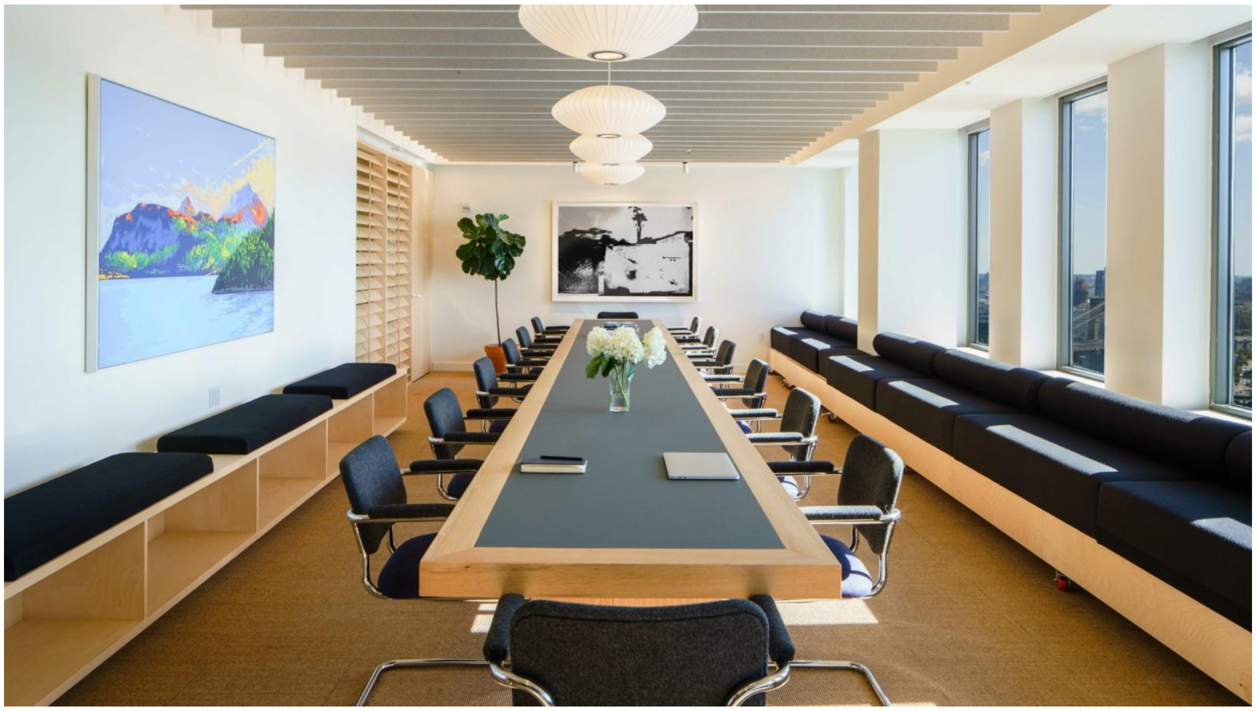
MEETING ROOM BOOKING SYSTEM



IST 659: DATABASE ADMINISTRATION CONCEPTS AND DATABASE MANAGEMENT SYSTEM

HEMANG GALA - 476855319
SUYASH GUPTE - 517756672
YASH SHIMPI - 3156038308

TABLE OF CONTENTS

1. INTRODUCTION

This project deals with the design and implementation of a database management system for handling the booking system of meeting rooms for the XYZ company. During the global COVID-19 pandemic, almost every company and organization transitioned to working remotely. However, there arise certain times when the employees have to come to the workspace in person for obtaining and working on data which is sensitive to be shared with employees or attend meetings about specific projects or operations which are core to the company and may be too confidential to be held on cloud-based meeting rooms like Zoom or Microsoft Teams. For such purposes we have designed a meeting room booking app, based on MSSQL database system in order to ease the process of booking rooms in advance for meetings or working, and relieve the administrator of the tedious task of manually assigning the rooms to the requesting employees.

In brief, this database includes the information of employees like their names, email, number and department; the departments in the office; meeting room information like the equipment available and capacity; and data about the buildings where the offices of the company are situated.

We designed the system in SQL in order to simplify the application. The system contains various abovementioned tables of information and queries which can be utilized by the manager to make the task simpler and faster. The user interface for the interaction with the database was made using Microsoft PowerApps. Microsoft PowerApps is very useful for developing applications efficiently, quickly and effortlessly as it provides all the features and APIs required to connect to a database in a single platform.

2. PROBLEM STATEMENT

Before the global COVID-19 pandemic, there was less frequency of reservation of meeting rooms, as the employees would come to the workspace on a daily basis, and the process of reservation could be done manually. However, now the situation has come up wherein the need to reserve rooms has increased, and the task of allotting rooms has become too tedious for the administrator to be handled manually. The need to provide rooms with necessary equipment to the employees has also increased the workload on the admin as they have to cross check each and every room before assigning to the requesting person. Moreover, the employees who are looking out for rooms have to spend a lot of precious time trying to find out the rooms' availability and their features before they can decide on one; this time could very easily be spent working on the company's projects and increasing productivity.

3. SUGGESTED SOLUTION

The solution that we have come up with to solve the above stated problem is a database management system for the company which was designed and implemented using MSSQL and Microsoft PowerApps. This database consists of all the salient information about the employees and the features and availability of the meeting rooms in the different offices of the company.

The database management system allows the administrator to store all the information about the employees and the meeting room in one database, making the allotment process much easier. Also, the employees save up a lot of time as they can just access the application to check the features and availability of the meeting rooms, and book one which satisfies their needs.

