Wolf Inns Database System

For Wolf Inns Hotel Chain

CSC 540: Database Management Concepts and Systems

Project Report #2

Project Team # 7:
Sachin Kumar
Shubhankar Reddy Katta
Shyam Prasad Katta
Qaiss Khan Alokozai

Assumptions

1. Executive Manager is administrator of all the hotels belonging to WolfInn chain of hotels.
2. There is only one Executive Manager for WolfInn chain of hotels.
3. We will be having a global Services table with services description defined for service id used in the project.
4. Each hotel has only one manager.
5. Manager is uniquely identified by manager_staff_id provided in hotel entity.
6. For all the local ER diagrams except that of Executive Manager, the diagram is represented for single hotel only.
7. Reports are not shown in the ER diagrams as we will be generating reports by joins across multiple tables, so since there is no dedicated table associated with it, hence it is not worthwhile showing it on ER diagram.
8. In every local ER diagram, we are just showing functionality applicable to that user and other detailed functionalities are shown in respective local diagrams.
9. Manager is the administrator of a particular hotel records, to which he is associated with.
10. Every relation will have a table associated with it storing keys of entities to the relations.
11. A master list of Services offered by a particular hotel and their associated prices is created.
12. All Services are offered only during the office hours and the customer has to explicitly avail the service by requesting the associated service staff personnel.
13. Dedicated Staff means that staff is assigned to a reservation for its entire duration of existence, with that particular staff is unable to serve other reservations for that duration. However non-dedicated staff can serve more than one reservations.
14. Service charge is applicable to each time the service is availed. The same charge will be reflected in a particular customer reservation.
15. All the individual Services rates is assumed to be uniform across all the hotels of WolfInn chain. Each hotel which provides a service has the same service charges as that of other hotels in WolfInn chain, is our assumption.
16. A flag named “Serving_premium” column will be maintained in service staff table to keep track of people who’re assigned to the presidential suite and stores the reservation id as long as they’re serving that particular suite/reservation.
17. Prices vary by location and type of services offered. We maintain two global tables price_by_location and price_by_room_class.
18. A room unit prices is calculated by summation of price by its location and class of the services of a room.
19. Executive manager can add entries in the prices tables by region and category. In case of price absence we use default prices.
20. It is assumed that these two tables(price_by_location and price_by_room_class) are global tables and are not represented in any ER diagrams.
21. It is assumed that the Front Desk Representative checks if the customer preferred room category is available. If available makes the reservation, if not let the customer know the available room types. Later, makes the reservation accordingly.

22. There are two attributes under staff, namely department and job_title. The distinction between them is that one or more people with different job_title can be present in same department. Detailed examples can be seen in insert statement of staff under SQL sub section(3).
23. When a entry for hotel is inserted then by default manager_staff_id will be 1, which we will be updating with another manager's staff id as required.
24. Some of the records are inserted and deleted to complete the operations listed in narrative, and some are added in some specific tables to show up in the results of operations,so overall consistency of records across different tables might differ.

1) Database Schema

The schemas are all in 3NF form because all entities have unique IDs associated with them which makes it easier to identify a specific tuple.

All functional dependencies are obvious because of the IDs associated with all entities.

Executive_Manager(<u>Manager_Name</u>) -
Manager_Name -> Manager_Name
This relation is in BCNF (and thus 3NF) because it contains one attribute which is the key.

Hotel(<u>Hotel_ID</u>, Hotel_Name, Manager_Staff_Id, Hotel_Phone_no, City, Street_Name) -
Hotel_ID -> Hotel_Name, Manager_Staff_Id, Hotel_Phone_no, City, Street_Name holds because each ,Hotel is identified by a unique id, has a Hotel_Name, Manager_Staff_Id, Hotel_Phone_no, City, Street_Name.
This relation is in BCNF (and thus 3NF) for the following reasons:

1. Any combination of Hotel_Name, Manager_Staff_Id, Hotel_Phone_no, City, Street_Name cannot functionally determine Hotel_id ,mainly because there can be different hotels in the Wolfinn chain with multiple hotelid's which might also have same combination of these attributes.
2. Hotel_Phone_no,City,Street_Name -> Hotel_Name or Manager_Staff_Id does not hold because there can be different hotels with either shared phone number,city or same street_name.

