DB2 – Task

**مدرس**: **د** محمد عبد السلام احمد العشماوي

**معيد**: **د** منه الله ممدوح محمود محمد حسنين

الفرقة الرابعة – نظم معلومات - كلية تجارة وإدارة اعمال – جامعة حلوان – القاهرة

**الموضوع**: قاعدة بيانات نظام إدارة الفندق

**رقم المجموعة**: 11

العام الجامعي: 2021-2022 / 25-12-2021

|  |  |
| --- | --- |
| الاسم | رقم الجلوس |
| احمد شعبان احمد عبد الجابر | 18719 |
| خالد محمود عبد المنعم الجابري | 18742 |
| [زكريا شاهين صالح علواني صالح](https://github.com/zakaria-shahen/) | [18752](mailto:zakariya.shahen@gmail.com) |
| عبد الرحمن صالح احمد امين | 18762 |
| عمر محمد كمال مهدي الشيخ | 18771 |

**\***يعتمد علي MS SQL

**Index**

[Business Rules 2](#_Toc91359889)

[EERD 3](file:///C:\Users\Zakaria%20Shahen\Desktop\Code_now\DB2_project_1\DB2-Hotel\DB2_project1_v2.docx#_Toc91359894)

[Schema 4](file:///C:\Users\Zakaria%20Shahen\Desktop\Code_now\DB2_project_1\DB2-Hotel\DB2_project1_v2.docx#_Toc91359895)

[Query (Select & View) 5](#_Toc91359896)

[Script: Create Database and tables 7](#_Toc91359897)

[Query 7](#_Toc91359898)

[Tools 13](#_Toc91359899)

Qr code

Description automatically generated***\*****يمكنك الانتقال الي العنوان عن طريق Click علي العنوان*

**1 لعرض النسخة الرقمية**

# Business Rules

المكان: فندق

المشكلة: تواجهه إدارة الفندق مشكلة في عملية حوكمة عمليات حجز خدمات وغرف الفندق، وعدم توفر امكانية تحليل البيانات بسبب عدم تخزين البيانات بشكل رقمي ومنظم

