Resort Management System

Dinesh Sai Pappuru

Sejal Hirji Gothi

DS 5110 - Introduction to Data Management

Instructor - Yehoshua Roi

Fall 2022

Table of Contents

Introduction:

The main objective of our project is to create a database management system for a resort. A resort is a beehive of multiple activities, including the front desk, reservations, booking, inventory management, quality management, housekeeping, security, employee management and more. As a result, a resort requires a well-organized management system that can readily handle all its activities and data. Our resort management system keeps track of customers and their membership details, rooms, room types, employees, billings, other services that a customer can reserve things like laundry, meal-plans, games, spa, room services for a luxurious resort.

Database design:

We started the project by creating an ER diagram that included all the necessary tables and their corresponding columns for a resort. Table relationships were also added. The tables were maintained in a manner that prevents data redundancy and is simple to understand to people working at a resort. ER Diagram for the system is available below.

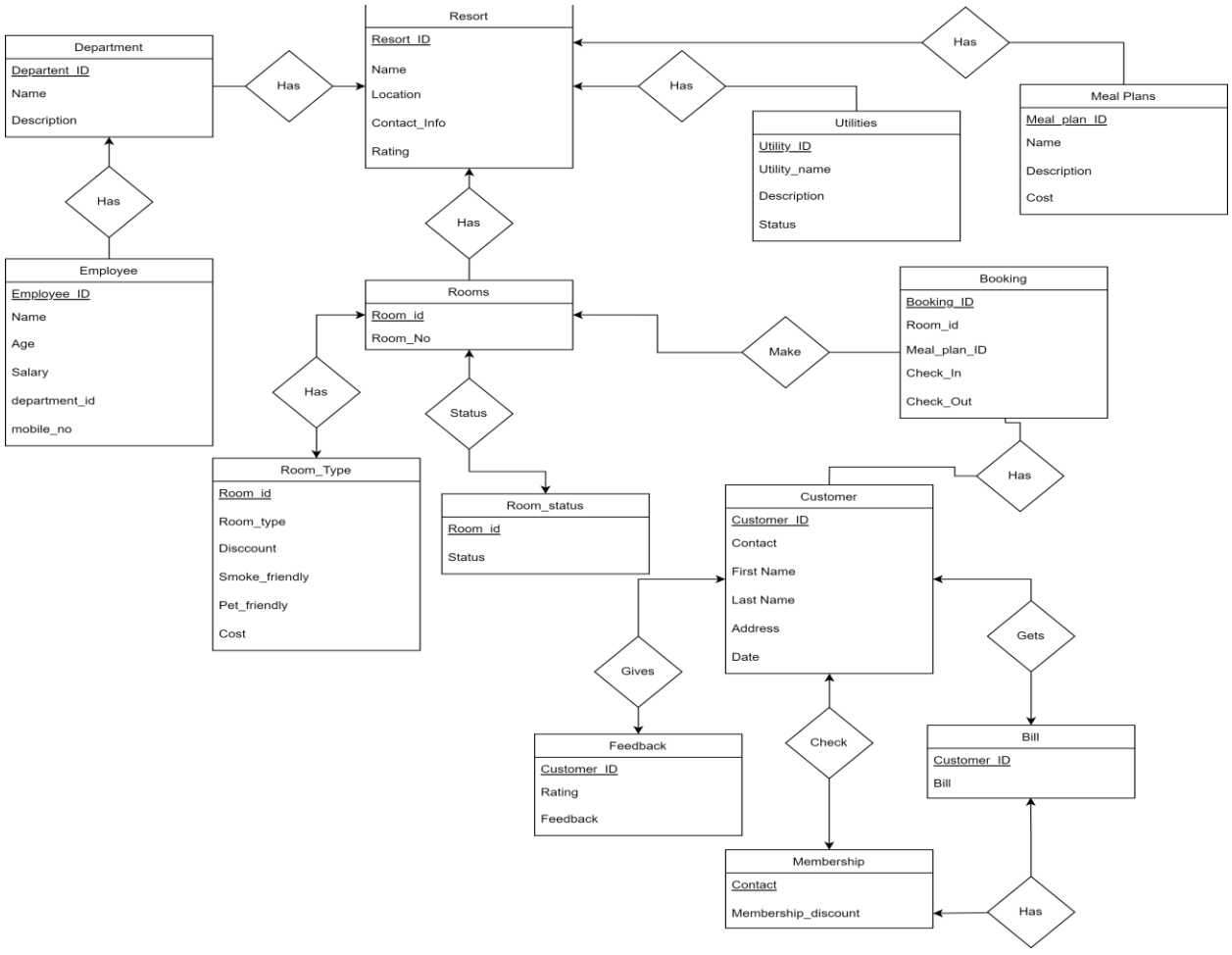


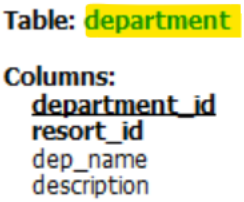
Figure 1: ER Diagram - Resort management System

Tables Definitions:

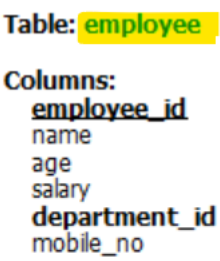
We created following tables with columns mentioned below:

1. **Resort**: A table named resort was created and populated with information such as resort ID, name, address, phone number, and ratings. Primary key for the table is resort ID This table was designed to allow us flexibility if, in the future, business grows, and we need to include another resort in the same database.

2. **Department**: This table was created with following columns: department ID, resort ID, department name, description. Primary key for the table is department ID



3. **Employee**: This table was created with following columns: employee ID, name, age, salary, department ID, mobile number. Primary key of the table is employee ID and foreign key for the table is department ID which references to department table's department ID with one-to-many relationship.



4. **Rooms**: This table was created to add details about rooms available in resort. It has following columns: room number, room ID, resort ID. Primary key for the table is room ID and foreign key for the table is resort ID which references to resort table's resort ID with one-to-many relationship.

