Resort Management System

By

Team Data Sonic

Mrudula Didde – 20006086

Gnanasekar Mani – 20020329

Bathula Rajesh - 20024482

Table of Contents

Introduction

Details of the Choose Business Case

Our Resort Management System is designed to streamline the process of booking rooms along with the activities that are offered to guests. The rooms are categorized into single occupancy, double occupancy, luxury suites, glass houses, tents, or tree houses.

Guests can book activities such as hiking with a guide, night campfires, petting zoos, bike or RV rides, and nature-connected trips to engage in during their stay using the system based on the availability. Our system also provides services to modify their food preferences, order food, or order housekeeping during booking.

Additionally, we will be incorporating details of the nearest attractions and bus stops to our resorts using an external Maps API. This enhancement aims to elevate the user experience by providing valuable location-based information.

We have also integrated Admin Management into our tool, which can be utilized by our employees for various purposes ranging from employee views to comprehensive administrative tasks.

The main reason for selecting this use case is that we didn't find any noticeable solution which does all these integrations keeping both customer and admin needs in mind. This is why our solution will be one stop solution and our service provides an end-to-end solution for all of the organizational needs.

Key features will be:

1. Frontend functionality for guests
2. Backend management tool
3. Customer insights
4. Feedback system

5. Nearby attractions

Scope of the Database System

In Scope:

- **Reservation Management:** The system will allow guests to book accommodation, food, and activities, streamlining the reservation process.
- **Guest Service:** Guests can use the service for online check-in/check-out, room service request, concierge service and others.
- **Admin Management:** This is to manage resources such as room inventory, staff scheduling, and inventory management of facilities and equipment.
- **Safety and Security Management:** Industry standard security principles to protect guest information, restrict access to sensitive information
- **Reporting and Analysis:** This system will produce information on occupancy, revenue, guest feedback and other key performance indicators for insights.
- **Navigation System:** Guests will be able to view nearest attractions

Out of Scope:

- **Billing and Payments:** Using third party payment solutions for the billing process and payments.
- **Marketing and Promotion:** Promoting the business through email marketing, loyalty programs and advertising campaigns to attract and retain visitors.
- **Payment Solution:** Using third party payment solutions for bookings.

Business Requirements Assumed

Room Availability:

The rooms should be only booked if they are available for the specified dates and do not overlap with existing bookings for the same.

Package Deals:
If the Guest book more than 3 activities provided while booking, they can get discounted price by 20 to 30% overall price of activities combined.

Special Requests Handling:

Guests will be able to make special requests while booking such as inhouse assistance, room preferences, celebrations, wheelchair request and so much more. This will be handled by using XML data during the booking process.

Cancellation Policy:

Enforcing cancellation rules including deadlines for free cancellation, penalties for late cancellations, and refund processing.

Reservation Constraints:

Enforcing rules such as minimum and maximum length for the stay and number of free children in each booking.

Guest Information Validation:

Guests should be provide information such as Name, DOB, contact information such as Email ID, phone number for the booking.

Naming Convention for Hotel ID:

All the hotel ID should start as HOTELXXXX and should be incremented by 1, for example: HOTEL1001

Naming Convention for Activities ID:

All the activities ID should start as ACTXXXX and should be incremented by 1, for example: ACT1001

Naming Convention for Invoice ID:

All the Invoice ID should start as INVXXXXX and should be incremented by 1, for example: INV10001

Naming Convention for Employee ID:

All the employees ID should start as EMPXXXX and should be incremented by 1, for example: EMP1001

Unique Guest Mail IDs:

All the Guest should have unique mail ID and that is required for the bookings.

Unique Employee Mail IDs:

All the employee Generated Mail ID should be unique.

Child Policy Constraint:

The child with age less than 6 years of age can stay will us for free of cost, however for each room booking max 1 free child is applicable.

Room Booking Trigger:

On reserving the room by the Guest, this trigger will update the Availability of the room by reducing based upon the number of rooms reserved.

Check-Out Process Trigger:

On the scheduled checkout time, this trigger will update the availability based upon the number of reserved rooms.

Cancellation Trigger:

Due to unfortunate circumstances, Guests cancel the booking, the room availability status will be updated accordingly.

Business Rules to be Implemented.

Room Availability:

The rooms should be only booked if they are available for the specified dates and do not overlap with existing bookings for the same, will be done through the Stored Procedure.

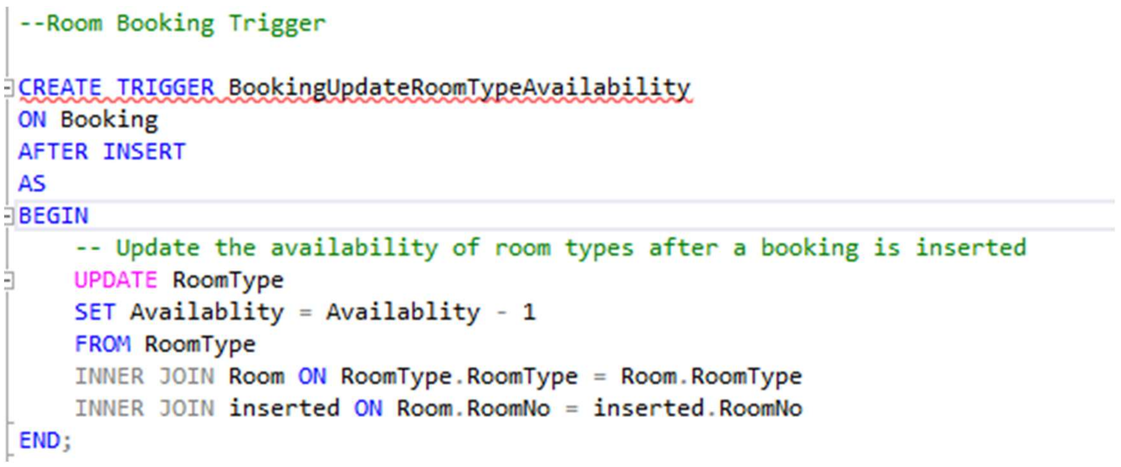
Special Requests Handling:

Guests will be able to make special requests while booking such as inhouse assistance, room preferences, celebrations, wheelchair request and so much more. This will be handled by using XML data during the booking process.

Room Booking Trigger:

On reserving the room by the Guest, this trigger will update the Availability of the room by reducing based upon the number of rooms reserved in the RoomType Table for the specified room.

