**PRODUCT DESCRIPTION**

|  |  |
| --- | --- |
| **Project:** | **Database Design & Implementation**  **Experimental Case** |
| Date: 07-12-2022 |  |
| **Product Title: Hotel Management** |  |
| Authors:  Diljot Singh Narula,  Pardeep Kaur,  Harmanpreet Kaur,  Randeep Singh |  |
|  |
|  |
|  |
|  |
|  |  |
| Client: | INFO 2312 – KPU |
| Document Ref: | Product description |
| Version No: | 0.1 |

# PRODUCT DESCRIPTION HISTORY

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision date** | **Previous revision date** | **Summary of Changes** | **Changes marked** |
|  |  | First issue |  |

## Distribution

This document has been distributed to:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Title** | **Date of Issue** | **Version** |
| Cesar Lopez Castellanos | Instructor | 09-12-2022 | 0.1 |

**Table of Contents**

*page*

[1. PRODUCT DESCRIPTION HISTORY 2](#_Toc499424668)

[1.1. Revision History 2](#_Toc499424669)

[1.2. Distribution 2](#_Toc499424670)

[2. OVERVIEW 4](#_Toc499424671)

[2.1. Identifier 4](#_Toc499424672)

[2.2. Title 4](#_Toc499424673)

[2.3. Purpose 4](#_Toc499424674)

[2.4. Composition 6](#_Toc499424675)

[3. USING THE PRODUCT 21](#_Toc499424676)

[3.1. Special considerations 21](#_Toc499424677)

[3.2. Instructions for use 21](#_Toc499424678)

# OVERVIEW

## Identifier

INFO2312-Section S11-Team 4

## Title

Hotel Management (Room-booking)

## Purpose

The purpose behind this project goes way beyond history. The Hotel is an integral part of this community and has been around since the cities of Surrey and Langley were small towns, the local communities are attached to this hotel, and it is a tourist attraction and vacation home for many tourists from around the world. With the advent of technology has affected the overall business, as new hotels and more tech-savvy, multiple options are available in the market, the hotel we are designing this database foo is losing its significant share in the Hotel business. The Hotel needs a new Database Management System. As Previous Database system design was not up to date, did not have access to other branches. The Hotel has opened two new branches, so they need an organized structure to operate, modify and provide the database to these branches remotely. The hotel wants to share data within hotel’s database more efficiently and without any hassle.

**Some Functions of the product:**

* Customers will have the option to book the room according to their preference and customer can only exist if they make a payment.
* Both branches will have access to each other’s data so if needed they can make the changes needed.
* This system will remove the cluster of data from some employees and will store them in different entities or relations to store information efficiently, and to ease the looking for data for a special thing.
* Customer will have the option to order food, and electronic gadgets as it is a new service added by the hotel.
* Newly added order will have its own id and will be linked to the product it contains.
* Every branch offers different product at different times, so data can be changed easily as the availability alters.

**Level of Quality required for the product:**

* Consistency: The database should be consistent overall and must have an easy-to-use User Interface.
* Accuracy: The database must be accurate in term of how it is designed and how it operates, so the hotel’s need could be met.
* Relevance: The database must have relevant data because queries depend on the data for them to run. If there is no relevant data, DBMS will fail and could lead to massive losses for the hotel.
* Timeliness: The database should run queries in a timely manner and it should be updated automatically with new data entries in real time.
* Responsive: The database should be responsive and all functions should work.
* Robustness: The database will have the ability to run adequately without any error and it will store all the data accurately.

**The activities required to produce, review and approve the product:**

* Design: This is the first step in production of the product where a team is identified for all tasks and then customer demands and expectations are analysed. The Design phase of the product undergoes heavy workload and changes as it is based on how the end user will use the system and hence must be measurable, detailed and meaningful. There can also be some non-functional requirements such as performance or deadlines to meet which need to be incorporated during the design/production phase. All the requirements, comments and reviews should be documented so that progress can be measured, and product demands can be met.
* Implement: After the product has been designed on paper and has been approved by the client with complete documentation the project enters implementation phase where the real UI is designed, and all changes requested are implemented. For smooth implementation, tools are identified for test management which will help in creating and assigning tasks, tracking the test progress, identifying roadblocks and generating reports indicating progress.
* Testing: After the product is designed and all changes are implemented the product is sent for testing to check for any bugs and errors in the system. This ensures the smooth running of the product. The testing team should be well versed with enhancement of a previously existing system and must be trained not only technically but also with respect to the overall test process being followed.
* Review: When the product is finally ready it is given a final review before sending the final product to client. This review is an in-depth review of all the elements of the product. This is a crucial element as this will govern the stability of the product, which will have a direct impact on the errors solved.