4. TOPIC SELECTION

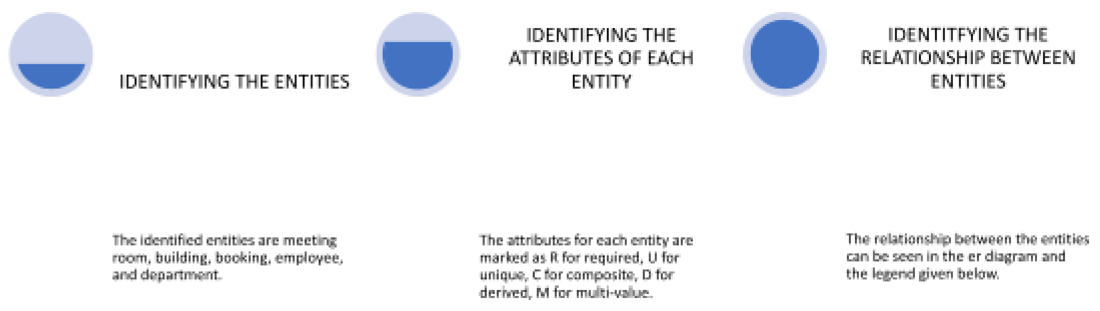
We selected the topic of Meeting Room Booking System as there has been a massive overhaul in the operations and functioning of many companies due to the global COVID-19 pandemic. Employees have started working remotely and almost the entire work has been shifted over to cloud based systems. However, at times the employees need to report to the workspace in order to work on certain parts of projects or assignments, as they cannot be undertaken on the cloud due to confidentiality or technical restrictions. So, we decided to design and implement a meeting room booking system for the company in order to reduce the wastage of time and energy of the employees to lookup for one and increase their efficiency towards the company assignments instead. Furthermore, the project was also undertaken with the intention of streamlining the process of managing the process of booking for the administrator by providing them with a single, unified platform to hold, maintain and manage the data of the employees and the meeting rooms across the different offices of the company in New York.

5. IDENTIFICATION OF DATA LOGIC

Throughout the project we have implemented data logic like Transactions and Triggers. Transactions are used to insert data into the tables. They are particularly helpful when inserting large amounts of data in one go. If there is any error in inserting any entry, the database rolls back to its original state without adding any entries. Triggers are used in making the meeting room status to not available in case an employee books a room. Apart from this we have used joins to display multiple tables. The code for this logic can be found at the Script of Database section.

6. CONCEPTUAL DATA MODEL

For the conceptual data model, we have designed an entity relationship diagram. The ER diagram below is used to show the relationship of different entities with one another. The symbols used in the diagram can be referred to the legend given below.

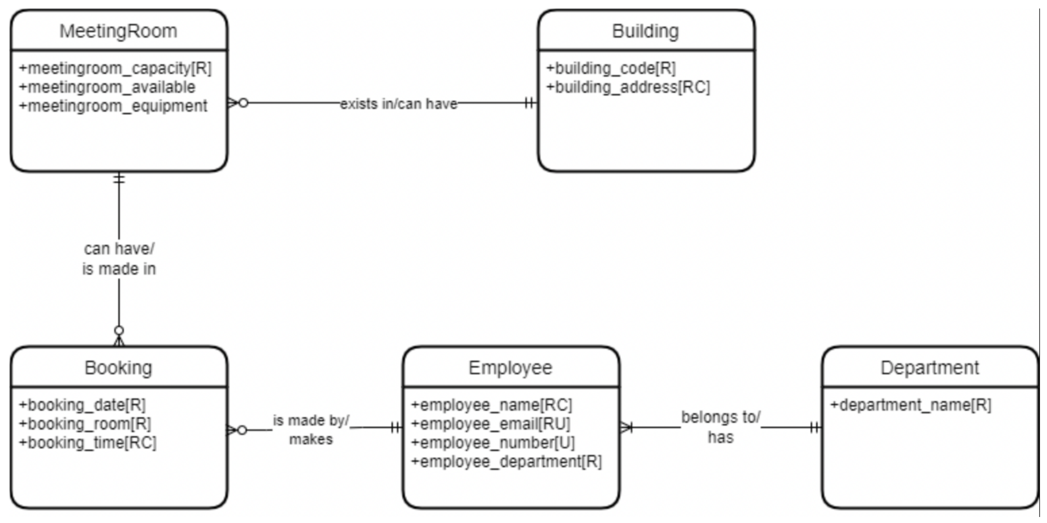

IDENTIFYING THE ENTITIES
IDENTIFYING THE ATTRIBUTES OF EACH ENTITY
IDENTITFYING THE RELATIONSHIP BETWEEN ENTITIES

The identified entities are meeting room, building, booking, employee, and department.

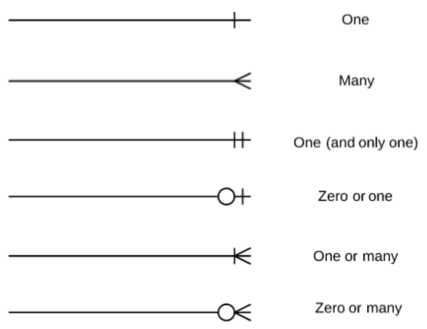
The attributes for each entity are marked as R for required, U for unique, C for composite, D for derived, M for multi-value.

The relationship between the entities can be seen in the er diagram and the legend given below.