5. **Room_Type**: This table was created to add details about room type depending on room ID. It has following columns: room ID, room type, discount, smoke friendly, pet friendly, cost. Foreign key for the table is room ID which references to rooms table's room ID with one-to-one relationship.

Table: room_type

Columns:
- **room_id**
- room_type
- discount_percent
- smoke_friendly
- pet_friendly
- cost

6. **Room_Status:** This table was created to get status of room if its available for not. It contains following columns: room ID, status. Foreign key for the table is room ID which references to room ID of rooms table with one-to-one relationship, and it is a weak entity.

Table: room_status

Columns:
- **room_id** int PK
- status tinyint(1)

7. **Meal_Plans**: This table gives information about meal plans available for customers. It contains following columns: meal plan ID, resort ID, name, description, cost. Primary key of the table is meal plan ID. Foreign key is resort ID which references to resort ID of resort table with one-to-many relationship.

Table: meal_plan

Columns:
- **meal_plan_id**
- **resort_id**
- name
- description
- cost

8. **Utilities**: This table gives information about utilities available for customers. It contains following columns: utility ID, resort ID, name, status. Primary key of the table is utility ID. Foreign key is resort ID which references to resort ID of resort table with one-to-many relationship.

Table: utilities

Columns:
- **utility_id**
- **resort_id**
- utility_name
- decription
- status

9. **Customer**: This table was created to store customer information. It contains following columns: customer ID, first name, last name, address, contact, date. Primary key for the table is customer ID and contact is a unique key.

Table: customer

Columns:
- **customer_id**
- first_name
- last_name
- address
- **contact**
- date

10. **Membership**: This table was created to store customer's membership information. It contains following columns: customer contact, discount. Foreign key for the table is customer contact which references to contact of customer table with one-to-one relationship.

Table: membership

Columns:

contact	bigint PK
discount	int

11. **Feedback**: This table was created to store feedback from each customer for future improvements. It contains following columns: customer ID, rating, feedback message. Foreign key of the table is customer ID which references to customer ID of customer table.

Table: feedback

Columns:
- **customer_id**
- rating
- feedback

12. **Bill**: This table gives information about billings for a particular customer. It contains following columns: customer ID, bill Amount. Foreign key of the table is customer ID which references to customer ID of customer table with one-to-one relationship.