لذلك قررت إدارة الفندق رقمنه عمليات حجز الخدمات والغرف

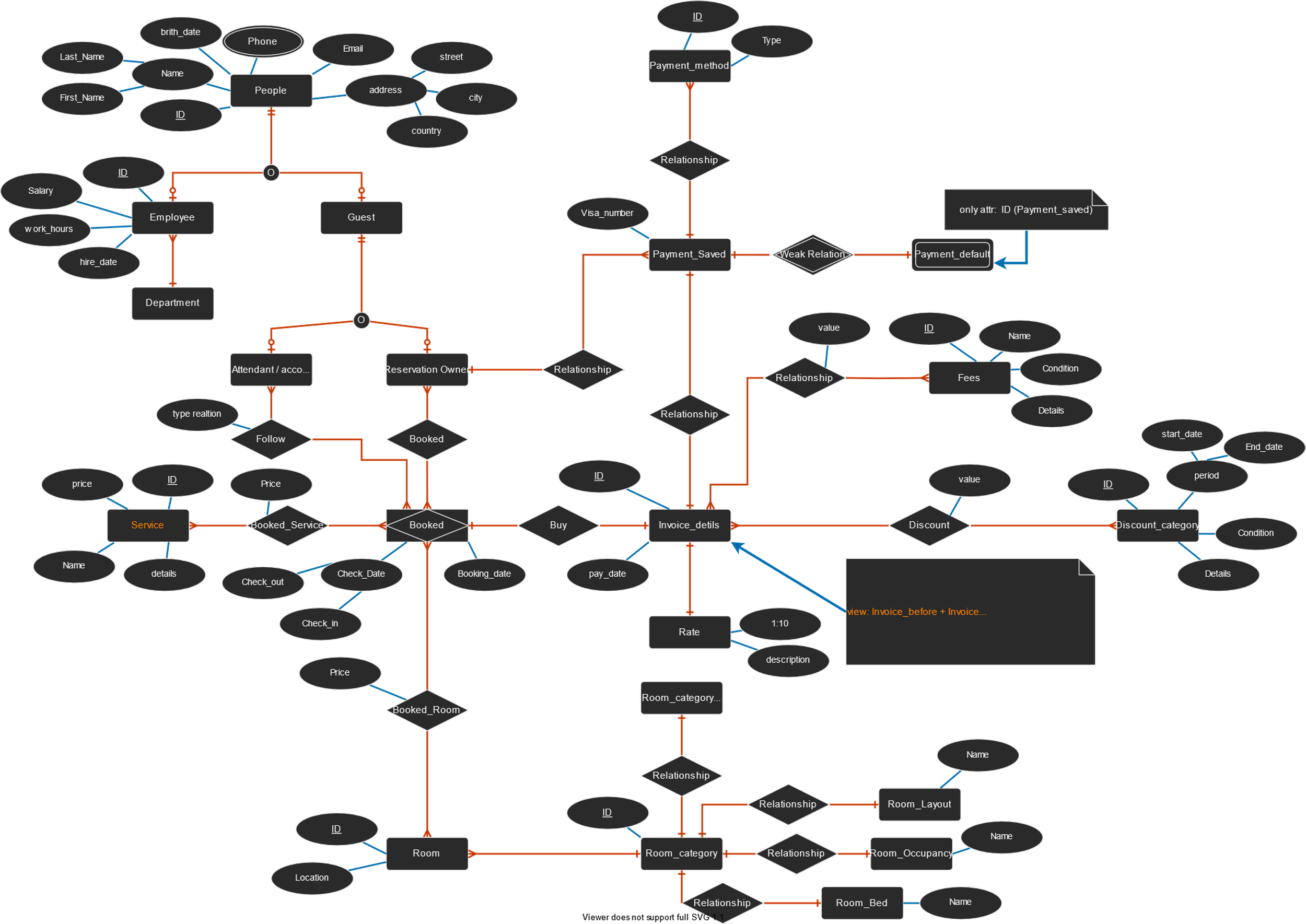
## Tech Stack

MS-SQL, ReacJS, C#(.Not Core)