**The following people are required to produce, review and approve the product:**

* Product owner/ Customer: This is the person who reviewed the product he is demanding the service and is the end user of the product.
* Project manager: This person is the main liaison between the customer and the team to deliver the project demands and request of the owner to team and to make sure everything is delivered on time and all demands are met.
* Data Administrator: This person is responsible for processing data and makes sure what data is needed and what needs to be stored in the database.
* Database Designer: This person is responsible for designing the whole database and make sure all the functions outlined are met, error free.
* Database Administrator: This person is responsible for making sure the database is secure and once the database is designed and running database administrator updates any changes required by the customer.

## Composition

The E-R Diagram of the hotel management database depicts information about the hotel with branch, rooms, customer, employees, order, payment, product as entities. With this, customer can book a room or can give an order in order to be considered a customer of the hotel, a person can become a customer by paying first. Employee, Customer, product, branch were given entities because they have their own descriptions. Payments, order and booking were considered entities because they may be needed at any time in future. One employee will be the supervisor for all other Employees, more supervisor could be added in near future if needed.

* Branch form a relationship with the employee through the act of ‘has’ with 1-to-many

multiplicity which means one branch can have one or many employees and but one employee will work in only one branch at a time.

* Also, Branch form a relationship with the product by ‘offers’ with many to many multiplicity which means many products can be offered by many branches and single product could be available in both branches.
* Branch entity is also connected with room entity with the act of ‘have’. It has 1-to-many multiplicity which tells that one branch can have many rooms. It is obvious that one room will be in one branch
* The entity ‘Booking’ forms a relationship named ‘books’ with the entity Room. It has many-to-many relationship which shows that many rooms can be booked with one booking id and one room can have multiple booking sat different time.
* The payment entity is connected to entities booking and order through the act of confirm and order respectively. Both forms one-to-one relationship meaning that one booking and order can have one payment. Payment is needed for order and booking to exist.
* The entity ‘customer’ forms a relationship named ‘makes’ with entity ‘payment’. It has one-to-many relationship which means that one customer can make many payments.
* The entity order is connected with product through the relation contains and it has many-to-many multiplicity which means many an order can contain many product and an product can be ordered at different times from multiple orders.

REFERENCE TO E-R: E-R diagram is in next page, page no: 7

**Relational Schema:**

Employee (empId(pk), fName, lName, contact, email, salary, empPosition, dob, gender, hiringDate, branchId, supId)

Branch (branchId(pk), street, city, postcode, buildingNo)

Offers ((branchId, prodId) (pk))

Rooms (roomNo(pk), roomType, roomPrice, branchId)

Customer (cusId(pk), fName, lName, email)

Product (prodId(pk), price, prodType)

Payment (paymentId(pk), paymentDnT, paymentType, paymentFor, cusId)

Booking (bookingId(pk), startDate, endDate, paymentId)

Books ((bookingId, roomNo) (pk))

Oorder (orderId(pk), paymentId)

contains ((orderId, prodId) (pk))

Employee

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| empId | int | Yes |  |  | NO |  |  |  |  |  |
| fName | varchar |  | NO |  | NO |  |  |  |  |  |
| lName | varchar |  | NO |  | NO |  |  |  |  |  |
| contact | char |  | NO |  | NO |  |  |  |  |  |
| email | varchar |  | NO |  | NO |  |  |  |  |  |
| salary | smallint |  | NO |  | NO |  |  |  |  |  |
| empPosition | varchar |  | NO |  | NO |  |  |  |  |  |
| dob | Date |  | NO |  | NO |  |  |  |  |  |
| gender | Char |  | NO |  | NO |  |  |  |  |  |
| hiringDate | date |  | NO |  | NO |  |  |  |  |  |
| supId | int |  | NO | YES |  | Employee | supId | Cascade | Cascade |  |
| branchid | Int |  | NO | YES |  | Branch | branchId | Cascade | Cascade |  |