STEPS FOR ER MODELING



ENTITY RELATIONSHIP DIAGRAM

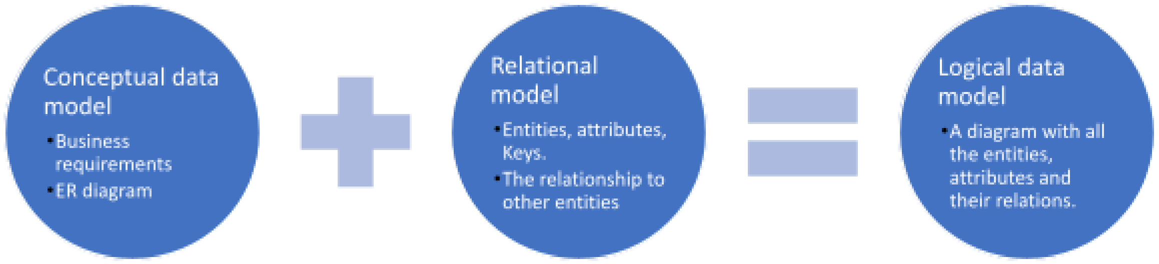


LEGEND

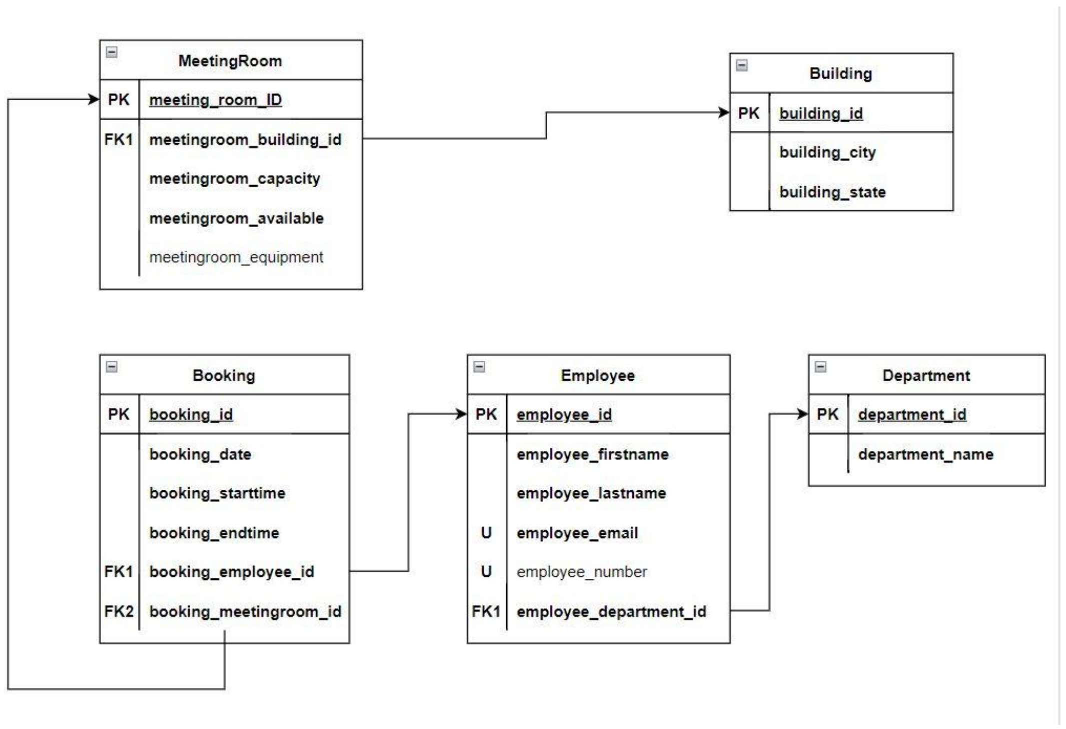
The meeting room has the attributes capacity, availability and equipment. One meeting room can exist in one and only one building and a building can have zero or many meeting rooms. Building has the attributes code and address. The entity booking has the attributes date, room and time. A meeting room can have zero or many bookings, but a booking is made in one and only one meeting room. The entity employee has the attributes name, email, number and department. An employee can make zero or many bookings, but a booking is made by one and only one employee. Department entity has two attributes; name and code. A department can have one or many employees, but an employee can belong to one and only one department.

7. LOGICAL DATA MODEL

A logical data model is essential in the sense that the final implementation depends on it, though it may not be exactly similar to the logical data model. It is the combination of conceptual data model and the implementation model. It involves the mapping of the conceptual model to the implementation model. It is useful as it provides documentation of the database entities and attributes and helps the engineer to construct a database.


Conceptual data model
Business requirements
ER diagram
Relational model
Entities, attributes, Keys.
The relationship to other entities
Logical data model
A diagram with all the entities, attributes and their relations.

LOGICAL DATA MODEL STEPS WITH RELATION MODEL



LOGICAL DATA MODEL

i) **MeetingRoom**: PK is meeting_room_id and the FK is meetingroom_building_id and the foreign key points to the primary key of the building entity.

ii) **Building**: PK is building_id and there are no foreign keys in this table.

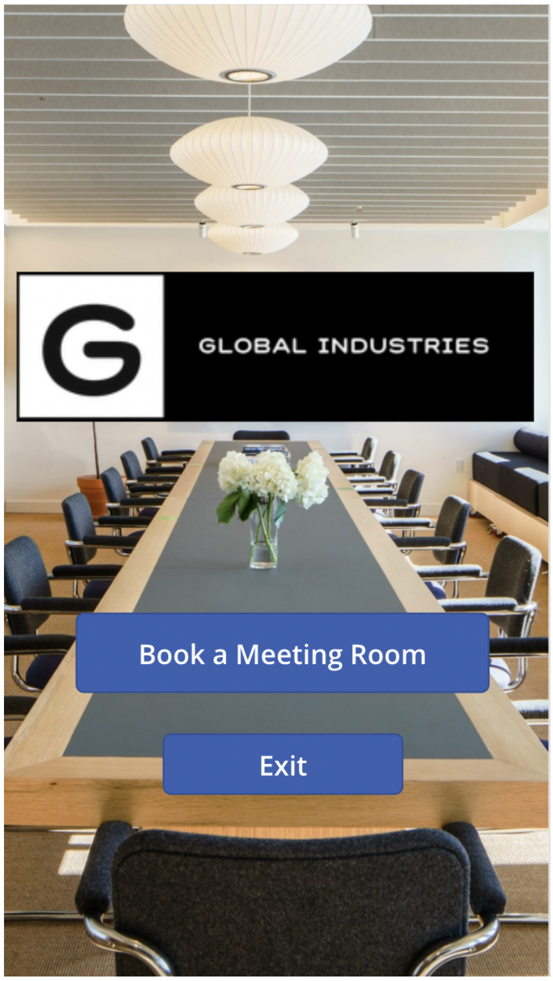
iii) **Booking**: PK is booking_id and the FKs in this table are booking_employee_id and booking_meetingroom_id. The booking_employee_id FK points to the primary key of Employee and the booking_meetingroom_id points to the primary key of MeetingRoom.

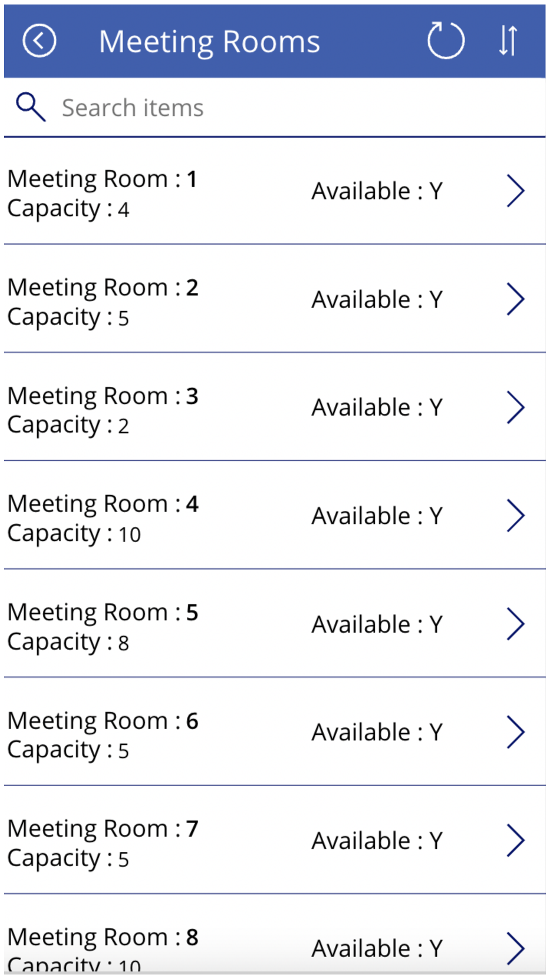
iv) **Employee**: PK is employee_id and the FK is employee_department_id. The FK points to the primary key of the table Department. There are two attributes with unique properties which are employee_email and employee_number.

v) **Department**: PK is department_id and there are no foreign keys in this table.