Table: bill

Columns:
- **customer_id**
- bill

13. **Booking**: This table was created to store booking information of each customer. It contains following columns: booking ID, customer ID, room ID, check in date, check out date, meal plan. Primary key for the table is booking ID. Foreign key of the table is customer ID which references to customer ID of customer table and room ID which references to room ID of rooms table.

Table: booking

Columns:
- **booking_id**
- **customer_id**
- **room_id**
- check_in
- check_out
- meal_plan_id

Normalization: Data Normalization is a process that helps to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management. It is important that data is at least in third normal form without any transitive partial dependency. Hence, we normalized out data to third normal form.

Data collection:
We created our own data due to unavailability of data on the web for our project.

Application description:
After creating database based on final ERD, we created python-based command line application. We used python because it has many open-source libraries which help in securely connecting and maintaining connection with the mysql database.

Some of the libraries which we used in python are as follows:

1. Mysql connector: Establish connection from python to database.
2. Getpass: To hide passwords to enhance security.
3. Pandas: To show the data in a tabular format from sql queries.
4. Numpy: To alter table columns based on relevant conditions.
5. plotText, Matplot: To show pictorial representation of the data/results.

Main features provided by the application are as follows:

1. Secure Connection to Database
2. Add/Update/Delete rows from table and prevention to SQL injection
3. Showing statistical data from database
4. Ability to add multiple resort data in same database structure
5. Implemented stored procedures, functions, triggers, views and joins between multiple tables to show result effectively.