Branch

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| branchId | int | Yes |  |  |  |  |  |  |  |  |
| Street | varchar |  |  |  |  |  |  |  |  |  |
| city | varchar |  |  |  |  |  |  |  |  |  |
| postalcode | varchar |  |  |  |  |  |  |  |  |  |
| buildingNo | varchar |  |  |  |  |  |  |  |  |  |

product

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| prodId | int | YES |  |  | NO |  |  |  |  |  |
| Price | Smallint |  | NO |  | NO |  |  |  |  |  |
| prodType | varchar |  | NO |  | NO |  |  |  |  |  |

offers

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| branchId | Int | YES |  | YES |  | Branch | branchId | CASCADE | CASCADE |  |
| prodId | int | YES |  | YES |  | product | prodId | CASCADE | CASCADE |  |

Room

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| roomNo | Int | YES |  |  |  |  |  |  |  |  |
| roomType | VARCHAR |  | NO |  |  |  |  |  |  |  |
| roomPrice | Int |  | NO |  |  |  |  |  |  |  |
| branchId | int |  | NO | YES |  | Branch | branchid | cascade | cascade |  |

Customer

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| cusId | Int | YES |  |  | NO |  |  |  |  |  |
| fName | Varchar |  | NO |  | NO |  |  |  |  |  |
| lName | Varchar |  | NO |  | NO |  |  |  |  |  |
| contact | Char |  | NO |  | NO |  |  |  |  |  |
| email | varchar |  | NO |  | NO |  |  |  |  |  |

payment

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| paymentId | Int | YES |  |  | NO |  |  |  |  |  |
| PaymentDnT | Datetime |  | NO |  | NO |  |  |  |  |  |
| paymentType | Varchar |  | NO |  | NO |  |  |  |  |  |
| paymentFor | Varchar |  | NO |  | NO |  |  |  |  |  |
| cusId | int |  | NO | YES |  | Customer | cusId | cascade | cascade |  |

Booking

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| bookingId | Int | YES |  |  | NO |  |  |  |  |  |
| StartDate | Date |  | NO |  | NO |  |  |  |  |  |
| endDate | Date |  | NO |  | NO |  |  |  |  |  |
| paymentId | int |  | NO | YES | NO | payment | paymentId | cascade | cascade |  |

books

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| bookingId | Int | YES |  | YES |  | Booking | bookingId | Cascade |  |  |
| roomNo | int | YES |  | YES |  | room | roomNo | cascade |  |  |

Oorder

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| orderId | Int | YES |  |  |  |  |  |  |  |  |
| paymentId | int |  | NO | YES |  | payment | paymentId | cascade | cascade |  |

ccontains

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Datatype** | **Primary Key** | | **Foreign Key** | | **Referential Integrity** | | | | **Additional Comments** |
| **Yes** | **No** | **Yes** | **No** | **Parent** | | **On Update** | **On Delete** |
| **Table** | **Attribute** |
| orderId | Int | YES |  | YES |  | oorder | orderId | Cascade |  |  |
| prodId | int | YES |  | YES |  | product | prodId | cascade |  |  |

**Following are the MySQL script:**

* **For table Employee**

CREATE TABLE Employee(

empId int NOT NULL PRIMARY KEY,

fName VARCHAR(20) NOT NULL DEFAULT '\_',

lName VARCHAR(20) NOT NULL DEFAULT '\_',

contact CHAR(10) NOT NULL,

email VARCHAR(20) NOT NULL DEFAULT '\_',

salary smallint NOT NULL,

empPosition VARCHAR(10) NOT NULL,

dob DATE,

gender char(1),

hiringDate date NOT NULL,

branchId int NOT NULL,

FOREIGN KEY(branchId)