Code Snippet of the same:



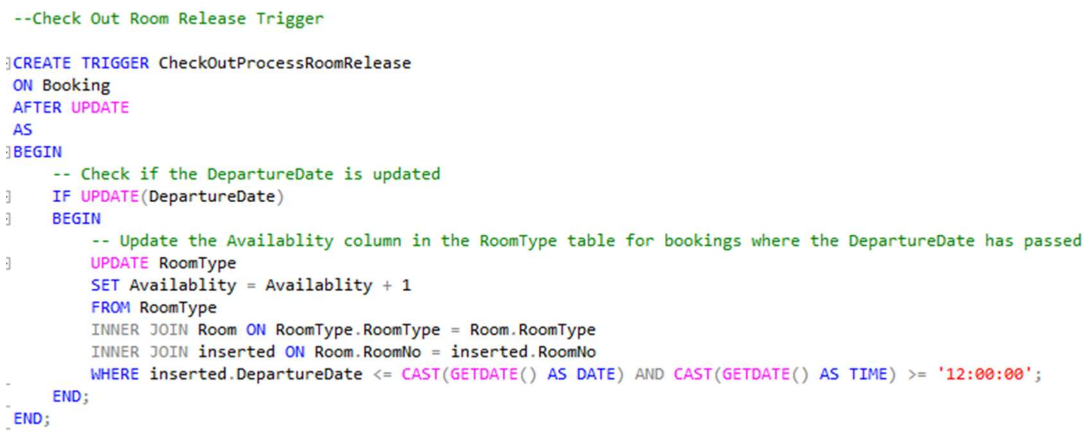
```
--Room Booking Trigger

CREATE TRIGGER BookingUpdateRoomTypeAvailability
ON Booking
AFTER INSERT
AS
BEGIN
    -- Update the availability of room types after a booking is inserted
    UPDATE RoomType
    SET Availablity = Availablity - 1
    FROM RoomType
    INNER JOIN Room ON RoomType.RoomType = Room.RoomType
    INNER JOIN inserted ON Room.RoomNo = inserted.RoomNo
END;
```

Check-Out Process Trigger:

On the scheduled checkout time, this trigger will update the availability based upon the number of reserved rooms in the Room Type table for the specified room.

Code Snipped of the same:



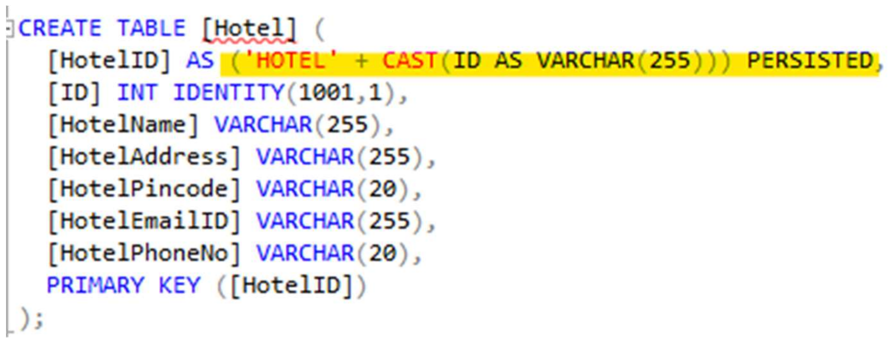
```
--Check Out Room Release Trigger

CREATE TRIGGER CheckOutProcessRoomRelease
ON Booking
AFTER UPDATE
AS
BEGIN
    -- Check if the DepartureDate is updated
    IF UPDATE(DepartureDate)
    BEGIN
        -- Update the Availablity column in the RoomType table for bookings where the DepartureDate has passed
        UPDATE RoomType
        SET Availablity = Availablity + 1
        FROM RoomType
        INNER JOIN Room ON RoomType.RoomType = Room.RoomType
        INNER JOIN inserted ON Room.RoomNo = inserted.RoomNo
        WHERE inserted.DepartureDate <= CAST(GETDATE() AS DATE) AND CAST(GETDATE() AS TIME) >= '12:00:00';
    END;
END;
```

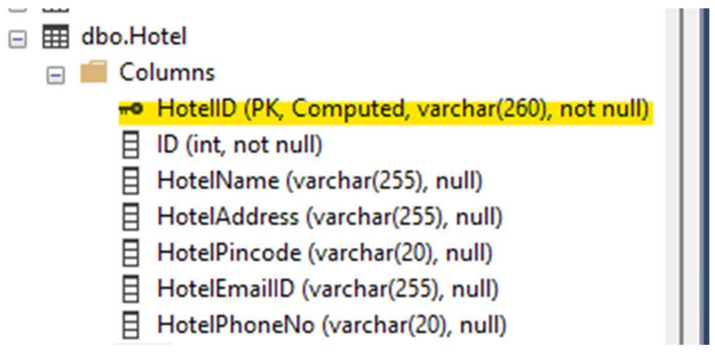
Naming Convention for Hotel ID:

All the hotel ID should start as HOTELXXXX and should be incremented by 1, for example: HOTEL1001

Code Snipped of the Same:



```
CREATE TABLE [Hotel] (
  [HotelID] AS ('HOTEL' + CAST(ID AS VARCHAR(255))) PERSISTED,
  [ID] INT IDENTITY(1001,1),
  [HotelName] VARCHAR(255),
  [HotelAddress] VARCHAR(255),
  [HotelPincode] VARCHAR(20),
  [HotelEmailID] VARCHAR(255),
  [HotelPhoneNo] VARCHAR(20),
  PRIMARY KEY ([HotelID])
);
```

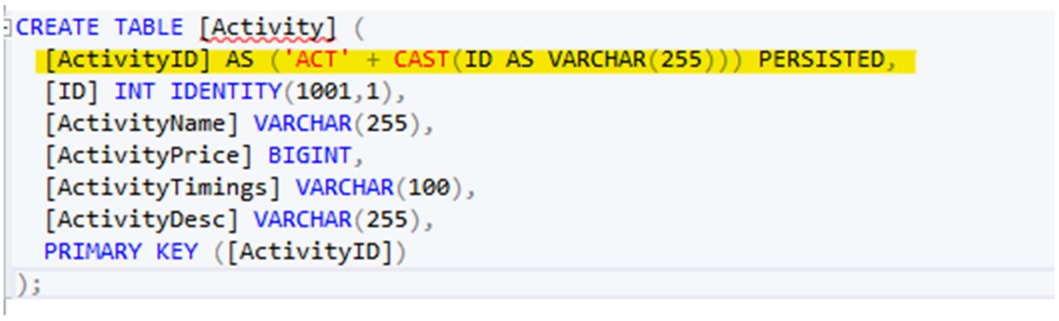


```
dbo.Hotel
  Columns
    HotelID (PK, Computed, varchar(260), not null)
    ID (int, not null)
    HotelName (varchar(255), null)
    HotelAddress (varchar(255), null)
    HotelPincode (varchar(20), null)
    HotelEmailID (varchar(255), null)
    HotelPhoneNo (varchar(20), null)
```

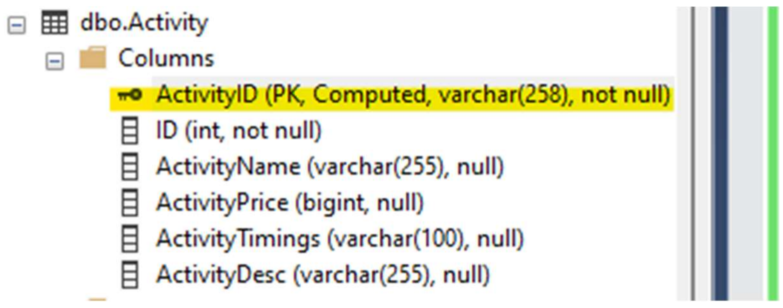
Naming Convention for Activities ID:

All the activities ID should start as ACTXXXX and should be incremented by 1, for example: ACT1001