8. APPLICATION SCREENS

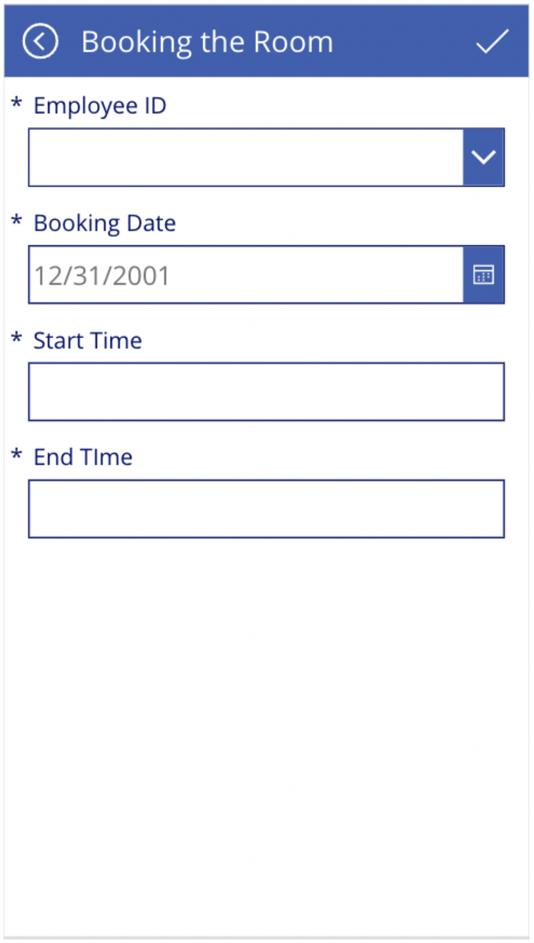
The basic user interface for the users to interact with the database has been designed and implemented using Microsoft PowerApps. This allows the administrator to check the list of employees and allows the employees to alter the database by booking rooms straight from the application.





Meeting Room : **1** Capacity : 4	Available : Y
Meeting Room : **2** Capacity : 5	Available : Y
Meeting Room : **3** Capacity : 2	Available : Y
Meeting Room : **4** Capacity : 10	Available : Y
Meeting Room : **5** Capacity : 8	Available : Y
Meeting Room : **6** Capacity : 5	Available : Y
Meeting Room : **7** Capacity : 5	Available : Y
Meeting Room : **8** Capacity : 10	Available : Y

Screen 1 Screen 2


Booking the Room

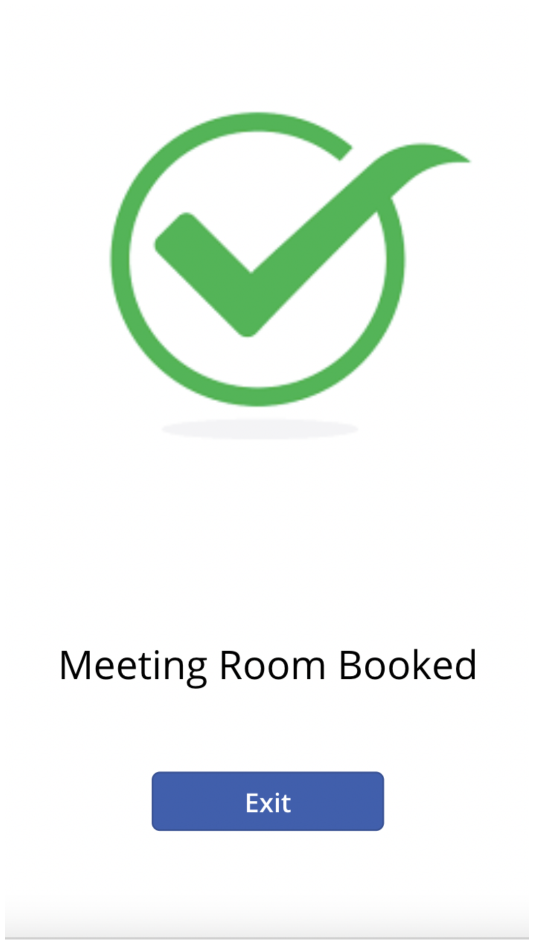
* Employee ID

* Booking Date

12/31/2001

* Start Time

* End TIme


Meeting Room Booked

Screen 3 Screen 4

Screen 1 shows two options one to book a meeting room and other to Exit the application. Once employees select the ‘Book a Meeting Room’ option, the app takes them to **Screen 2** which gives them the list of meeting rooms. This list is linked to the Company’s meeting room database and shows 3 things: Meeting Room number, Meeting Room capacity and if it is available. If the meeting room gets booked it gets dismissed from this list. Once an employee selects which meeting room to book according to the capacity, they will go to **Screen 3** where they must enter some basic details about themselves. First, they will get to select their employee id from the drop down, then the meeting date and lastly start and end time of their meeting. Once they fill this detail, they can press on the tick mark on the top right corner and the room will be booked successfully. **Screen 4** will show that a meeting room is booked successfully. Employees can now close the application, or they can press exit which will navigate them back to the first page where they can book another meeting room.

9. SCRIPT FOR THE DATABASE

CREATE THE DATABASE:

```
if not exists(select * from sys.databases where name='office')
create database office
GO

use office
Go
```