REFERENCES Branch(branchId)

ON UPDATE CASCADE ON DELETE CASCADE,

supId int NULL DEFAULT NULL REFERENCES employee(empId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table Branch**

CREATE TABLE Branch(

branchId int NOT NULL PRIMARY KEY,

street VARCHAR(20) NOT NULL DEFAULT '\_',

city VARCHAR(20) NOT NULL DEFAULT '\_',

postcode VARCHAR(20) NOT NULL DEFAULT '\_',

buildingNo VARCHAR(20) NOT NULL DEFAULT '\_'

);

* **For table Product**

CREATE TABLE product(

prodId int NOT NULL PRIMARY KEY,

price smallint NOT NULL,

prodType varchar(20) NOT NULL

);

* **For table offers**

CREATE TABLE offers(

branchId int NOT NULL,

prodId int NOT NULL,

PRIMARY KEY(branchId,prodId),

FOREIGN KEY(branchId)

REFERENCES Branch(branchId)

ON UPDATE CASCADE ON DELETE CASCADE ,

FOREIGN KEY(prodId)

REFERENCES product(prodId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table Room**

CREATE TABLE Room(

roomNo INT NOT NULL PRIMARY KEY,

roomType VARCHAR(20) NOT NULL DEFAULT '\_',

roomPrice INT NOT NULL,

branchId INT NOT NULL,

FOREIGN KEY(branchId)

REFERENCES Branch(branchId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table Customer**

CREATE TABLE Customer(

cusId INT NOT NULL PRIMARY KEY,

fName VARCHAR(20) NOT NULL DEFAULT '\_',

lName VARCHAR(20) NOT NULL DEFAULT '\_',

contact CHAR(10) NOT NULL,

email VARCHAR(50) NOT NULL DEFAULT '\_'

);

* **For table payment**

CREATE TABLE payment (

paymentId INT NOT NULL PRIMARY KEY,

paymentDnt datetime NOT NULL ,

paymentType varchar (10) NOT NULL,

paymentFor varchar (5) NOT NULL,

cusId int NOT NULL,

FOREIGN KEY(cusId)

REFERENCES Customer(cusId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table Booking**

CREATE TABLE Booking (

bookingId INT NOT NULL PRIMARY KEY,

startDate date NOT NULL,

endDate Date NOT NULL,

paymentId INT NOT NULL,

FOREIGN KEY (paymentId)

REFERENCES payment(paymentId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table books**

CREATE TABLE books(

bookingId INT NOT NULL,

roomNo INT NOT NULL,

PRIMARY KEY(bookingId,roomNo),

FOREIGN KEY(roomNo)

REFERENCES Room(roomNo)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(bookingId)

REFERENCES Booking(bookingId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table Oorder**

CREATE TABLE Oorder (

orderId int NOT NULL PRIMARY KEY,

paymentId INT NOT NULL,

FOREIGN KEY (paymentId)

REFERENCES payment(paymentId)

ON UPDATE CASCADE ON DELETE CASCADE

);

* **For table ccontains**

CREATE TABLE ccontains(

orderId int NOT NULL,

prodId int NOT NULL,

PRIMARY KEY(orderId,prodId),

FOREIGN KEY(orderId)

REFERENCES Oorder(orderId)