Code Snippet of the Same:



```
CREATE TABLE [Activity] (
  [ActivityID] AS ('ACT' + CAST(ID AS VARCHAR(255))) PERSISTED,
  [ID] INT IDENTITY(1001,1),
  [ActivityName] VARCHAR(255),
  [ActivityPrice] BIGINT,
  [ActivityTimings] VARCHAR(100),
  [ActivityDesc] VARCHAR(255),
  PRIMARY KEY ([ActivityID])
);
```

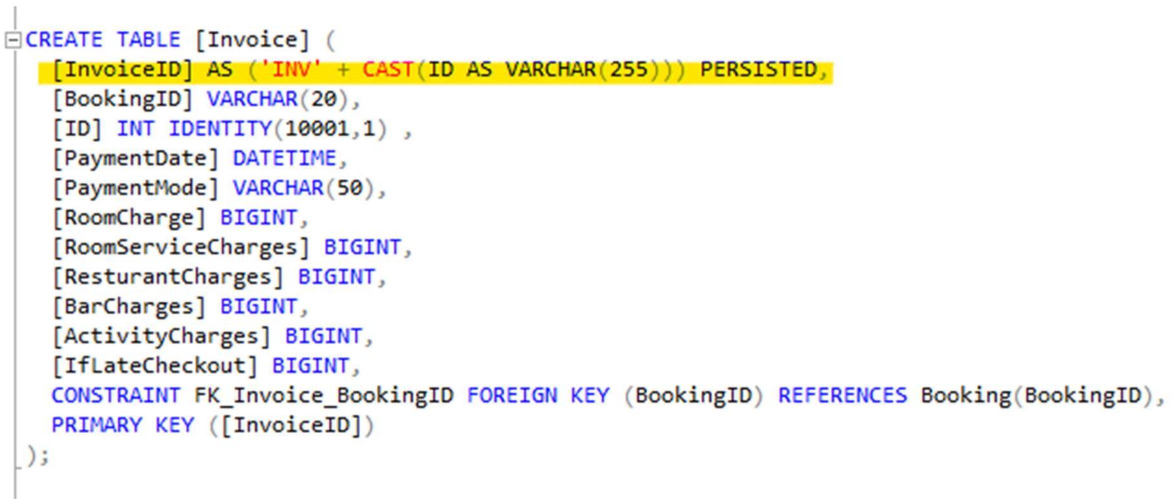


```
dbo.Activity
  Columns
    ActivityID (PK, Computed, varchar(258), not null)
    ID (int, not null)
    ActivityName (varchar(255), null)
    ActivityPrice (bigint, null)
    ActivityTimings (varchar(100), null)
    ActivityDesc (varchar(255), null)
```

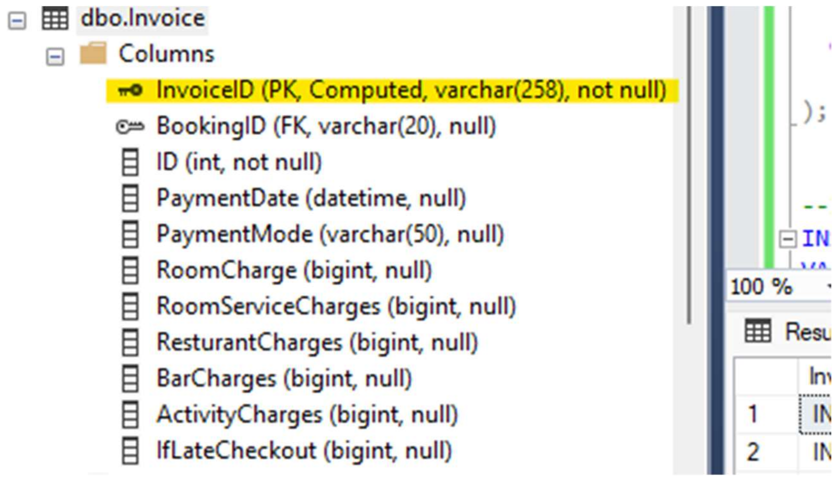
Naming Convention for Invoice ID:

All the Invoice ID should start as INVXXXXX and should be incremented by 1, for example: INV10001

Code Snippet of the same:



```
CREATE TABLE [Invoice] (
  [InvoiceID] AS ('INV' + CAST(ID AS VARCHAR(255))) PERSISTED,
  [BookingID] VARCHAR(20),
  [ID] INT IDENTITY(10001,1) ,
  [PaymentDate] DATETIME,
  [PaymentMode] VARCHAR(50),
  [RoomCharge] BIGINT,
  [RoomServiceCharges] BIGINT,
  [ResturantCharges] BIGINT,
  [BarCharges] BIGINT,
  [ActivityCharges] BIGINT,
  [IfLateCheckout] BIGINT,
  CONSTRAINT FK_Invoice_BookingID FOREIGN KEY (BookingID) REFERENCES Booking(BookingID),
  PRIMARY KEY ([InvoiceID])
);
```

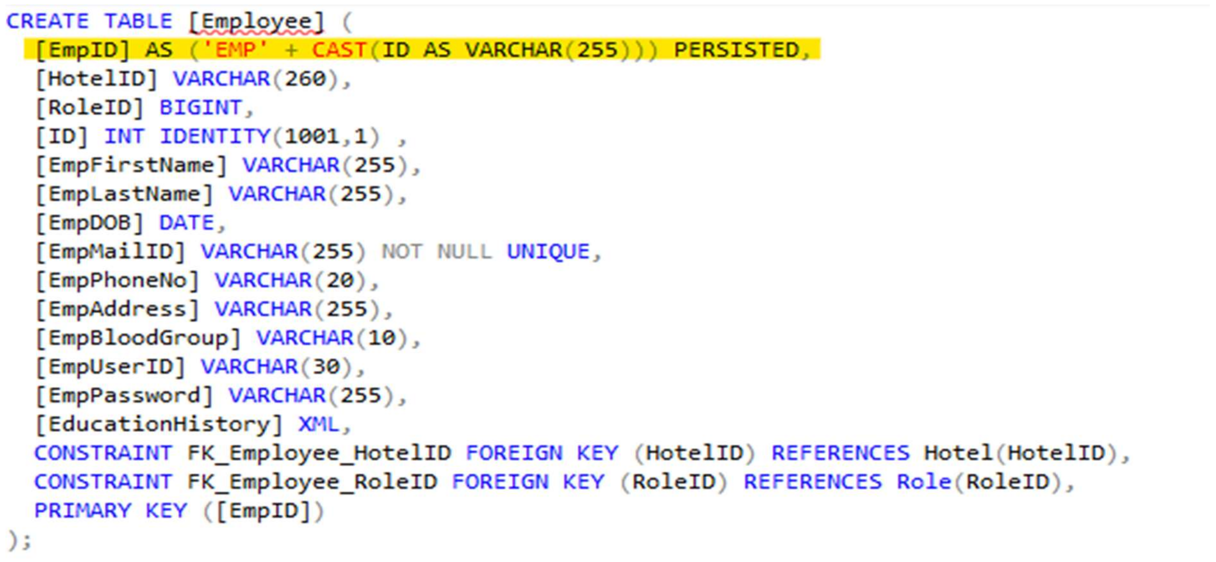


dbo.Invoice
- Columns
  - InvoiceID (PK, Computed, varchar(258), not null)
  - BookingID (FK, varchar(20), null)
  - ID (int, not null)
  - PaymentDate (datetime, null)
  - PaymentMode (varchar(50), null)
  - RoomCharge (bigint, null)
  - RoomServiceCharges (bigint, null)
  - ResturantCharges (bigint, null)
  - BarCharges (bigint, null)
  - ActivityCharges (bigint, null)
  - IfLateCheckout (bigint, null)