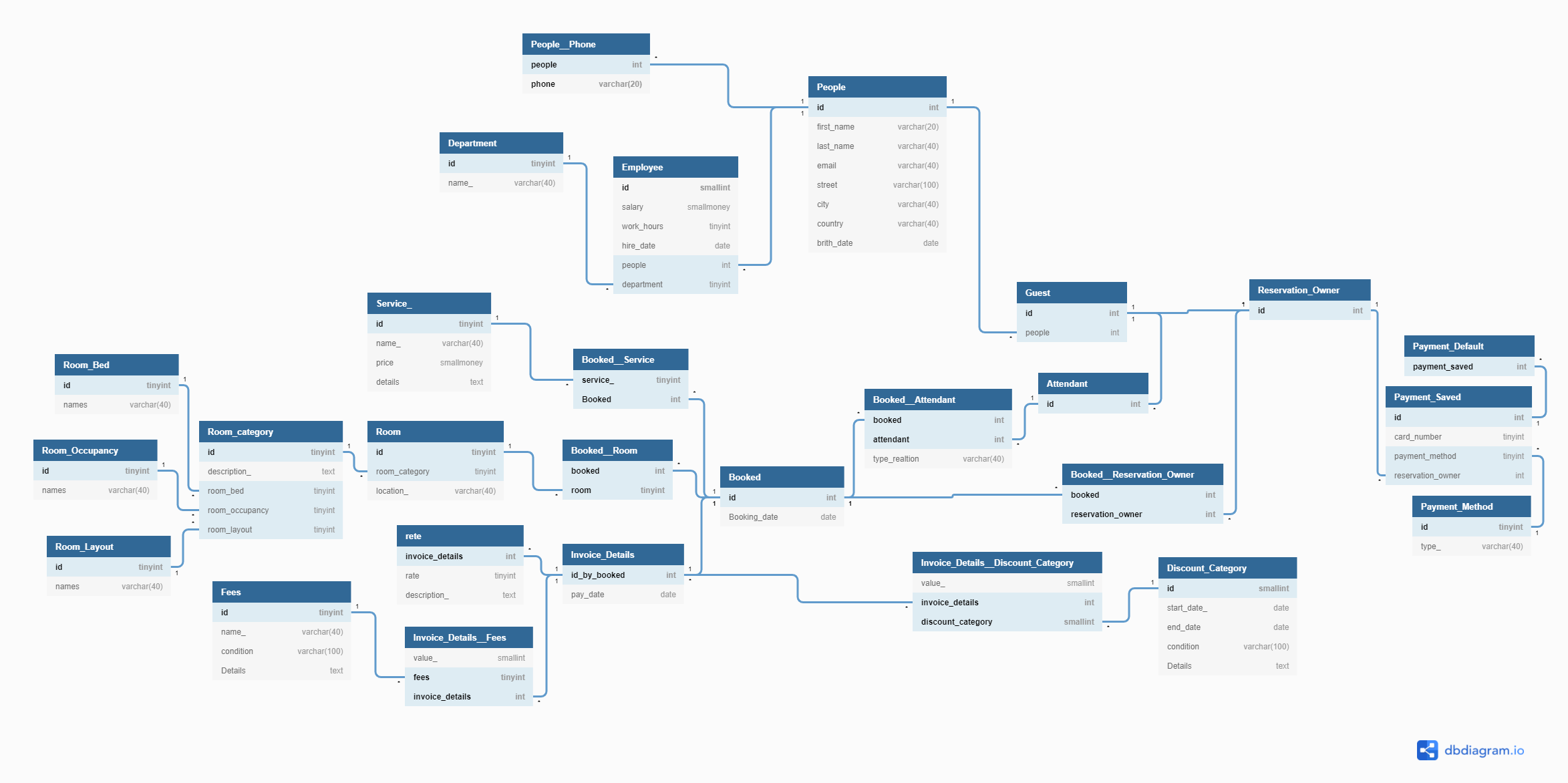
## Database Specifications

* يمكن ان يكون العميل هو صاحب الحجز او يكون العميل هو مرافق لصاحب الحجز من الأسرة او صديق
* يجب تخزين بيانات المرافقين كـعملاء حيث يمكن ان يكون المرافق صاحب حجز في المستقبل... وايضا لتوفير فرصة لتحليل البيانات في المستقبل لتحسين خدمات الفندق
  + إمكانية معرفة أي عميل كان مرافق لمن في أي حجز معين
* يتوفر طرق للدفع، ويمكن للعملاء حفظ وسيلة الدفع المفضلة لهم
  + وأيضا تخزين بيانات جميع وسائل الدفع للعميل (رقم الفيزا)
* يتم الدفع عند الخروج من الفندق
* يمكن ان يحصل العميل علي خصم بموجب كوبون خصم ويمكن ان يتعدد الخصومات
  + حيث يقوم البرنامج بالتحقق من شروط تحقق الخصم ويطبق عليه الخصم
* يتم تطبيق ضرائب ورسوم علي قيمة الفاتورة بعد الخصم
  + مع توفير مرونة للتعامل مع التغيرات الضريبية المستمرة
  + من الممكن تحقق اكثر من قاعدة ضريبية علي نفس الفاتورة
* قد يتم تقديم خدمات اضافيه للعميل (حسب الطلب) وله سعر معين
* منتجات الفندق من غرف والخ.. تنقسم الي 3 متغيرات(مواصفات) رئيسية مع وجود إمكانية للإضافة متغيرات اخري
  + يمكن ان يتكرر نفس المواصفات لأكثر من 100 غرف مع اختلاف مكان الغرفة فقط
* يجب تخزين أسعار الغرف والخدمات بالأسعار القديمة إذا تم بيع غرفة او خدمة بهذا السعر القديم.... لتوفير فرصة لتحليل البيانات في المستقبل
* لكل موظف قسم محدد، وإتاحة إمكانية التوسع ليشمل النظام إدارة الكاملة للموظفين
* توفير إمكانية تقييم الفندق عند الدفع...حيث يكون سؤالين.. سؤال اختر من 1 الي 10.. وسؤال مفتوح لكتابة الاقتراحات