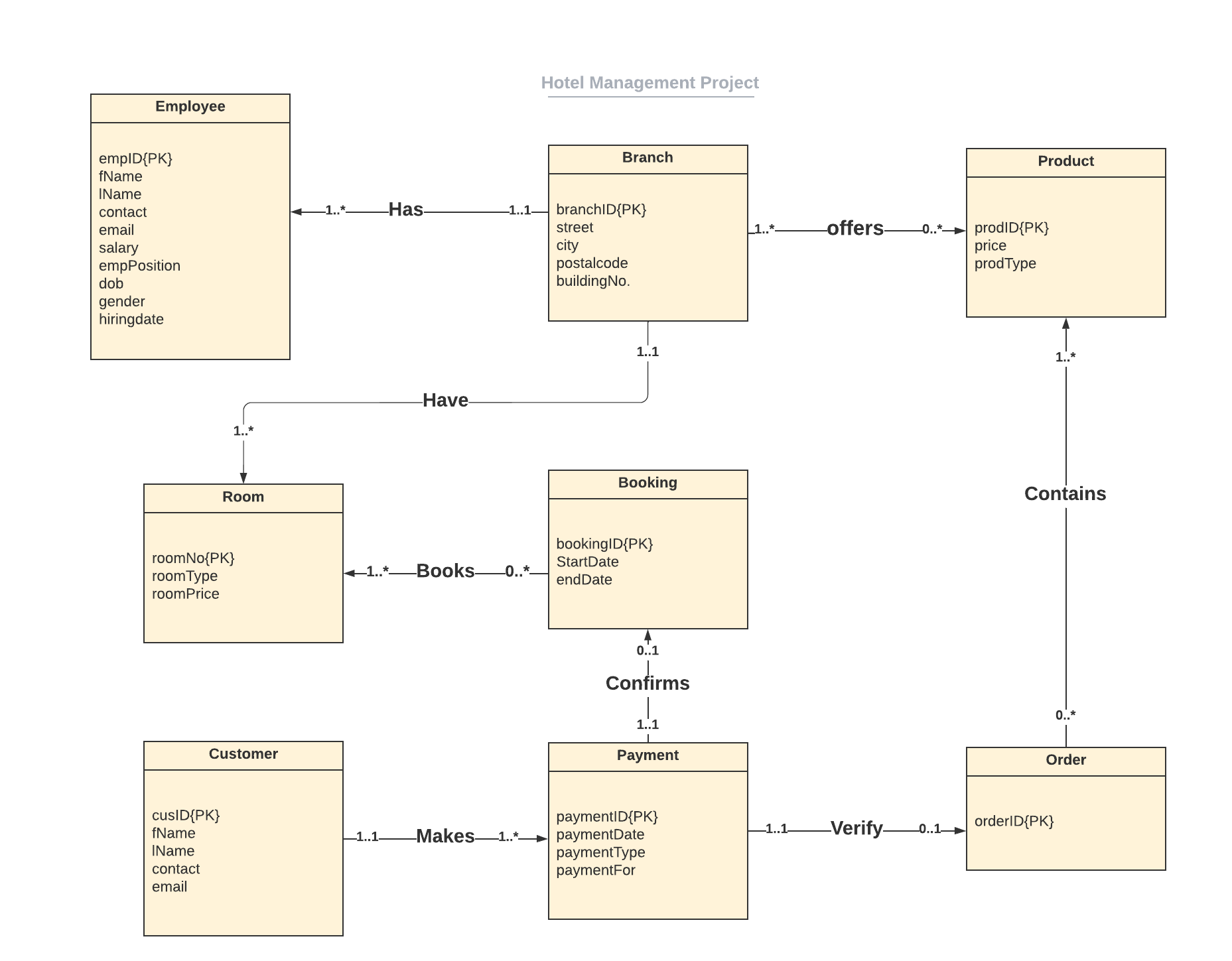
ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(prodId)

REFERENCES Product(prodId)

ON UPDATE CASCADE ON DELETE CASCADE);

E-R Diagram:



COMPLETE SQL SCRIPT CONTAINING TABLES, DATA VALUES AND SOME QUERIES

Drop database 2312FinalProject;

Create database 2312FinalProject;

use 2312FinalProject;

CREATE TABLE Branch(

branchId int NOT NULL PRIMARY KEY,

street VARCHAR(20) NOT NULL DEFAULT '\_',

city VARCHAR(20) NOT NULL DEFAULT '\_',

postcode VARCHAR(20) NOT NULL DEFAULT '\_',

buildingNo VARCHAR(20) NOT NULL DEFAULT '\_'

);

insert into Branch(branchId,street,city,postcode,buildingNo)

values (1,'12 Park pl','Surrey','V3W4S2', '52145'),

(2,'53 188St','Langley','V3S7C3','32145');