UP/DOWN SCRIPT FOR THE CREATION OF TABLES:

```
-- DOWN
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where Constraint_Name = 'fk_bookings_booking_employee_id')
    alter table bookings drop constraint fk_bookings_booking_employee_id
if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where Constraint_Name = 'fk_bookings_booking_meetingroom_id')
    alter table bookings drop constraint fk_bookings_booking_meetingroom_id
drop table if exists bookings

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where Constraint_Name = 'fk_meetingrooms_meetingroom_building_code')
    alter table meetingrooms drop constraint fk_meetingrooms_meetingroom_building_code
drop table if exists meetingrooms

drop table if exists buildings

if exists(select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    where Constraint_Name = 'fk_employees_employee_department_id')
    alter table employees drop constraint fk_employees_employee_department_id
drop table if exists employees

drop table if exists departments


-- UP

create table departments (
    department_id int identity not null,
    department_name varchar(50) not null,
    constraint pk_departments_department_id primary key (department_id),
)


create table employees (
    employee_id int identity not null,
    employee_firstname varchar(50) not null,
    employee_lastname varchar(50) not null,
    employee_email varchar(50) not null,
    employee_number numeric(10) null,
    employee_department_id int not null,
    constraint pk_employees_employee_id primary key (employee_id),
    constraint u_employees_employee_email unique (employee_email),
    constraint u_employees_employee_number unique (employee_number),
)
GO

alter table employees
    add constraint fk_employees_employee_department_id foreign key (employee_department_id)
    references departments(department_id)
GO

create table buildings (
    building_id int identity not null,
    building_code varchar(50) not null,
    building_city varchar(50) not null,
    building_state varchar(50) not null
    constraint pk_buildings_building_id primary key (building_id),
    constraint u_buildings_building_code unique (building_code)
)
GO

create table meetingrooms (
    meetingroom_id int identity not null,
    meetingroom_capacity int not null,
    meetingroom_available char(1) not null,
    meetingroom_equipment varchar(50) null,
    meetingroom_building_code varchar(50) not null,
    constraint pk_meetingrooms_meetingroom_id primary key (meetingroom_id),
)
GO

alter table meetingrooms
    add constraint fk_meetingrooms_meetingroom_building_code foreign key (meetingroom_building_code)
    references buildings(building_code)
GO

create table bookings (
    booking_id int identity not null,
    booking_date date not null,
    booking_starttime time not null,
    booking_endtime time not null,
    booking_employee_id int not null,
    booking_meetingroom_id int not null,
    constraint pk_bookings_booking_id primary key (booking_id),
    constraint ck_valid_startend_time check (booking_starttime<=booking_endtime)
)
GO

alter table bookings
    add constraint fk_bookings_booking_employee_id foreign key (booking_employee_id)
    references employees(employee_id)
GO

alter table bookings
    add constraint fk_bookings_booking_meetingroom_id foreign key (booking_meetingroom_id)
    references meetingrooms(meetingroom_id)
GO
```

INSERT DATA INTO THE COLUMNS:

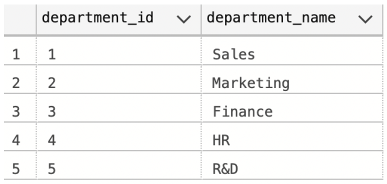
```
--UP DATA
insert into departments (department_name) VALUES
   ('Sales'),('Marketing'),('Finance'),('HR'),('R&D')

BEGIN
BEGIN TRY
   BEGIN TRANSACTION

      Insert into employees values ('Lanette','Presman','lpresman0@baidu.com',9733888212,2)
      Insert into employees values ('Dolph','Frere','dfrere1@sourceforge.net',3852140023,5)
      Insert into employees values ('Conrade','Bachelor','cbachelor2@cpanel.net',7624389531,3)
      Insert into employees values ('Hildy','Louca','hlouca3@weebly.com',6287691532,5)
      Insert into employees values ('Forester','Bolduc','fbolduc4@geocities.jp',9502800927,2)
      Insert into employees values ('Virgilio','Sneesbie','vsneesbie5@tuttocitta.it',1138203357,2)
      Insert into employees values ('Marylou','McGurn','mmcgurn6@photobucket.com',4112016426,5)
      Insert into employees values ('Karisa','De Francesco','kdefrancesco7@hud.gov',8211034045,2)
      Insert into employees values ('Jany','Jacson','jjacson8@1und1.de',6265285679,2)
      Insert into employees values ('Hi','Wethered','hwethered9@comsenz.com',1414550360,2)
      Insert into employees values ('Armin','MacKill','amackilla@jigsy.com',4349564534,4)
      Insert into employees values ('Clarance','Ollerhead','collerheadb@seesaa.net',2028597101,5)
      Insert into employees values ('Victor','Onion','vonionc@blogger.com',3483859126,5)
      Insert into employees values ('Heidi','Maleby','hmalebyd@sun.com',5298423653,2)
      Insert into employees values ('Estrellita','Goldthorpe','egoldthorpee@g.co',1619318766,1)
      Insert into employees values ('Giffer','Powdrell','gpowdrellf@weebly.com',7929827614,1)
      Insert into employees values ('Sigismond','Tagg','staggg@squidoo.com',6844760467,5)
      Insert into employees values ('Daisey','Worsham','dworshamh@bizjournals.com',2835853909,4)
      Insert into employees values ('Bank','Simkins','bsimkinsi@slashdot.org',7016610627,2)
      Insert into employees values ('Rafael','Manueau','rmanueauj@aol.com',9455730941,4)
      Insert into employees values ('Sari','Chidwick','schidwickk@feedburner.com',3943757453,4)
      Insert into employees values ('Jilli','Danell','jdanelll@privacy.gov.au',6853400028,5)
      Insert into employees values ('Margit','Pilling','mpillingm@1und1.de',7197552066,1)
      Insert into employees values ('Angy','Howship','ahowshipn@washington.edu',4959334431,5)
      Insert into employees values ('Alex','Widdecombe','awiddecombeo@cisco.com',9972291154,1)
      Insert into employees values ('Rahal','Kobu','rkobup@icio.us',4903028132,4)
      Insert into employees values ('Hendrika','Lumsdale','hlumsdaleq@soup.io',3221630930,5)
      Insert into employees values ('Corinne','Pettie','cpettier@ox.ac.uk',3087974236,2)
      Insert into employees values ('Janaye','Drewet','jdrewets@surveymonkey.com',4667964020,5)
      Insert into employees values ('Theadora','Brettor','tbrettort@unblog.fr',6815000606,5)
      Insert into employees values ('Alfred','Hance','ahanceu@tamu.edu',1003705298,4)
      Insert into employees values ('Dyana','Stenners','dstennersv@tinypic.com',2762243563,3)
      Insert into employees values ('Evelin','Bennough','ebennoughw@weather.com',7912330681,1)
      Insert into employees values ('Gayelord','Simak','gsimakx@eepurl.com',4271526241,2)
      Insert into employees values ('Tarrah','Ballister','tballistery@noaa.gov',6161303046,3)
      Insert into employees values ('Creigh','Tatham','ctathamz@miitbeian.gov.cn',4523529607,2)
      Insert into employees values ('Westbrooke','Schult','wschult10@statcounter.com',7436436265,2)
      Insert into employees values ('Jodee','Snedden','jsnedden11@chronoengine.com',9714596799,3)
      Insert into employees values ('Marney','Edwardes','medwardes12@printfriendly.com',1847678778,4)
      Insert into employees values ('Hetty','Tabourier','htabourier13@weebly.com',5053058315,1)
      Insert into employees values ('Carl','Barttrum','cbarttrum14@bravesites.com',3782964287,5)
      Insert into employees values ('Vail','Evelyn','vevelyn15@forbes.com',8052286678,5)
      Insert into employees values ('Rolland','Rickertsen','rrickertsen16@time.com',2934091239,5)
      Insert into employees values ('Obed','Fletham','ofletham17@behance.net',9913541089,5)
      Insert into employees values ('Sigvard','Scotting','sscotting18@naver.com',3966419689,4)
      Insert into employees values ('Aharon','Dellatorre','adellatorre19@webnode.com',8469998692,5)
      Insert into employees values ('Valentine','Ghidoni','vghidoni1a@feedburner.com',1332411327,1)
      Insert into employees values ('Leslie','Watson-Brown','lwatsonbrown1b@mtv.com',2456908324,5)
      Insert into employees values ('Damaris','Ritchman','dritchman1c@usa.gov',1373959809,5)
      Insert into employees values ('Dalila','Franklen','dfranklen1d@bizjournals.com',1241204345,4)

   COMMIT
END TRY
```

```sql
BEGIN CATCH
   ROLLBACK ;
   THROW
END CATCH
END

BEGIN
BEGIN TRY
   BEGIN TRANSACTION

insert into buildings(building_code,building_city,building_state) VALUES
   ('A1','New York','NY'),
   ('A2','New York','NY'),
   ('A3','New York','NY'),
   ('B1','New York','NY'),
   ('B2','New York','NY'),
   ('B3','New York','NY'),
   ('B4','New York','NY')

   COMMIT
END TRY
BEGIN CATCH
   ROLLBACK ;
   THROW
END CATCH
END


BEGIN
BEGIN TRY
   BEGIN TRANSACTION

insert into
meetingrooms(meetingroom_capacity,meetingroom_available,meetingroom_equipment,meetingroom_building_code)
VALUES
   (4,'Y','Projector','A1'),
   (5,'Y','Projector, Conference Table','A1'),
   (2,'Y','Projector','A2'),
   (10,'Y','Projector, Conference Table','A3'),
   (8,'Y','Projector, Conference Table','A3'),
   (5,'Y','Projector, Conference Table','B1'),
   (5,'Y','Projector, Conference Table','B2'),
   (10,'Y','Projector, Conference Table','B3'),
   (10,'Y','Projector, Conference Table','B3'),
   (10,'Y','Projector, Conference Table','B3'),
   (2,'Y','Projector','A2'),
   (8,'Y','Projector, Conference Table','B4'),
   (8,'Y','Projector','B4'),
   (4,'Y','Projector','B1'),
   (4,'Y','Projector, Conference Table','A1')

   COMMIT
END TRY
BEGIN CATCH
   ROLLBACK ;
   THROW
END CATCH
END

GO
```