ونعرض هنا تحليل وتصميم وتنفيذ جزء Database



# EERD



# Schema

# Query (Select & View)

يمكنك عرض الكود بشكل أفضل علي GitHub

<https://github.com/zakaria-shahen/DB2_EERD>

* **لمعرفة اجمالي الفاتورة قبل الضرائب والخصم**

-- Create view

create view Invoice\_Before as

select

    iif(sum(Booked\_\_Service.price) is null, sum(Booked\_\_Room.price),

        sum(Booked\_\_Room.price) + sum(Booked\_\_Service.price)) as "Total berfore Discount"

from Invoice\_Details

Left join Booked\_\_Room on Invoice\_Details.id\_by\_booked = Booked\_\_Room.booked

left join Booked\_\_Service on Invoice\_Details.id\_by\_booked = Booked\_\_Service.booked;

-- usgin View

select \* from Invoice\_Before;

* **لمعرفة قيمة الفاتورة مع الضرائب والخصم**

-- create view

create view Invoice as

select

    iif(sum(Booked\_\_Service.price) is null, sum(Booked\_\_Room.price),

        sum(Booked\_\_Room.price) + sum(Booked\_\_Service.price)) as "Price",

    sum(Invoice\_Details\_\_Discount.value\_) as "Discount",

    sum(Invoice\_Details\_\_Fees.value\_) as "Fees"

from Invoice\_Details

Left join Booked\_\_Room on Invoice\_Details.id\_by\_booked = Booked\_\_Room.booked

left join Booked\_\_Service on Invoice\_Details.id\_by\_booked = Booked\_\_Service.booked

left join Invoice\_Details\_\_Discount on Invoice\_Details.id\_by\_booked = Invoice\_Details\_\_Discount.invoice\_details

left join Invoice\_Details\_\_Fees on invoice\_details.id\_by\_booked = Invoice\_Details\_\_Fees.invoice\_details;

-- using view

select \* from Invoice;

* **عرض بيانات كل غرف الفندق**

select

    Room.id as "ID",

    Room.location\_ as "Loaction",

    Room\_Bed.name\_ as "Room\_Bed",

    Room\_Layout.name\_ as "Room\_Layout",

    Room\_Occupancy.name\_ as "Room\_Occupancy",

    Room\_category.description\_ as "description\_"

from Room

left join Room\_category on Room.room\_category = Room\_category.id

left join Room\_Bed on Room\_category.room\_bed = Room\_Bed.id

left join Room\_Layout on Room\_category.room\_bed = Room\_Layout.id

left join Room\_Occupancy on Room\_category.room\_bed = Room\_Occupancy.id

# Script: Create Database and tables

يمكنك عرض الكود بشكل أفضل علي GitHub

<https://github.com/zakaria-shahen/DB2_EERD>

## Query

-- Create a new database called 'Hotel'

use master;

drop database if exists Hotel;

create database Hotel;

use Hotel;

-- Cerate tables

create table People(

    id int identity(1, 1),

    first\_name varchar(20) not null,

    last\_name varchar(40),

    email varchar(40),

    street varchar(100),

    city varchar(40),

    country varchar(40),

    brith\_date date,

    primary key(id)

);

create table People\_\_Phone(

    people int,

    phone varchar(20),

    primary key(people, phone),

    foreign key(people) references People(id)

);

create table Department(

    id tinyint identity(1, 1),

    name\_ varchar(40),

    primary key(id)

);

create table Employee(

    id smallint identity(1, 1),

    salary smallmoney,

    work\_hours tinyint,

    hire\_date date,

    people int not null unique,

    department tinyint,

    primary key(id),

    foreign key(people) references People(id),

    foreign key(department) references Department(id)

);

create index people on Employee(people);

create index department on Employee(department);

create table Guest(

    id int identity(1, 1),

    people int not null unique,

    primary key(id),

    foreign key(people) references People(id)

);

create index people on Guest(people);

create table Reservation\_Owner(

    id int,

    primary key(id),

    foreign key(id) references Guest(id)

);

--  other name table=> accompanying

create table Attendant(

    id int,

    primary key(id),

    foreign key(id) references Guest(id)

);

create table Booked(

    id int identity(1, 1),

    Booking\_date date,

    Check\_in date,

    Check\_out date,

    primary key(id)

);

create table Booked\_\_Reservation\_Owner(

    booked int,

    reservation\_owner int,

    primary key(booked, reservation\_owner),

    foreign key(booked) references Booked(id),

    foreign key(reservation\_owner) references Reservation\_Owner(id)