CREATE TABLE Employee(

empId int NOT NULL PRIMARY KEY,

fName VARCHAR(20) NOT NULL DEFAULT '\_',

lName VARCHAR(20) NOT NULL DEFAULT '\_',

contact CHAR(10) NOT NULL,

email VARCHAR(20) NOT NULL DEFAULT '\_',

salary smallint NOT NULL,

empPosition VARCHAR(10) NOT NULL,

dob DATE,

gender char(1),

hiringDate date NOT NULL,

branchId int NOT NULL,

FOREIGN KEY(branchId)

REFERENCES Branch(branchId)

ON UPDATE CASCADE ON DELETE CASCADE,

supId int NULL DEFAULT NULL REFERENCES employee(empId)

ON UPDATE CASCADE ON DELETE CASCADE

);

insert into Employee(empID,fName,lName,contact,email,salary,empPosition,dob,gender,hiringDate,branchId,supId)

values (1,'Harman','Kaur','9657473854','harman@gmail.com','2000','Supervisor','2001-12-20','F','2020-10-12',1,NULL),

(2,'Randeep','Singh','8554973598','randeep@gmail.com','1500','Associate','2001-12-22','M','2019-10-12',2,'1'),

(3,'Pardeep','Brar','7845962145','rardeep@gmail.com','1800','Associate','2001-12-23','F','2021-10-12',1,'1'),

(4,'Diljot','Sidhu','7896547895','diljot@gmail.com','900','Associate','2001-12-16','O','2022-10-12',1,'1'),

(5,'Parteek','Mann','1234547996','parteek@gmail.com','1300','Associate','2001-12-15','M','2020-10-12',2,'1');

CREATE TABLE product(

prodId int NOT NULL PRIMARY KEY,

price smallint NOT NULL,

prodType varchar(20) NOT NULL

);

Insert into Product (prodId,price,prodType)

Values(1, 350, 'food'),

(2, 100, 'electronic'),

(3, 320, 'food'),

(4 ,350 ,'food'),

(5, 225, 'electronic'),

(6, 100,'electronic'),

(7, 200, 'food');

CREATE TABLE offers(

branchId int NOT NULL,

prodId int NOT NULL,

PRIMARY KEY(branchId,prodId),

FOREIGN KEY(branchId)

REFERENCES Branch(branchId)

ON UPDATE CASCADE ON DELETE CASCADE ,

FOREIGN KEY(prodId)

REFERENCES product(prodId)

ON UPDATE CASCADE ON DELETE CASCADE

);

insert into offers(branchId,prodId)

values (1,1),

(1,2),

(1,3),

(2,3),

(2,4),

(2,5),

(1,6),

(2,7);

CREATE TABLE Room(

roomNo INT NOT NULL PRIMARY KEY,

roomType VARCHAR(20) NOT NULL DEFAULT '\_',

roomPrice INT NOT NULL,

branchId INT NOT NULL,

FOREIGN KEY(branchId)

REFERENCES Branch(branchId)

ON UPDATE CASCADE ON DELETE CASCADE

);

insert into Room(roomNo,roomType,roomPrice,branchId )

values (1, 'King' ,500,1),

(2, 'queen',600, 1),

(3, 'king' ,500, 1),

(4, 'King' ,700, 2),

(5, 'queen' ,800,2);

CREATE TABLE Customer(

cusId INT NOT NULL PRIMARY KEY,

fName VARCHAR(20) NOT NULL DEFAULT '\_',

lName VARCHAR(20) NOT NULL DEFAULT '\_',

contact CHAR(10) NOT NULL,

email VARCHAR(50) NOT NULL DEFAULT '\_'

);

insert into Customer(cusId, fName, lName, contact, email)

values(1,'Rozy','Gupta' ,'2045555512','sozy.pdt@gmail.com'),

(2,'Prabh' ,'Kaur' ,'2567342002','gill.prabh12@gmail.com'),

(3,'Gurnam' , 'Singh' ,'4432156987','singh.grmn@gmail.com'),

(4, 'David','Singh','4563258965','singh.david@gmail.com');

CREATE TABLE payment (

paymentId INT NOT NULL PRIMARY KEY,

paymentDnt datetime NOT NULL ,

paymentType varchar (10) NOT NULL,

paymentFor varchar (5) NOT NULL,

cusId int NOT NULL,

FOREIGN KEY(cusId)