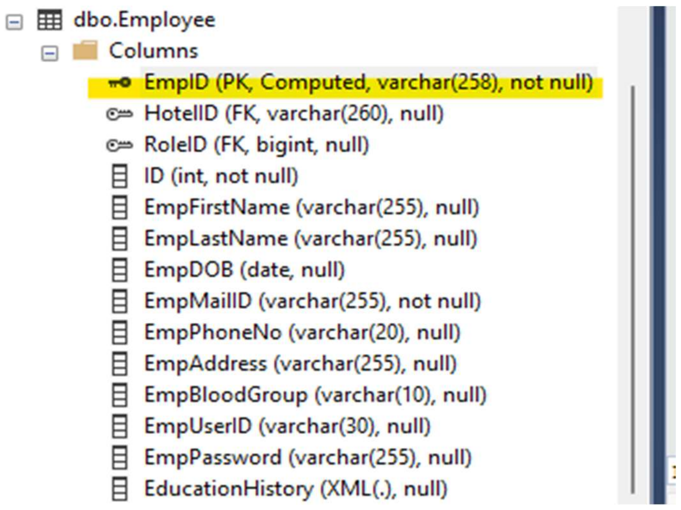
Naming Convention for Employee ID:

All the employees ID should start as EMPXXXX and should be incremented by 1, for example: EMP1001

Code Snippet of the same:



```sql
CREATE TABLE [Employee] (
  [EmpID] AS ('EMP' + CAST(ID AS VARCHAR(255))) PERSISTED,
  [HotelID] VARCHAR(260),
  [RoleID] BIGINT,
  [ID] INT IDENTITY(1001,1) ,
  [EmpFirstName] VARCHAR(255),
  [EmpLastName] VARCHAR(255),
  [EmpDOB] DATE,
  [EmpMailID] VARCHAR(255) NOT NULL UNIQUE,
  [EmpPhoneNo] VARCHAR(20),
  [EmpAddress] VARCHAR(255),
  [EmpBloodGroup] VARCHAR(10),
  [EmpUserID] VARCHAR(30),
  [EmpPassword] VARCHAR(255),
  [EducationHistory] XML,
  CONSTRAINT FK_Employee_HotelID FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID),
  CONSTRAINT FK_Employee_RoleID FOREIGN KEY (RoleID) REFERENCES Role(RoleID),
  PRIMARY KEY ([EmpID])
);
```

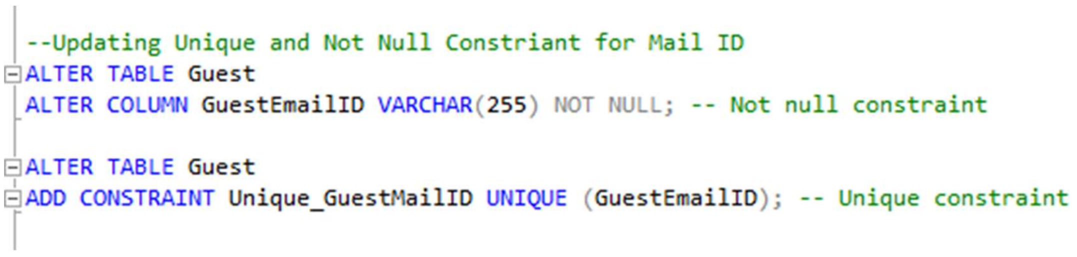


```
dbo.Employee
  Columns
    EmpID (PK, Computed, varchar(258), not null)
    HotelID (FK, varchar(260), null)
    RoleID (FK, bigint, null)
    ID (int, not null)
    EmpFirstName (varchar(255), null)
    EmpLastName (varchar(255), null)
    EmpDOB (date, null)
    EmpMailID (varchar(255), not null)
    EmpPhoneNo (varchar(20), null)
    EmpAddress (varchar(255), null)
    EmpBloodGroup (varchar(10), null)
    EmpUserID (varchar(30), null)
    EmpPassword (varchar(255), null)
    EducationHistory (XML(.), null)
```

Unique Guest Mail IDs:

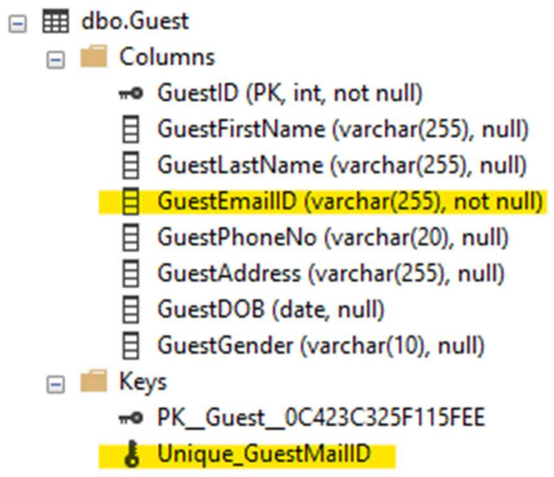
All the Guest should have unique mail ID and it should not be null and that is required for the bookings.

Code Snippet:



```sql
--Updating Unique and Not Null Constriant for Mail ID
ALTER TABLE Guest
ALTER COLUMN GuestEmailID VARCHAR(255) NOT NULL; -- Not null constraint

ALTER TABLE Guest
ADD CONSTRAINT Unique_GuestMailID UNIQUE (GuestEmailID); -- Unique constraint
```

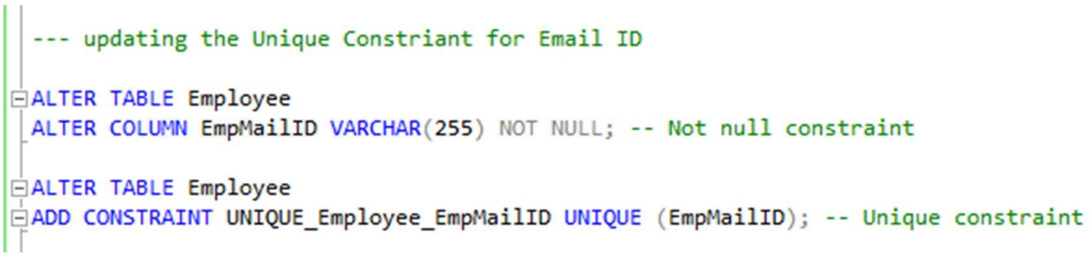


```
dbo.Guest
  Columns
    GuestID (PK, int, not null)
    GuestFirstName (varchar(255), null)
    GuestLastName (varchar(255), null)
    GuestEmailID (varchar(255), not null)
    GuestPhoneNo (varchar(20), null)
    GuestAddress (varchar(255), null)
    GuestDOB (date, null)
    GuestGender (varchar(10), null)
  Keys
    PK__Guest__0C423C325F115FEE
    Unique_GuestMailID
```

Unique Employee Mail IDs:

All the employee Generated Mail ID should be unique.