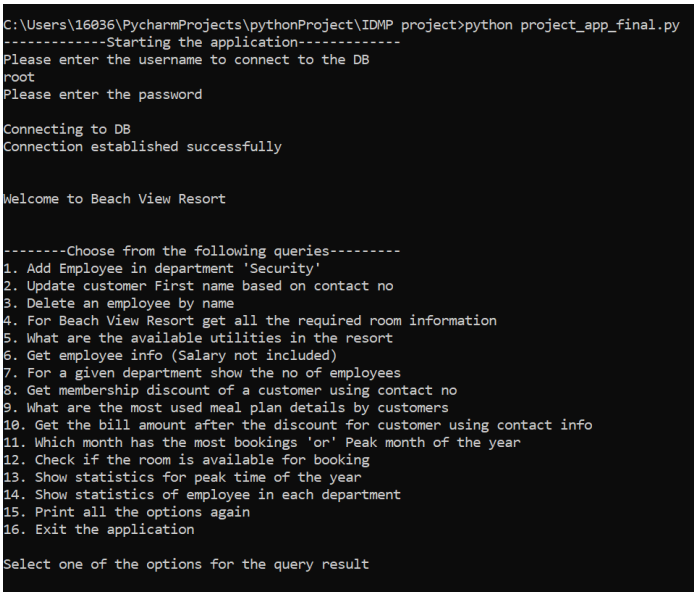


Figure 2: Command-line application

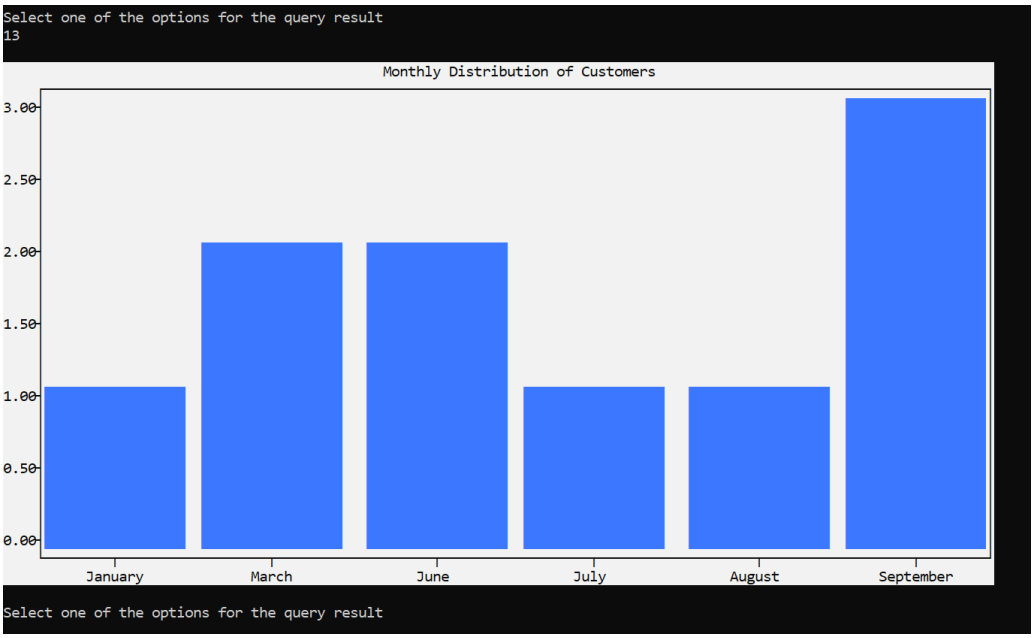


Figure 3: Statistics showing monthly distribution of customers

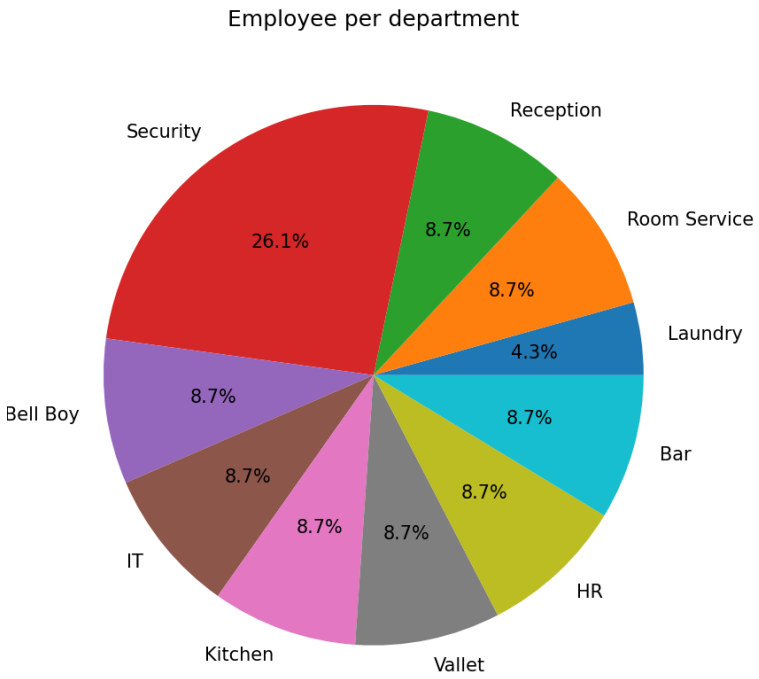


Figure 4: Statistics showing employee distribution in each department

Conclusions / future directions:

This project was very useful in demonstrating the value of having a database management system that can handle your data and how it makes it simple to perform a query and obtain the desired results. However, it's crucial that we establish logical, normalization-compliant links between tables. We developed a command-line tool that enables users to add, delete, or edit any record from a certain table. We would want to continue developing a more aesthetically pleasing and user-friendly UI/UX-based application if given additional

time. Below is a prototype for the first iteration of a UI-based application. For future DS5110 students, we would advise them to focus more on creating an efficient ER diagram and database and everything else will fall in place once that part is done.