REFERENCES Customer(cusId)

ON UPDATE CASCADE ON DELETE CASCADE

);

Insert into payment(paymentId, paymentDnt, paymentType,paymentFor, cusID)

Values(1,'2020-10-11 12:34','card','room', 1),

(2,'2020-10-14 10:23','cash','room',1),

(3,'2020-10-15 10:23','card','room',2),

(4,'2020-10-16 10:23','card','room',2),

(5,'2020-12-17 02:45','card','room',3),

(6,'2020-12-11 14:30','card','order',1),

(7,'2020-12-15 12:30','cash','order',2),

(8,'2022-10-03 04:09','card','room',4),

(9,'2022-10-04 16:40','cash','order',4);

CREATE TABLE Booking (

bookingId INT NOT NULL PRIMARY KEY,

startDate date NOT NULL,

endDate Date NOT NULL,

paymentId INT NOT NULL,

FOREIGN KEY (paymentId)

REFERENCES payment(paymentId)

ON UPDATE CASCADE ON DELETE CASCADE

);

Insert into booking(bookingId,startDate,endDate,paymentId)

values (1,'2020-11-12','2020-11-16',1),

(2,'2020-10-16','2020-10-30',2),

(3,'2020-11-17','2020-11-23',3),

(4,'2020-10-17','2020-10-29',4),

(5,'2020-12-25','2020-12-31',5),

(6,'2022-10-03','2022-10-05',8);

CREATE TABLE books(

bookingId INT NOT NULL,

roomNo INT NOT NULL,

PRIMARY KEY(bookingId,roomNo),

FOREIGN KEY(roomNo)

REFERENCES Room(roomNo)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(bookingId)

REFERENCES Booking(bookingId)

ON UPDATE CASCADE ON DELETE CASCADE

);

Insert into books(bookingId, roomNo)

values (1,1),

(2,2),

(3,3),

(4,4),

(5,1),

(5,2),

(6,1),

(6,2),

(6,3);

CREATE TABLE Oorder (

orderId int NOT NULL PRIMARY KEY,

paymentId INT NOT NULL,

FOREIGN KEY (paymentId)

REFERENCES payment(paymentId)

ON UPDATE CASCADE ON DELETE CASCADE

);

Insert into Oorder(orderId,paymentId)

Value (1,6),

(2,7),

(3,9);

CREATE TABLE ccontains(

orderId int NOT NULL,

prodId int NOT NULL,

PRIMARY KEY(orderId,prodId),

FOREIGN KEY(orderId)

REFERENCES Oorder(orderId)

ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY(prodId)

REFERENCES Product(prodId)

ON UPDATE CASCADE ON DELETE CASCADE);

insert into ccontains(orderId ,prodId)

values (1,2),

(1,3),

(2,2),

(2,3),

(3,6),

(3,3),

(3,2);

-- which branches has the most no. of Booking

(SELECT branchId, NoOfBookings

FROM Room, (SELECT roomNo, NoOfBookings

FROM (SELECT roomNo, COUNT(bookingId) AS NoOfBookings

FROM books

GROUP BY roomNo) AS RB

WHERE NoOfBookings IN (SELECT MAX(NoOfBookings)

FROM (SELECT roomNo, COUNT(bookingId) AS NoOfBookings

FROM books

GROUP BY roomNo )AS RB))AS BB

GROUP BY branchId

ORDER BY branchId);

-- Full name of Oldest employee along with No of months they are working for

SELECT empId, CONCAT(fName,' ',lName) AS 'FULL NAME' ,FLOOR(DATEDIFF(CURRENT\_DATE(),hiringDate)/12) AS MONTHS

FROM Employee

WHERE empId = ( SELECT empId

FROM( SELECT empId, CONCAT(fName,' ',lName) AS 'FULL NAME' ,FLOOR(DATEDIFF(CURRENT\_DATE(),hiringDate)/12) AS MONTHS

FROM Employee) AS a

WHERE MONTHS = ( SELECT MAX(MONTHS)