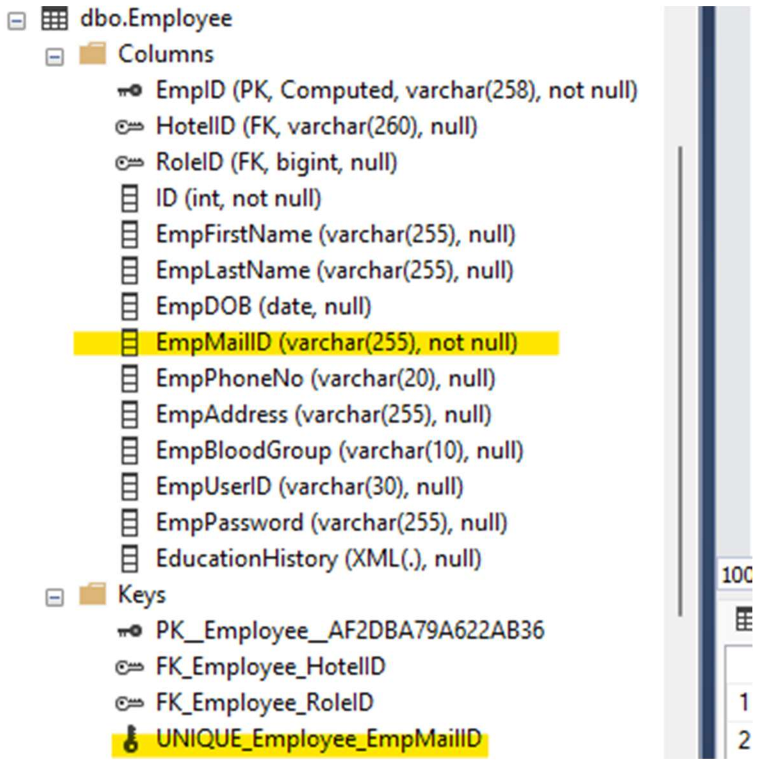
Code Snippet:



```sql
--- updating the Unique Constriant for Email ID

ALTER TABLE Employee
ALTER COLUMN EmpMailID VARCHAR(255) NOT NULL; -- Not null constraint

ALTER TABLE Employee
ADD CONSTRAINT UNIQUE_Employee_EmpMailID UNIQUE (EmpMailID); -- Unique constraint
```



```
dbo.Employee
  Columns
    EmpID (PK, Computed, varchar(258), not null)
    HotelID (FK, varchar(260), null)
    RoleID (FK, bigint, null)
    ID (int, not null)
    EmpFirstName (varchar(255), null)
    EmpLastName (varchar(255), null)
    EmpDOB (date, null)
    EmpMailID (varchar(255), not null)
    EmpPhoneNo (varchar(20), null)
    EmpAddress (varchar(255), null)
    EmpBloodGroup (varchar(10), null)
    EmpUserID (varchar(30), null)
    EmpPassword (varchar(255), null)
    EducationHistory (XML(.), null)
  Keys
    PK__Employee__AF2DBA79A622AB36
    FK_Employee_HotelID
    FK_Employee_RoleID
    UNIQUE_Employee_EmpMailID
```

Relational Schema – 3NF, Hybrid Database

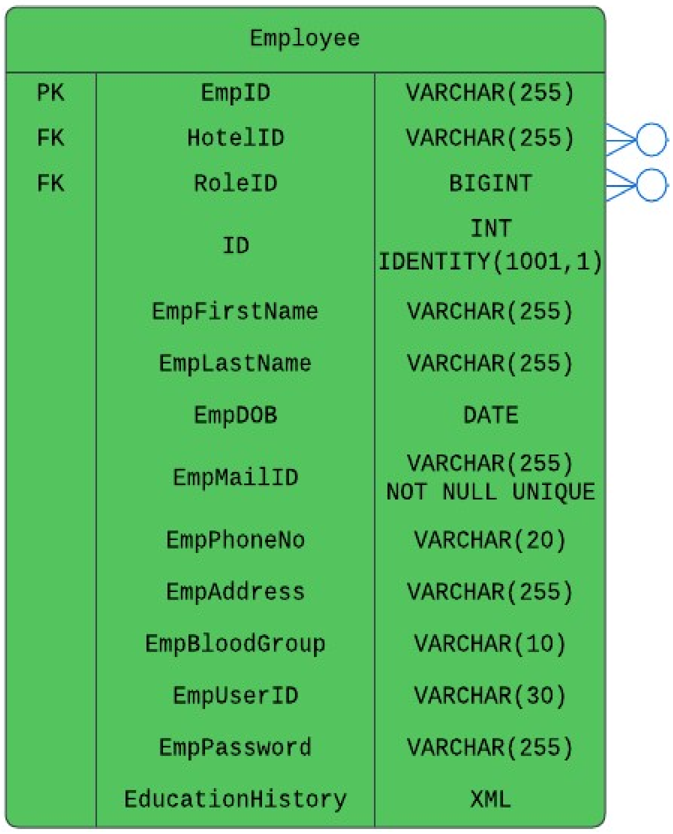
Relational Scheme – 3NF:

Employee Table:

This table is used to store all the information related to the Employee, it's Primary key is Employee ID (EmpID), and it has two Foreign Keys RoleID and HotelID. Here, RoleID is generated in Role Table and HotelID is generated in the Hotel Table. And the EmpID will be created automatically based upon the intentity business rule created, for example: EMP1001, EMP1002

Also, one more information of the Employee is Education History(EducationHistory) which will be saved in this table as the XML data which makes our DB Hybrid as well.

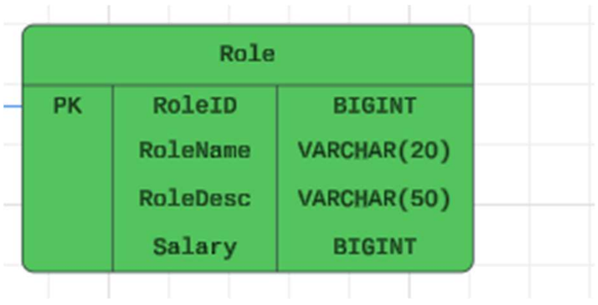
Note: We have one more business rule constraint where the Employee Mail ID(EmpMailID) will not be null and the same will be unique.



Employee		
PK	EmpID	VARCHAR(255)
FK	HotelID	VARCHAR(255)
FK	RoleID	BIGINT
	ID	INT IDENTITY(1001,1)
	EmpFirstName	VARCHAR(255)
	EmpLastName	VARCHAR(255)
	EmpDOB	DATE
	EmpMailID	VARCHAR(255) NOT NULL UNIQUE
	EmpPhoneNo	VARCHAR(20)
	EmpAddress	VARCHAR(255)
	EmpBloodGroup	VARCHAR(10)
	EmpUserID	VARCHAR(30)
	EmpPassword	VARCHAR(255)
	EducationHistory	XML

Role Table:

The role table gives the details about the role such as Name, Salary, description. This table has only one Primary Key Role ID(RoleID) and it isn't auto generated.