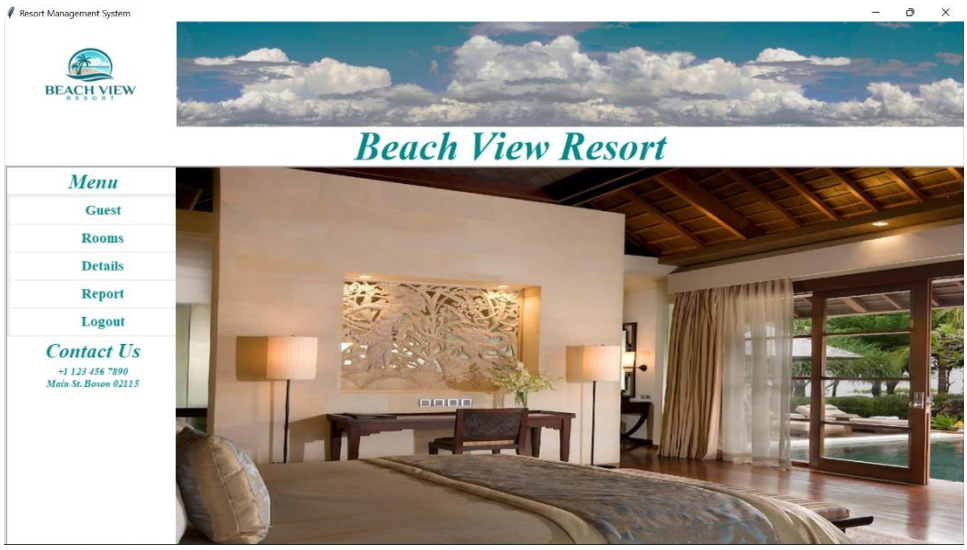
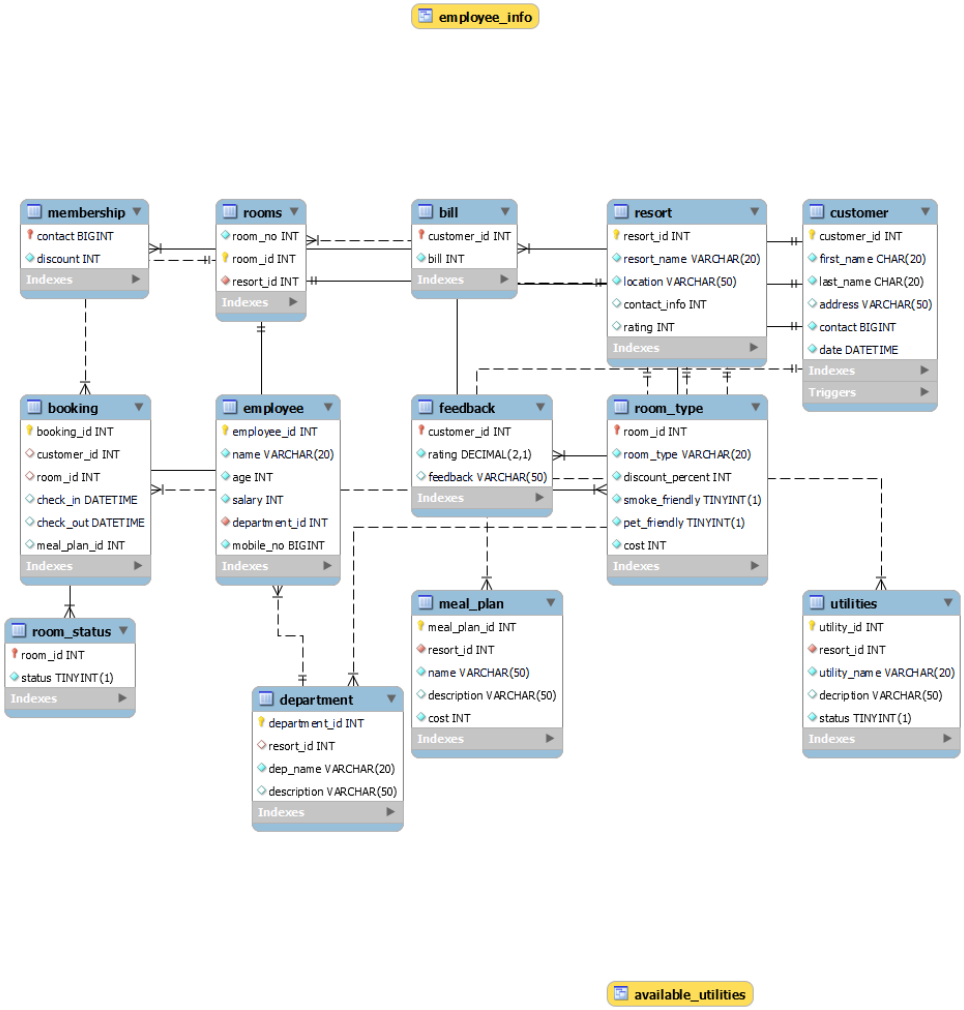