FROM (SELECT FLOOR(DATEDIFF(CURRENT\_DATE(),hiringDate)/12) AS MONTHS

FROM Employee) AS bbbb));

-- Name and email of Customers who oredered the electronic products to aware them of a software issue

SELECT CONCAT(fName,' ', lName) AS NAME, email

FROM Customer

WHERE cusID IN ( SELECT cusId

FROM Payment

WHERE paymentId IN (SELECT paymentId

FROM Oorder

WHERE orderId IN (SELECT orderId

FROM ccontains

WHERE prodId IN ( SELECT prodId

FROM product

WHERE prodType = 'electronic'))));

-- A customer wants to know about the price of booking of 3 king and 2 queen rooms separately from langley and Surrey branch respectively

SELECT Kprice,Qprice

FROM(SELECT SUM(roomPrice)\*3 AS KPrice

FROM Room

WHERE roomType = 'King' AND branchId = (SELECT branchId

FROM Branch

WHERE city = 'LANGLEY')) AS a, (SELECT SUM(roomPrice)\*2 AS QPrice

FROM Room

WHERE roomType = 'Queen' AND branchId = (SELECT branchId

FROM Branch

WHERE city = 'SURREY')) AS b;

-- manager want to add a newly hired employee

INSERT into Employee(empID,fName,lName,contact,email,salary,empPosition,dob,gender,hiringDate,branchId,supId)

values (6, 'Raman', 'Kaur', '2098765467','raman@gmail.com','1900', 'Associate', '2002-12-09','F', '2022-12-10','2', '1');

# USING THE PRODUCT

## Special considerations

The simplicity of the data and shared foreign keys is out of common on its own, we thought it could never work completely accurate but after refilling the whole data and doing some minor changes to the E-R, whole data started to make sense.

## Instructions for use

At first, in this database design the main focus is on the customer booking rooms and giving orders by making payments at different times. Then the data will be shared from payment entity to booking and order and then ahead. So if the database Administrator want to include some new data , he can add data to room,branch,product and employee entities directly. But for booking, order , customer and payments and other many to many relations, data should be added in a sequence. The transfer of primary keys as foreign keys should be kept in mind and it must be accurate.

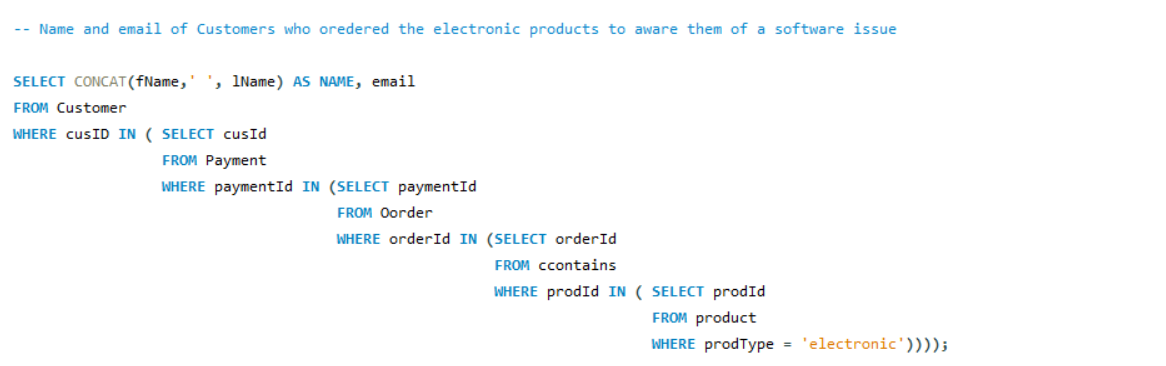
The sequence is written below.

Customer < Payment <booking/order<books/contains<rooms/product.

To insert data following query should be followed for any table, keeping in mind the attribute datatypes and foreign keys.



Other queries for deletion, updation and retrieving data could be modified but our project consists of 5 queries (only 4 were explained in presentation, MAC wasn’t running a query)

 This query is not common as it will be used rarely it tells which customers bought electronic products. It uses 5 different tables to retrieve data.