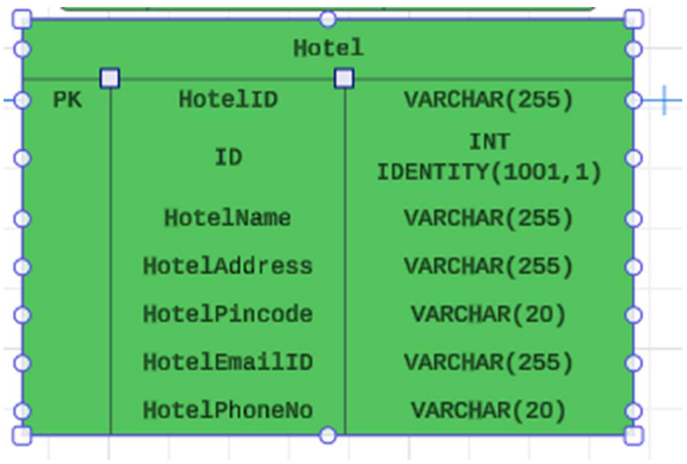


Role		
PK	RoleID	BIGINT
	RoleName	VARCHAR(20)
	RoleDesc	VARCHAR(50)
	Salary	BIGINT

Hotel Table:

This table gives the overall information about the Hotel/ resort. The information includes HotelID which is also the Primary key of the table, Hotel Name, address, Pincode, Email address, Phone number.

Here, The hotel ID is auto generated as per the business rule as HOTELXXXX, for example: HOTEL1001, HOTEL1002.

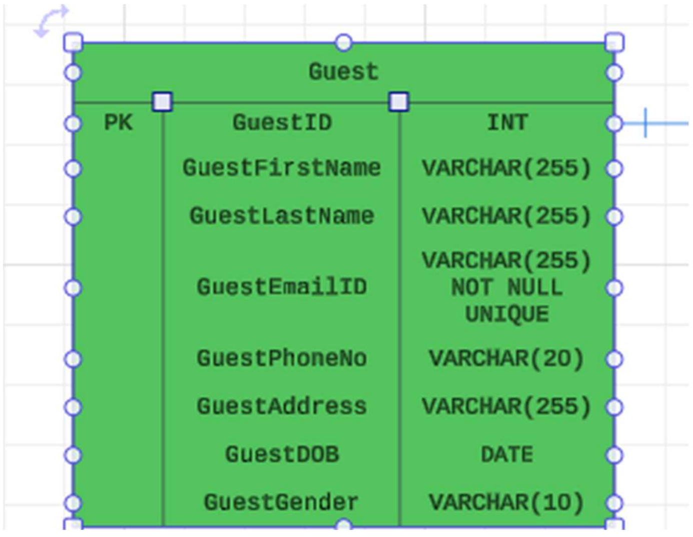


Hotel		
PK	HotelID	VARCHAR(255)
	ID	INT IDENTITY(1001,1)
	HotelName	VARCHAR(255)
	HotelAddress	VARCHAR(255)
	HotelPincode	VARCHAR(20)
	HotelEmailID	VARCHAR(255)
	HotelPhoneNo	VARCHAR(20)

Guest Table:

This Table gives the details of the Guest who books our Resort for the stay. This table will be useful to gather information about Guest ID, Name, Email, Phone Number, Address, Date of Birth, Gender. Here, Guest ID acts as the Primary Key for this table.

Also, we have implemented business rule for Guest Email ID stating that the email ID should be unique and the same shouldn't be null.

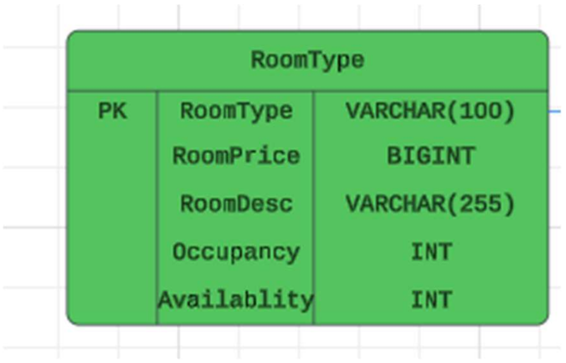


Guest		
PK	GuestID	INT
	GuestFirstName	VARCHAR(255)
	GuestLastName	VARCHAR(255)
	GuestEmailID	VARCHAR(255) NOT NULL UNIQUE
	GuestPhoneNo	VARCHAR(20)
	GuestAddress	VARCHAR(255)
	GuestDOB	DATE
	GuestGender	VARCHAR(10)

Room Type Table:

The Room Type table gives the details about the types of Rooms available in our Resort. Also, Room Type(RoomType) is the Primary Key of the table.

This is one of the important table as the vacancies in each type of the room information is present here. Also, we will be able to gather information of Room Price, Description, Occupancy can be found.

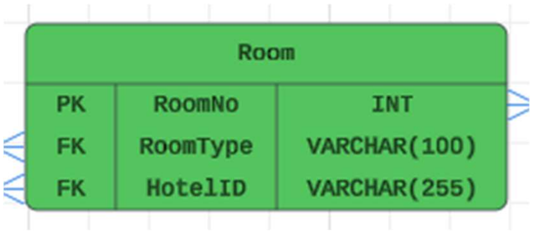


RoomType		
PK	RoomType	VARCHAR(100)
	RoomPrice	BIGINT
	RoomDesc	VARCHAR(255)
	Occupancy	INT
	Availablity	INT

Room Table:

Room table acts as the bridge between RoomType Table and the Hotel Table, using the Room Number we will be able to do the booking for the Guest. This table Primarily helps us to pin point the booking of the Guest in which Hotel and the type of room that has been booked.

Here, the Primary key will be Room Number(RoomNo) and there are two foreign keys RoomType and HotelID.

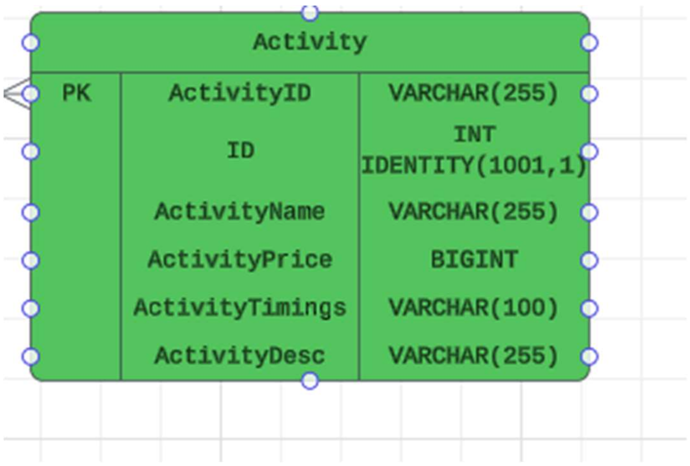


Room		
PK	RoomNo	INT
FK	RoomType	VARCHAR(100)
FK	HotelID	VARCHAR(255)

Activity Table:

This table will be able to retrieve information regarding the activities that are available in our resort and it gives us the information about the Name, Price, Timings and description of the activities.

Here, ActivityID acts as the Primary key for the table and the same is Autogenerated as per the Business rule like ACT1001, ACT1002.