Figure 5: GUI based Application Prototype

Appendix

Schema Diagram:



SQL DDL Queries:

```
CREATE TABLE `bill` (
  `customer_id` int NOT NULL,
  `bill` int NOT NULL,
  PRIMARY KEY (`customer_id`),
  CONSTRAINT `bill_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`)
)
CREATE TABLE `booking` (
  `booking_id` int NOT NULL AUTO_INCREMENT,
  `customer_id` int DEFAULT NULL,
```

```
  `room_id` int DEFAULT NULL,
  `check_in` datetime DEFAULT NULL,
  `check_out` datetime DEFAULT NULL,
  `meal_plan_id` int DEFAULT NULL,
  PRIMARY KEY (`booking_id`),
  KEY `room_id` (`room_id`),
  KEY `booking_ibfk_2` (`customer_id`),
  CONSTRAINT `booking_ibfk_1` FOREIGN KEY (`room_id`) REFERENCES `rooms` (`room_id`),
  CONSTRAINT `booking_ibfk_2` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`)
)

CREATE TABLE `customer` (
  `customer_id` int NOT NULL AUTO_INCREMENT,
  `first_name` char(20) NOT NULL,
  `last_name` char(20) NOT NULL,
  `address` varchar(50) DEFAULT NULL,
  `contact` bigint NOT NULL,
  `date` datetime NOT NULL,
  PRIMARY KEY (`customer_id`),
  UNIQUE KEY `contact_UNIQUE` (`contact`)
)

CREATE TABLE `department` (
  `department_id` int NOT NULL AUTO_INCREMENT,
  `resort_id` int DEFAULT NULL,
  `dep_name` varchar(20) NOT NULL,
  `description` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`department_id`),
  KEY `resort_id` (`resort_id`),
  CONSTRAINT `department_ibfk_1` FOREIGN KEY (`resort_id`) REFERENCES `resort` (`resort_id`)
)
```

```
CREATE TABLE `employee` (
  `employee_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  `age` int NOT NULL,
  `salary` int NOT NULL,
  `department_id` int NOT NULL,
  `mobile_no` bigint NOT NULL,
  PRIMARY KEY (`employee_id`),
  KEY `department_id` (`department_id`),
  CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`department_id`) REFERENCES `department` (`department_id`)
)

CREATE TABLE `feedback` (
  `customer_id` int NOT NULL,
  `rating` decimal(2,1) NOT NULL,
  `feedback` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`customer_id`),
  CONSTRAINT `feedback_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`)
)

CREATE TABLE `meal_plan` (
  `meal_plan_id` int NOT NULL AUTO_INCREMENT,
  `resort_id` int NOT NULL,
  `name` varchar(50) NOT NULL,
  `description` varchar(50) DEFAULT NULL,
  `cost` int NOT NULL,
  PRIMARY KEY (`meal_plan_id`),
  KEY `resort_id` (`resort_id`),
  CONSTRAINT `meal_plan_ibfk_1` FOREIGN KEY (`resort_id`) REFERENCES `resort` (`resort_id`)
```

```
)

CREATE TABLE `membership` (
  `contact` bigint NOT NULL,
  `discount` int NOT NULL,
  PRIMARY KEY (`contact`),
  CONSTRAINT `membership_ibfk_1` FOREIGN KEY (`contact`) REFERENCES `customer` (`contact`)
)

CREATE TABLE `resort` (
  `resort_id` int NOT NULL AUTO_INCREMENT,
  `resort_name` varchar(20) NOT NULL,
  `location` varchar(50) NOT NULL,
  `contact_info` int DEFAULT NULL,
  `rating` int DEFAULT NULL,
  PRIMARY KEY (`resort_id`)
)

CREATE TABLE `room_status` (
  `room_id` int NOT NULL,
  `status` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`room_id`),
  CONSTRAINT `room_status_ibfk_1` FOREIGN KEY (`room_id`) REFERENCES `rooms` (`room_id`)
)

CREATE TABLE `room_type` (
  `room_id` int NOT NULL,
  `room_type` varchar(20) NOT NULL,
  `discount_percent` int NOT NULL,
  `smoke_friendly` tinyint(1) NOT NULL DEFAULT '1',
  `pet_friendly` tinyint(1) NOT NULL DEFAULT '1',
```

```
  `cost` int NOT NULL,
  PRIMARY KEY (`room_id`),
  CONSTRAINT `room_type_ibfk_1` FOREIGN KEY (`room_id`) REFERENCES `rooms` (`room_id`)
)

CREATE TABLE `rooms` (
  `room_no` int NOT NULL,
  `room_id` int NOT NULL AUTO_INCREMENT,
  `resort_id` int NOT NULL,
  PRIMARY KEY (`room_id`),
  KEY `resort_id` (`resort_id`),
  CONSTRAINT `rooms_ibfk_1` FOREIGN KEY (`resort_id`) REFERENCES `resort` (`resort_id`)
)

CREATE TABLE `utilities` (
  `utility_id` int NOT NULL AUTO_INCREMENT,
  `resort_id` int NOT NULL,
  `utility_name` varchar(20) NOT NULL,
  `decription` varchar(50) DEFAULT NULL,
  `status` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`utility_id`),
  KEY `resort_id` (`resort_id`),
  CONSTRAINT `utilities_ibfk_1` FOREIGN KEY (`resort_id`) REFERENCES `resort` (`resort_id`)
)
```