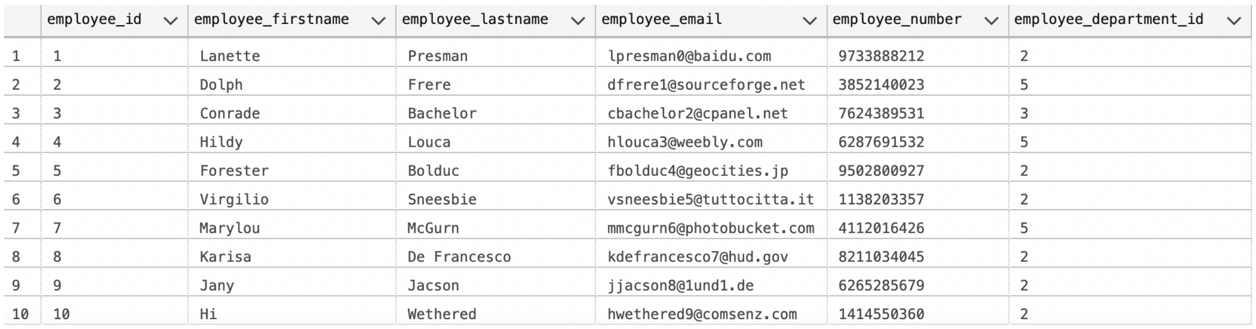
VERIFICATION OF THE TABLES:

```
-- VERIFY
select * from departments
select * from employees
select * from meetingrooms
select * from buildings
GO
```

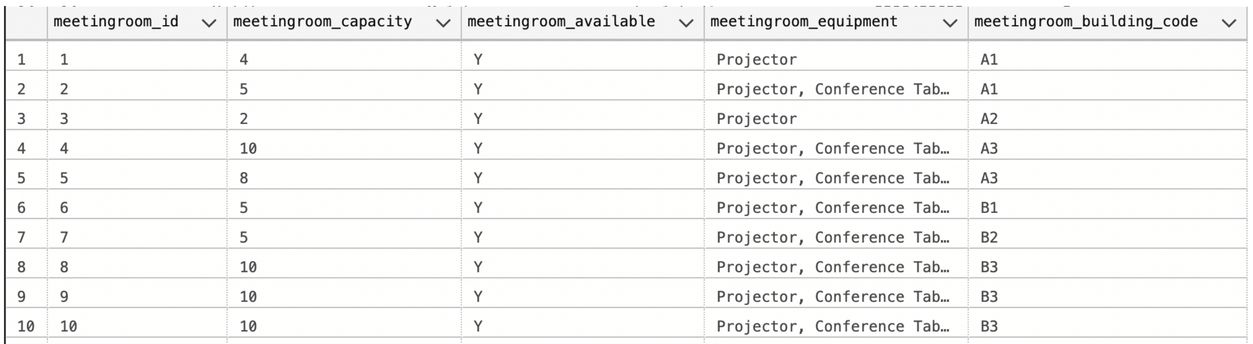
SCREENSHOT OF TABLES:



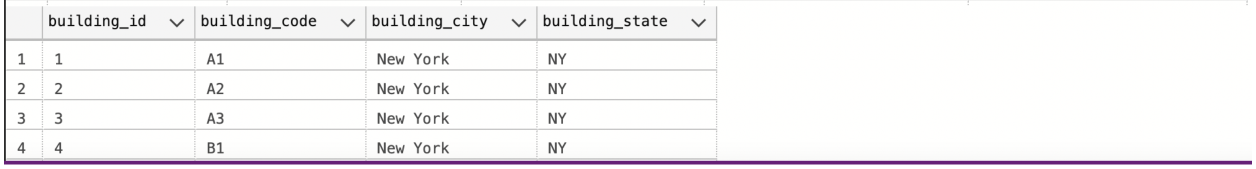
	department_id	department_name
1	1	Sales
2	2	Marketing
3	3	Finance
4	4	HR
5	5	R&D



	employee_id	employee_firstname	employee_lastname	employee_email	employee_number	employee_department_id
1	1	Lanette	Presman	lpresman0@baidu.com	9733888212	2
2	2	Dolph	Frere	dfrere1@sourceforge.net	3852140023	5
3	3	Conrade	Bachelor	cbachelor2@cpanel.net	7624389531	3
4	4	Hildy	Louca	hlouca3@weebly.com	6287691532	5
5	5	Forester	Bolduc	fbolduc4@geocities.jp	9502800927	2
6	6	Virgilio	Sneesbie	vsneesbie5@tuttocitta.it	1138203357	2
7	7	Marylou	McGurn	mmcgurn6@photobucket.com	4112016426	5
8	8	Karisa	De Francesco	kdefrancesco7@hud.gov	8211034045	2
9	9	Jany	Jacson	jjacson8@1und1.de	6265285679	2
10	10	Hi	Wethered	hwethered9@comsenz.com	1414550360	2



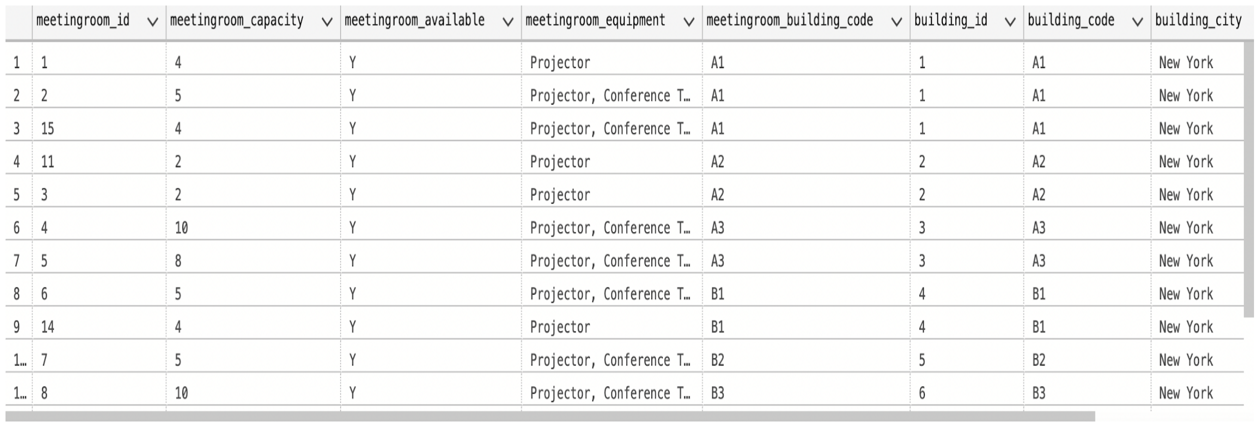
	meetingroom_id	meetingroom_capacity	meetingroom_available	meetingroom_equipment	meetingroom_building_code
1	1	4	Y	Projector	A1
2	2	5	Y	Projector, Conference Tab…	A1
3	3	2	Y	Projector	A2
4	4	10	Y	Projector, Conference Tab…	A3
5	5	8	Y	Projector, Conference Tab…	A3
6	6	5	Y	Projector, Conference Tab…	B1
7	7	5	Y	Projector, Conference Tab…	B2
8	8	10	Y	Projector, Conference Tab…	B3
9	9	10	Y	Projector, Conference Tab…	B3
10	10	10	Y	Projector, Conference Tab…	B3



	building_id	building_code	building_city	building_state
1	1	A1	New York	NY
2	2	A2	New York	NY
3	3	A3	New York	NY
4	4	B1	New York	NY

RUNNING QUERY ON THE DATABASE:

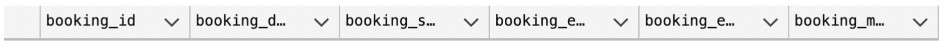
```
select *
    from meetingrooms m
    join buildings b
    on m.meetingroom_building_code = b.building_code
    order by b.building_code
```



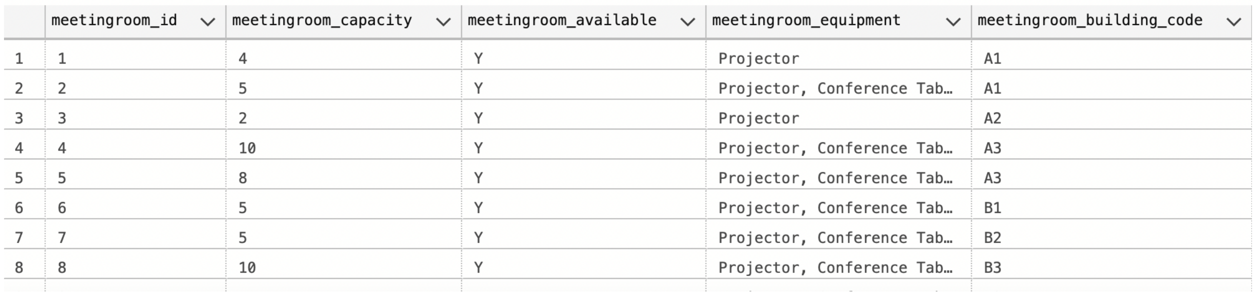
	meetingroom_id	meetingroom_capacity	meetingroom_available	meetingroom_equipment	meetingroom_building_code	building_id	building_code	building_city
1	1	4	Y	Projector	A1	1	A1	New York
2	2	5	Y	Projector, Conference T…	A1	1	A1	New York
3	15	4	Y	Projector, Conference T…	A1	1	A1	New York
4	11	2	Y	Projector	A2	2	A2	New York
5	3	2	Y	Projector	A2	2	A2	New York
6	4	10	Y	Projector, Conference T…	A3	3	A3	New York
7	5	8	Y	Projector, Conference T…	A3	3	A3	New York
8	6	5	Y	Projector, Conference T…	B1	4	B1	New York
9	14	4	Y	Projector	B1	4	B1	New York
1…	7	5	Y	Projector, Conference T…	B2	5	B2	New York
1…	8	10	Y	Projector, Conference T…	B3	6	B3	New York