Activity		
PK	ActivityID	VARCHAR(255)
	ID	INT IDENTITY(1001,1)
	ActivityName	VARCHAR(255)
	ActivityPrice	BIGINT
	ActivityTimings	VARCHAR(100)
	ActivityDesc	VARCHAR(255)

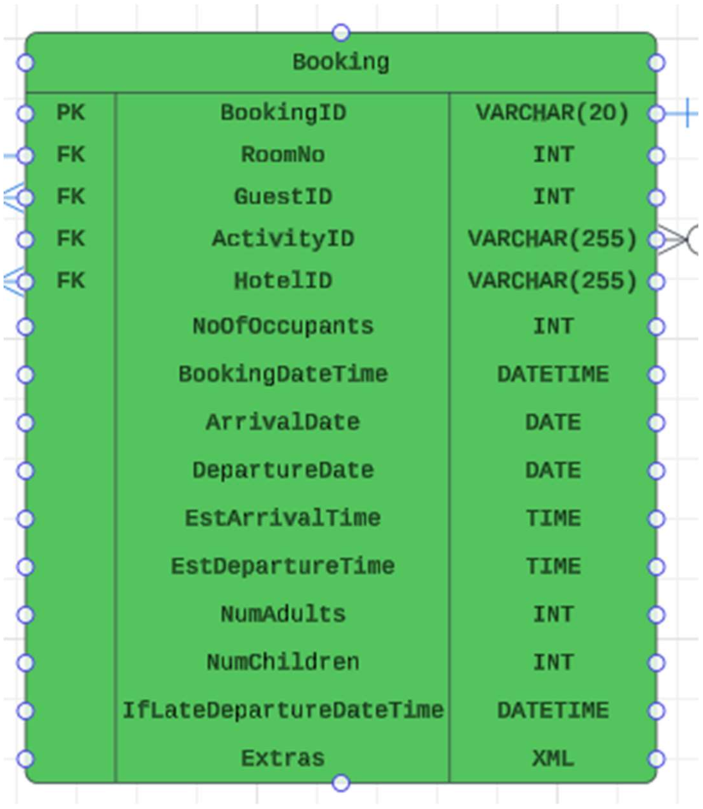
Booking Table:

This is one of the important table in our System and it acts like the center interface where most of the other tables work on this one. Here, we can find all the details regarding the booking from Booking ID, GuestID, RoomNo, HotelID, Number of Occupants, BookingDate, Arrival Date, Departure Date and time, Number of Occupants also one more important information such as Extras.

Here, Extras is the XML datatype where the Guest can add some extra requirements like inhouse assistance, room preferences, celebrations, wheelchair request and so much more.

And The BookingID acts as the Primary key in this table, with 4 Foreign Keys such as RoomNo, GuestID, ActivityID, HotelID.

This table also includes two triggers which acts automatically when there is a insert/ update to the table, which updates the room availability as per the bookings done.

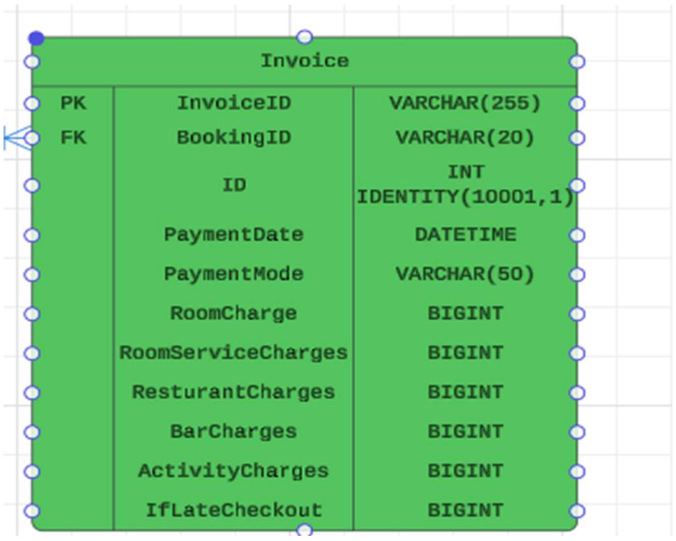


	Booking	
PK	BookingID	VARCHAR(20)
FK	RoomNo	INT
FK	GuestID	INT
FK	ActivityID	VARCHAR(255)
FK	HotelID	VARCHAR(255)
	NoOfOccupants	INT
	BookingDateTime	DATETIME
	ArrivalDate	DATE
	DepartureDate	DATE
	EstArrivalTime	TIME
	EstDepartureTime	TIME
	NumAdults	INT
	NumChildren	INT
	IfLateDepartureDateTime	DATETIME
	Extras	XML

Invoice Table:

This table comes into picture at the last stage during the billing of the Guest. The table gives the details of the total cost of the booking fetching the value through the other tables based upon the BookingID and gives the details of Payment Date, Mode and other information.

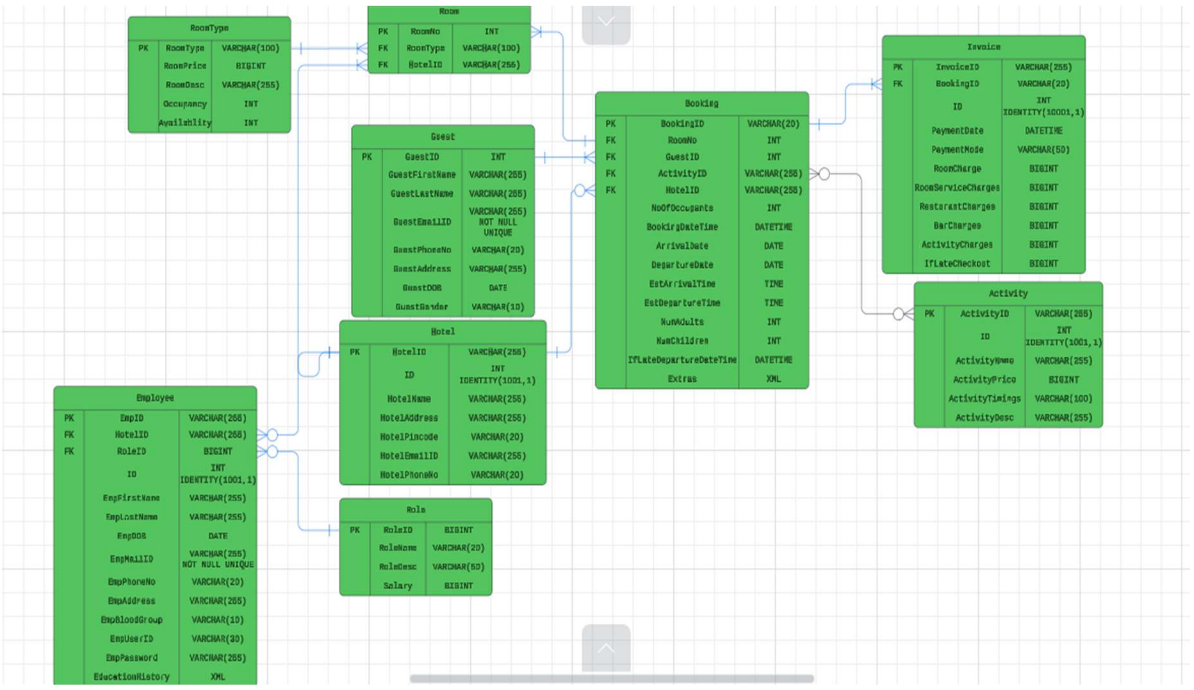
Here, InvoiceID is the Primary key and BookingID is the Foreign key of the table. Also, the InvoiceID will be auto generated as per business rule like INV10001, INV10002.