Views:

```
CREATE VIEW employee_info AS
    SELECT
        name, age, mobile_no
    FROM
        Employee
```

```
CREATE VIEW available_utilities AS
   SELECT
      utility_name
   FROM
      utilities
   WHERE
      status = 1;
```

Triggers:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `update_booking_on_customer` AFTER INSERT ON `customer` FOR EACH ROW begin
            declare temp int;
      set @temp = (select MAX(customer_id) from customer);
            insert into booking(customer_id) values (@temp);
end
```

Functions:

```
Delimiter $$

create function bill_amount_after_discount(cont bigint)
returns int
deterministic
begin
      declare discounted_bill int;
   select (b.bill - ((m.discount/100)* b.bill)) into discounted_bill
   from membership m, customer c, bill b
   where m.contact = c.contact and
            c.customer_id = b.customer_id and
      m.contact = cont;
      return discounted_bill;
end $$
```

```
delimiter ;
```

```
delimiter $$
create function room_availibity(room_number int)
returns varchar(20)
deterministic
begin
    declare status int;
   declare output varchar(20);
   select rs.status into status
   from rooms r, room_status rs
   where r.room_id = rs.room_id and
              r. room_no = room_number;

    if status = 1 then
              set output = 'Available';
   else
              set output = 'Not Available';
    end if;
   return output;
end $$
delimiter ;
```

```
delimiter $$
create function peak_time_of_the_year()
returns varchar(20)
deterministic
begin
    declare month varchar(20);
    select monthname(check_in) into month
    from booking
```

```
    group by Month(check_in)
    order by Month(check_in) desc
    limit 1;
    return month;
end $$
delimiter ;
```

Stored Procedure:

```
delimiter $$;
create procedure get_membership_discount(in contact bigint)
begin
    select first_name, last_name, discount
            from customer c, membership m
    where c.contact = m.contact
end$$
delimiter ;
```

```
delimiter $$
create procedure get_most_used_meal_plan_details()
begin
    select *
  from booking b, meal_plan m
  where b.meal_plan_id = m.meal_plan_id
  group by b.meal_plan_id
  order by b.meal_plan_id desc
  limit 1;
end$$
delimiter ;
```

```
delimiter $$
```

```
create procedure get_department_employees( in dept_name varchar(20))
begin
      select d.dep_name, count(e.employee_id) as no_of_employees
   from department d, employee e
   where d.department_id = e.department_id and d.dep_name = dept_name
   group by e.department_id;

end$$
delimiter ;
```