);

create table Booked\_\_Attendant(

    booked int,

    attendant int,

    type\_realtion varchar(40),

    primary key(booked, attendant),

    foreign key(booked) references Booked(id),

    foreign key(attendant) references Attendant(id)

);

create table Room\_Bed (

    id tinyint identity(1, 1),

    names varchar(40),

    primary key(id)

);

create table Room\_Occupancy(

    id tinyint identity(1, 1),

    names varchar(40),

    primary key(id)

);

create table Room\_Layout(

    id tinyint identity(1, 1),

    names varchar(40),

    primary key(id)

);

create table Room\_category(

    id tinyint identity(1, 1),

    description\_ text,

    room\_bed tinyint,

    room\_occupancy tinyint,

    room\_layout tinyint,

    primary key(id),

    foreign key(room\_bed) references Room\_Bed(id),

    foreign key(room\_occupancy) references Room\_Occupancy(id),

    foreign key(room\_layout) references Room\_Layout(id)

);

create index room\_bed on Room\_category(room\_bed);

create index room\_occupancy on Room\_category(room\_occupancy);

create index room\_layout on Room\_category(room\_layout);

create table Room(

    id tinyint identity(1, 1),

    room\_category tinyint,

    location\_ varchar(40),

    primary key(id),

    foreign key(room\_category) references Room\_category(id)

);

create index room\_category on Room(room\_category)

create table Booked\_\_Room(

    booked int,

    room tinyint,

    primary key(booked, room),

    foreign key(booked) references Booked(id),

    foreign key(room) references Room(id)

);

create index booked on Booked\_\_Room(booked)

create index room on Booked\_\_Room(room)

create table Payment\_Method(

    id tinyint,

    type\_ varchar(40),

    primary key(id)

);

create table Payment\_Saved(

    id int,

    card\_number tinyint,

    payment\_method tinyint,

    reservation\_owner int,

    primary key(id),

    foreign key(payment\_method) references Payment\_Method(id),

    foreign key(reservation\_owner) references Reservation\_Owner(id)

);

create index payment\_method on Payment\_Saved(payment\_method)

create index reservation\_owner on Payment\_Saved(reservation\_owner)

create table Payment\_Default(

    payment\_saved int,

    primary key(payment\_saved),

    foreign key(payment\_saved) references Payment\_Saved(id)

);

create table Invoice\_Details(

    id\_by\_booked int,

    pay\_date date,

    primary key(id\_by\_booked),

    foreign key(id\_by\_booked) references Booked(id)

);

create table Discount\_Category(

    id smallint,

    start\_date\_ date,

    end\_date date,

    condition varchar(100), -- or JSON

    Details text,

    primary key(id)

);

create table Invoice\_Details\_\_Discount\_Category(

    value\_ smallint,

    invoice\_details int,

    discount\_category smallint,

    primary key(invoice\_details, discount\_category),

    foreign key(invoice\_details) references Invoice\_Details(id\_by\_booked),

    foreign key(discount\_category) references Discount\_Category(id)

);

create table Fees(

    id tinyint,

    name\_ varchar(40),

    condition varchar(100),

    Details text,

    primary key(id)

);

create table Invoice\_Details\_\_Fees(

    value\_ smallint,

    fees tinyint,

    invoice\_details int,

    primary key(fees, invoice\_details),

    foreign key(fees) references Fees(id),

    foreign key(invoice\_details) references Invoice\_Details(id\_by\_booked)

);

create table rete(

    invoice\_details int,

    rate tinyint check(rate >= 1 and  rate <= 10),

    description\_ text,

    primary key(invoice\_details),

    foreign key(invoice\_details) references Invoice\_Details(id\_by\_booked)

);

create table Service\_(

    id tinyint identity(1, 1),

    name\_ varchar(40),

    price smallmoney,

    details text,

    primary key(id)

);

create table Booked\_\_Service(

    service\_ tinyint,

    Booked int,

    primary key(service\_, Booked),

    foreign key(service\_) references Service\_(id),

    foreign key(booked) references Booked(id)

);

## Tools

* VSCode
* Azure Data Studio
* draw.io
* <https://dbdiagram.io>