceCheckOutTime IS NOT NULL
GROUP BY i.GuestID, g.FirstName, g.LastName
```