TRIGGERS:

```
--Trigger
drop trigger if exists t_bookings_meeting_id_insert
go

create trigger t_bookings_meeting_id_insert
on bookings
after INSERT as
    begin
        declare @meetingroomid INT

        select @meetingroomid = inserted.booking_meetingroom_id
        from inserted

        update meetingrooms
        set meetingroom_available = 'N'
        where meetingroom_id = @meetingroomid
END

drop trigger if exists t_bookings_meeting_id_delete
go

create trigger t_bookings_meeting_id_delete
on bookings
after DELETE as
    begin
        declare @meetingroomid INT

        select @meetingroomid = deleted.booking_meetingroom_id
        from deleted
        update meetingrooms
        set meetingroom_available = 'Y'
        where meetingroom_id = @meetingroomid
END

select * from bookings
select * from meetingrooms
```



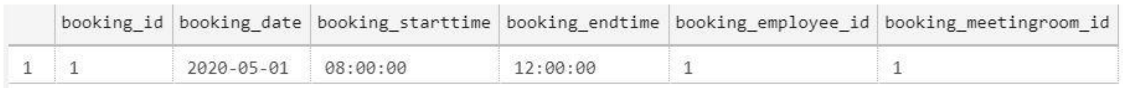
	booking_id	booking_d…	booking_s…	booking_e…	booking_e…	booking_m…



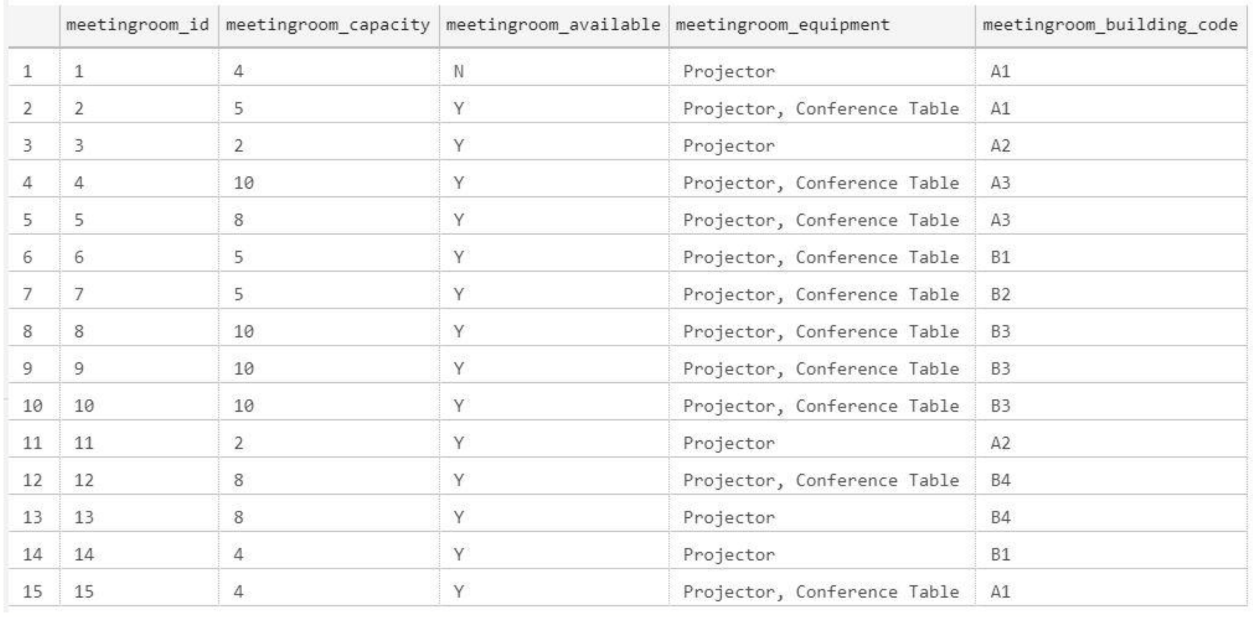
	meetingroom_id	meetingroom_capacity	meetingroom_available	meetingroom_equipment	meetingroom_building_code
1	1	4	Y	Projector	A1
2	2	5	Y	Projector, Conference Tab…	A1
3	3	2	Y	Projector	A2
4	4	10	Y	Projector, Conference Tab…	A3
5	5	8	Y	Projector, Conference Tab…	A3
6	6	5	Y	Projector, Conference Tab…	B1
7	7	5	Y	Projector, Conference Tab…	B2
8	8	10	Y	Projector, Conference Tab…	B3

INSERTING INTO BOOKINGS:

```
--Inserting into booking

BEGIN
BEGIN TRY
    BEGIN TRANSACTION
        insert into bookings(booking_date,booking_starttime,booking_endtime,booking_employee_id,
booking_meetingroom_id)
        VALUES
        ('2020-05-01','08:00:00','12:00:00',1,1)
        COMMIT
END TRY
BEGIN CATCH
    ROLLBACK ;
    THROW
END CATCH
END

select * from bookings
select * from meetingrooms
```



	booking_id	booking_date	booking_starttime	booking_endtime	booking_employee_id	booking_meetingroom_id
1	1	2020-05-01	08:00:00	12:00:00	1	1



	meetingroom_id	meetingroom_capacity	meetingroom_available	meetingroom_equipment	meetingroom_building_code
1	1	4	N	Projector	A1
2	2	5	Y	Projector, Conference Table	A1
3	3	2	Y	Projector	A2
4	4	10	Y	Projector, Conference Table	A3
5	5	8	Y	Projector, Conference Table	A3
6	6	5	Y	Projector, Conference Table	B1
7	7	5	Y	Projector, Conference Table	B2
8	8	10	Y	Projector, Conference Table	B3
9	9	10	Y	Projector, Conference Table	B3
10	10	10	Y	Projector, Conference Table	B3
11	11	2	Y	Projector	A2
12	12	8	Y	Projector, Conference Table	B4
13	13	8	Y	Projector	B4
14	14	4	Y	Projector	B1
15	15	4	Y	Projector, Conference Table	A1

10. TEAM LOG

Team Member	Work Done
Suyash Gupte	Collaborated with Yash making the logical model, Created Tables, Created Application Screens on Power Apps
Yash Shimpi	Coordinated with the team is completing the project in timely manner, Collaborated with Suyash making the logical model, Wrote the script for Up/Down for queries and inserting data into the database.
Hemang Gala	Made the conceptual model, Wrote the documentation for the project and made PPT.

11. REFERENCES

1. Fudge, Michael A. (2021). Applied Database Management.