Staff(<u>Staff_ID</u>, <u>Hotel_ID</u>,Age, Address, Department, Job_title, Phone_number, Staff_name)
Staff_ID, Hotel_ID -> Age, Address, Department, Job_title, Phone_number, Staff_name as each staff member can be uniquely identified using the (Staff_ID,Hotel_ID) tuple unique to every staff member throughout all hotels.
This relation is in BCNF (and thus 3NF) for the following reasons:

1. Staff id alone cannot functionally determine Staff_Name,Department,Job_title and Phone_number since there can be same staffid in different hotels of the chain.
2. No combination of Age, Address, Department, Job_title, Phone_number, Staff_name can functionally determine Staff_id and Hotel_id since there can be different hotels with the same combination of values of these attributes can have different combination of staff_id and hotel

3. Any combination of Age, Address, Department, Job_title, Phone_number, Staff_name -> Staff_ID as there can be an employee working for 2 different branches of WolfInn by different Staff ID for each hotel, hence these can't uniquely identify that employee
4. Any combination of Staff_ID, Age, Address, Department, Job_title, Phone_number, Staff_name -> Hotel_ID doesn't hold as there can be an employee working for 2 different branches of WolfInn by same Staff_ID, hence these can't uniquely identify that employee

FDR_Staff(Staff_ID, Hotel_ID)
The relation is in BCNF as Staff_Id, Hotel_Id ->Staff_Id ; Staff_Id, Hotel_Id ->Hotel_Id ;
We can't determine Staff_Id from Hotel_Id as there can be many staff working for a single hotel and the vice versa isn't true as the staff aren't unique across the hotels and need the Hotel_Id attribute to uniquely determine the staff belonging to a particular hotel.

Manager_hotel_association(**Manager_name, Hotel_id**)
Manager_name,Hotel_id->Manager_name,Hotel_id
This relation is in BCNF (and thus 3NF) because it contains two attributes.

Service_Staff(Staff_ID, Hotel_ID, Serving_Premium)
Note: A flag named "Serving_premium" column will be maintained in service staff table to keep track of people who're assigned to the presidential suite.
Staff_ID,Hotel_Id -> Serving_Premium holds because every serving premium is uniquely identified by Staff_ID and Hotel_Id
The relation is in BCNF(thus 3NF) for the following reasons:
1. Serving_Premium->Staff_Id or Serving_Premium->Hotel_id does not hold because there can be same serving premium flag value for different hotels or staff_id.

Customer_details(Customer_email, SSN, Customer_Name, DOB, Phone_Number, Address)
SSN->Customer_email, Customer_Name, DOB, Phone_Number, Address holds because each customer is uniquely identified by a SSN.
The relation is in BCNF(thus 3NF) for the following reasons:
1. any combination of Customer_email, Customer_Name, DOB, Phone_Number cannot functionally determine SSN as it is theoretically possible for 2 twins with same name to be sharing the same phone and mail id, hence its not possible to identify them uniquely without SSN.

Customer(Customer_Id, SSN)
Customer_ID -> SSN,holds as each customer id has to be associated with an SSN.
This relation is in BCNF (and thus 3NF) because it contains two attributes.

Reservation(Reservation_Id, Hotel_Id, Checkin_time, Checkout_time)
Reservation_Id,Hotel_Id -> Checkin_time, Checkout_time as the reservation ID of a particular hotel uniquely identifies each reservation.
The relation is in BCNF(thus 3NF) for the following reasons:

1. any combination of Checkin_time, Checkout_time -> Reservation_Id doesn't hold as there can be multiple reservations made at the same time, hence it won't be possible to identify each one uniquely.
2. any combination of Checkin_time, Checkout_time -> Hotel_Id doesn't hold as there can be reservations made in multiple hotels at the same time, hence it's not possible to identify to which hotel a reservation belongs.

Room(Room_no, Hotel_Id, Category, Capacity, Availability) -
Hotel_Id, Room_no -> Category, Capacity, Availability holds as the tuple (H_ID, Room_no) uniquely identifies each room in every hotel and hence determines the room's price, category (the one assigned during its creation), capacity and availability.
The relation is in BCNF(thus 3NF) for the following reasons:

1. Capacity -> availability doesn't hold as the the capacity of a room can't functionally determine whether it's occupied.
2. any combination of Category, Capacity, Availability, Room_no -> H_ID doesn't hold as there can be 2 rooms with the same category, capacity, availability and same room numbers among different hotels and can't be used to uniquely identify a hotel.
3. any combination of Category, Capacity, Availability, Hotel_Id -> Room_no doesn't hold as 2 rooms within a single hotel can have the same attributes and hence can't be used to uniquely identify the room number.

Billing_info(amount, Billing_ID)
Billing_ID -> amount holds as every bill is associated with an amount
This relation is in BCNF (and thus 3NF) because it contains two attributes.