	Invoice	
PK	InvoiceID	VARCHAR(255)
FK	BookingID	VARCHAR(20)
	ID	INT IDENTITY(10001,1)
	PaymentDate	DATETIME
	PaymentMode	VARCHAR(50)
	RoomCharge	BIGINT
	RoomServiceCharges	BIGINT
	ResturantCharges	BIGINT
	BarCharges	BIGINT
	ActivityCharges	BIGINT
	IfLateCheckout	BIGINT

Entity Relationship Diagram (ERD)

Link of the ERD Diagram through Lucid Chart: [AdvancedDB_Resort_CA1: Lucidchart]



RoomType		
PK	RoomType	VARCHAR(100)
	RoomPrice	BIGINT
	RoomDesc	VARCHAR(255)
	Occupancy	INT
	Availability	INT

Room		
PK	RoomNo	INT
FK	RoomType	VARCHAR(100)
FK	HotelID	VARCHAR(256)

Guest		
PK	GuestID	INT
	GuestFirstName	VARCHAR(255)
	GuestLastName	VARCHAR(255)
	GuestEmailID	VARCHAR(255) NOT NULL UNIQUE
	GuestPhoneNo	VARCHAR(20)
	GuestAddress	VARCHAR(255)
	GuestDOB	DATE
	GuestGender	VARCHAR(10)

Booking		
PK	BookingID	VARCHAR(20)
FK	RoomNo	INT
FK	GuestID	INT
FK	ActivityID	VARCHAR(256)
FK	HotelID	VARCHAR(256)
	NoOfOccupants	INT
	BookingDateTime	DATETIME
	ArrivalDate	DATE
	DepartureDate	DATE
	EstArrivalTime	TIME
	EstDepartureTime	TIME
	NumAdults	INT
	NumChildren	INT
	IfLateDepartureDateTime	DATETIME
	Extras	XML

Invoice		
PK	InvoiceID	VARCHAR(255)
FK	BookingID	VARCHAR(20)
	ID	INT IDENTITY(10001,1)
	PaymentDate	DATETIME
	PaymentMode	VARCHAR(50)
	RoomCharge	BIGINT
	RoomServiceCharges	BIGINT
	RestarantCharges	BIGINT
	BarCharges	BIGINT
	ActivityCharges	BIGINT
	IfLateCheckout	BIGINT

Activity		
PK	ActivityID	VARCHAR(255)
	ID	INT IDENTITY(1001,1)
	ActivityName	VARCHAR(255)
	ActivityPrice	BIGINT
	ActivityTimings	VARCHAR(100)
	ActivityDesc	VARCHAR(255)

Hotel		
PK	HotelID	VARCHAR(256)
	ID	INT IDENTITY(1001,1)
	HotelName	VARCHAR(255)
	HotelAddress	VARCHAR(255)
	HotelPincode	VARCHAR(20)
	HotelEmailID	VARCHAR(255)
	HotelPhoneNo	VARCHAR(20)

Employee		
PK	EmpID	VARCHAR(256)
FK	HotelID	VARCHAR(256)
FK	RoleID	BIGINT
	ID	INT IDENTITY(1001,1)
	EmpFirstName	VARCHAR(255)
	EmpLastName	VARCHAR(255)
	EmpDOB	DATE
	EmpMailID	VARCHAR(255) NOT NULL UNIQUE
	EmpPhoneNo	VARCHAR(20)
	EmpAddress	VARCHAR(255)
	EmpBloodGroup	VARCHAR(10)
	EmpUserID	VARCHAR(30)
	EmpPassword	VARCHAR(255)
	EducationHistory	XML

Role		
PK	RoleID	BIGINT
	RoleName	VARCHAR(20)
	RoleDesc	VARCHAR(50)
	Salary	BIGINT

XML Schema

In a Hybrid Database, XML Datatype is used to store huge amount of information in the XML format. And this can be useful to store and manage large amount of data in unstructured or semi-structured type.

In our system we are incorporating XML datatype in two of our tables namely Employee and Booking Table.

In Employee Table, the column EducationHistory is of datatype XML, where we can store the details of Education History of each employee, the main reason to make it as XML for this column is due to its variation, each employee can be of different levels and education backgrounds if it's of datatype XML we can easily accommodate this un structured data this makes one of the best options to make it XML rather than the usual one.

In the Booking Table, the column Extras is of datatype XML, here, the Guests can choose from many different variations of the extra services such as inhouse assistance, room preferences, celebrations, wheelchair request and so much more. Due to this availability this column makes one of the best candidate for being XML datatype rather than the usual one.

Conclusion

Our resort management system is well-Structured State of the art end to end system to accommodate booking of resort, hotels, activities, Admin Management and Guest Management. Our system also provides efficient data management and retrieval capabilities due to the way it's built.

And our system is built as per 3NF principles also includes XML datatype making it hybrid database.

And due to the presence of the Triggers and constraints in our systems making it future proof and requires very less maintenance.

Although it lacks some of the features such as end-to-end payroll management, customer feedback, marketing features, due to its core basic features integrated perfectly, the above mentioned features can be added as required in the future.

To conclude our resort management system is the one stop solution for your database needs for the resort business.

Individual Contributions

Gnanasekar Mani – 20020329

Creation of tables and databases in the SQL Server Management Studio (SSMS) to lay foundation of the Resort Management System.

Worked extensively on the creation of the Entity relationship diagram.

Will be creating the Insert queries for all the tables to evaluate the systems.

Will be working on the creation of the triggers and creation of constraints as per the business rules obtained or observed.

Contributed to the creation of the report on various stages, co-ordinated and created the final draft of the report.

Mrudula Didde – 20006086

Came up with the use-case for the resort management system with the addition to integrate the Activity management in one solution.

Have defined most of the Business Rules and concluded which are the business rules should we implement as per industry requirements.

Will be creating the Stored procedures as per the requirements.

Will be working on the creation of the views.

Will be working on creation of the test data for testing our systems to be foul proof.

Contributed to the creation of the report by giving mindful insights.

Bathula Rajesh – 20024482

Cameup with the names for all the tables and worked on the each of the table details, column information along with the data type.

Worked on the creation of the relationships between different tables.

Will be working on creation of the XML data for the columns required through the creation of the Stored Procedure.

Shown aptitude at designing schemas and modeling databases.

Contributed to the report creation.

Bibliography

Websites used for References:

Wikipedia: XML database - Wikipedia

Tutorials Point: XML - Databases (tutorialspoint.com)

Geeks for Geeks: SQL Tutorial - GeeksforGeeks

Wikipedia: Identity column - Wikipedia

References: Data Model for a Hotel Management System | Vertabelo Database Modeler

Software used:

Microsoft SQL Server: SQL Server Downloads | Microsoft

Microsoft SQL Server Management Studio: Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn

Notepad ++: Downloads | Notepad++ (notepad-plus-plus.org)

Microsoft Word: Free Online Document Editing with Microsoft Word | Microsoft 365

Lucid Chart: Intelligent Diagramming | Lucidchart