

# Data Dictionaries

A **data dictionary** is a centralised repository of information about the data within a database or information system. It serves as a reference guide for developers, analysts, and stakeholders, providing clarity on what data is stored, how it's structured, and how it should be used.

## What is a Data Dictionary?

A **data dictionary** outlines and defines the properties of data elements in a database. It typically includes:

- **Table and Column Names:** Every entity and attribute used in the system.
- **Data Types:** Specifications of each attribute's type (e.g., VARCHAR, INT, DATE).
- **Descriptions:** Details explaining each attribute's purpose and usage.
- **Constraints:** Rules applied to the data, such as primary keys, foreign keys, and restrictions (e.g., NOT NULL).
- **Default Values:** Any default settings applied when no data is provided for an attribute.

In other words, a data dictionary is a **blueprint for data management**, helping maintain data consistency, streamline communication, and prevent errors during database interaction.

## Why is a Data Dictionary Important?

For a hotel booking system, a data dictionary helps:

- **Maintain Consistency:** It ensures that data is entered, stored, and retrieved in a uniform way across the system.
- **Facilitate Data Integration:** Helps developers and database administrators understand how data tables relate, making it easier to scale or modify the database.
- **Enhance Collaboration:** Stakeholders can reference the data dictionary to understand the database structure, easing communication between teams.
- **Improve Data Quality:** By defining constraints and acceptable values, a data dictionary reduces the risk of erroneous data entry.

# Steps to Create a Data Dictionary for a Hotel Booking System

## Step 1: Identify Key Tables

To begin, list the tables (or entities) necessary for the hotel booking system. For example:

- **Customer**
- **Room**
- **Booking**
- **Payment**
- **Hotel**

Each table represents an entity within the system, with columns (attributes) holding specific information about that entity.

## Step 2: Define Attributes for Each Table

Next, list the attributes within each table, along with details such as data types and descriptions. For example, here are potential attributes for each table:

### 1. Customer Table

- **CustomerID**: INT – Primary Key, unique identifier for each customer.
- **Name**: VARCHAR(100) – The full name of the customer.
- **Email**: VARCHAR(100) – Customer's email address.
- **PhoneNumber**: VARCHAR(15) – Contact phone number for the customer.

### 2. Room Table

- **RoomID**: INT – Primary Key, unique identifier for each room.
- **RoomType**: VARCHAR(50) – Type of room (e.g., Single, Double, Suite).
- **PricePerNight**: DECIMAL(8, 2) – Nightly rate for the room.
- **RoomStatus**: VARCHAR(20) – Availability status (e.g., Available, Booked).

### 3. Booking Table

- **BookingID**: INT – Primary Key, unique identifier for each booking.
- **CustomerID**: INT – Foreign Key linking to Customer, identifies the customer making the booking.

- **RoomID**: INT – Foreign Key linking to Room, indicates the room being booked.
- **BookingDate**: DATE – Date when the booking was made.
- **CheckInDate**: DATE – Customer's check-in date.
- **CheckOutDate**: DATE – Customer's check-out date.

#### 4. Payment Table

- **PaymentID**: INT – Primary Key, unique identifier for each payment.
- **BookingID**: INT – Foreign Key linking to Booking, identifies the associated booking.
- **Amount**: DECIMAL(10, 2) – Total payment amount.
- **PaymentDate**: DATE – Date the payment was made.
- **PaymentMethod**: VARCHAR(50) – Method of payment (e.g., Credit Card, Cash).

#### 5. Hotel Table

- **HotelID**: INT – Primary Key, unique identifier for each hotel.
- **Name**: VARCHAR(100) – Name of the hotel.
- **Location**: VARCHAR(100) – Physical location of the hotel.
- **Phone**: VARCHAR(15) – Contact number for the hotel.

### Step 3: Define Data Types and Constraints

For each attribute, specify the data type and any constraints. This step involves designating primary and foreign keys, defining any NOT NULL constraints, and setting up unique constraints where necessary. For instance:

- **CustomerID** in the Customer table is a primary key (PK), NOT NULL, and should be unique.
- **RoomStatus** in the Room table should have a constraint to accept only predefined statuses (e.g., "Available" or "Booked").

### Step 4: Document the Relationships

In a data dictionary, relationships between tables are typically noted, especially when foreign keys are involved. For example:

- **CustomerID** in the Booking table is a foreign key referencing **CustomerID** in the Customer table, representing a one-to-many relationship.

- **RoomID** in the Booking table is a foreign key referencing **RoomID** in the Room table, indicating a one-to-many relationship with rooms.

### Step 5: Create the Data Dictionary Table

A data dictionary is often represented in a table format to allow for easy reading. Here's a sample layout for the data dictionary of the hotel booking system:

Table	Attribute	Data Type	Description	Constraints
<b>Customer</b>	CustomerID	INT	Unique identifier for each customer	PK, NOT NULL, Unique
	Name	VARCHAR(100)	Customer's full name	NOT NULL
	Email	VARCHAR(100)	Customer's email address	NOT NULL
	PhoneNumber	VARCHAR(15)	Contact number for customer	
<b>Room</b>	RoomID	INT	Unique identifier for each room	PK, NOT NULL, Unique
	RoomType	VARCHAR(50)	Type of room	NOT NULL
	PricePerNight	DECIMAL(8, 2)	Price per night for the room	
	RoomStatus	VARCHAR(20)	Availability status (e.g., Available)	CHECK(RoomStatus in ...)
<b>Booking</b>	BookingID	INT	Unique identifier for each booking	PK, NOT NULL, Unique
	CustomerID	INT	ID of the customer who made the booking	FK, NOT NULL, References Customer(CustomerID)

Table	Attribute	Data Type	Description	Constraints
	RoomID	INT	ID of the room booked	FK, NOT NULL, References Room(RoomID)
	BookingDate	DATE	Date of booking	NOT NULL
	CheckInDate	DATE	Check-in date	NOT NULL
	CheckOutDate	DATE	Check-out date	NOT NULL
<b>Payment</b>	PaymentID	INT	Unique identifier for each payment	PK, NOT NULL, Unique
	BookingID	INT	ID of the associated booking	FK, NOT NULL, References Booking(BookingID)
	Amount	DECIMAL(10, 2)	Total payment amount	
	PaymentDate	DATE	Date the payment was made	NOT NULL
<b>Hotel</b>	HotelID	INT	Unique identifier for each hotel	PK, NOT NULL, Unique
	Name	VARCHAR(100)	Hotel name	NOT NULL
	Location	VARCHAR(100)	Hotel location	
	Phone	VARCHAR(15)	Contact number	

## Conclusion

A data dictionary provides clarity and consistency for data handling in a system by documenting each data element's purpose, structure, and constraints. By establishing a comprehensive data dictionary, you ensure that the system can handle user interactions and information consistently, promoting efficient collaboration, reliable data integrity, and a smoother database management process.