Payment_details(Payment_ID, Billing_Address, Payment_Type, CCC_Details)
Payment_Id -> Billing_Address, Payment_Type, CCC_Details holds as the Payment_Id uniquely identifies each card/cheque/cash payment used for each transaction.
The relation is in BCNF(thus 3NF) for the following reasons:

1. any combination of Billing_Address, Payment_Type, CCC_Details -> Payment_Id doesn't hold as there can be 2 transactions paid with cash by the same person on different occasions, hence this combination can't determine the payment_Id.

Payment_relation(Billing_ID, Payment_ID)
Billing_id,Payment_ID -> Billing_id,Payment_ID
This relation is in BCNF (and thus 3NF) because it contains two attributes.

Check_In_Info(Hotel_ID, Staff_ID, Customer_ID, Room_no, Reservation_ID, Billing_ID)
Hotel_ID, Staff_ID, Customer_ID, Reservation_ID, Billing_ID->Hotel_ID, Staff_ID, Customer_ID, Reservation_ID, Billing_ID, Room_no
The relation is in BCNF(thus 3NF) for the following reasons:

1. It's a trivial functional dependency with all the attributes are needed in order to determine all the other attributes.
2. Hotel_ID, Staff_ID, Customer_ID, Reservation_ID, Billing_ID -> Room_no holds as the Reservation is unique within the context of a hotel and can uniquely identify the room

assigned for that reservation.

3. Any subcombination of (Hotel_ID, Staff_ID, Customer_ID, Reservation_ID,Billing_Id), Room_no -> (The excluded attribute on LHS) doesn't hold as the reservation could be within any hotel if hotel_Id was excluded, it could be any staff if Staff_Id was excluded, it could be by any customer if Customer_Id was excluded, it could be for any reservation if reservation_Id was excluded and could be any billing amount if billing_Id was excluded.

Services(Service_name, Service_Price, <u>Service_ID</u>)
Service_ID -> Service_name, Service_Price holds, as every service is uniquely identified by its Service_ID
The relation is in BCNF(thus 3NF) for the following reasons:

1. Service_name or Service_Price -> Service_ID doesn't hold as there can be services with same name being offered at different prices across the hotels; service price is not unique to each service, hence it also can't be used to identify each service uniquely.

Services_Used(Staff_Id, Hotel_Id, <u>Service_Instance_Id</u>, Reservation_Id, Service_Id)
Service_Instance_Id -> Staff_ID, H_ID, Reservation_ID, Service_ID -> Service_Instance_Id is a unique identifier which determines the staff belonging to a particular hotel who has serviced the reservation of the hotel with the service identified by service_Id.
The relation is in BCNF(thus 3NF) for the following reasons:

1. Any combination Staff_ID, H_ID, Reservation_ID, Service_ID -> Service_Instance_Id doesn't hold as the service may have been availed at same hotel, by same Staff, by same reservation but at a different point in time.

<u>Rooms_Price_Listing(City</u>, <u>Category</u>, Price)
City,Category->Price holds since every combination of city and room category will determine or will be associated with a room price.
The relation is in BCNF(thus 3NF) for the following reasons:

1. Price -> City, Category doesn't hold as the price of a "Deluxe" room in a city with high cost of living might be the same for "Presidential" room in a city with low cost of living.

2) Design Decisions

The mechanical approach was used to create the global schema with a few exceptions.

1. Each entity set was made into a relation with the same set of attributes
2. Relationships were replaced by a relation whose attributes are the keys for the connected entity sets

The E/R viewpoint was used to convert the subclasses into relations. This method was used so:

1. The system can differentiate between manager, front-desk representatives, and service staff.

2. All people including manager, front-desk representatives, and service staff can be referenced from one single table (Staff).

Many-to-one relationships were combined with other relations. Combining relations in this way makes it more efficient to answer queries that involve attributes of one relation than to answer queries involving attributes of several relations.

***Hotel*(Hotel_id, Hotel_name, Hotel_phone_no, City, Manager_Staff_id, Street_name)**
Hotel_id (Primary Key) – unique identifier
Hote_name (NOT NULL) – identification purposes
Hotel_phone_no - required for inter hotel communication in the chain of hotels
City (NOT NULL) – required for getting information about the hotels present in a particular city.
Manager_Staff_id (NOT NULL) **-** required for knowing the primary contact person for a hotel holding highest clearance level access to systems functionalities for a particular hotel
Street_name (NOT NULL) **-** required to locate hotel's physical address

***Customer_details*(SSN, Customer_email, Customer_name, DOB, Phone_number, Address)**
SSN (Primary Key) - To uniquely identify customer details
Customer_email (NOT NULL) - email-id is required for communicating customer about bills or offers
Address (NOT NULL) - required to send promotional offers to the customer.
Customer_name (NOT NULL) - name of the customer is required to generate bills.
DOB (NOT NULL) - date of birth of customer is required to identify whether the customer is a minor or adult to restrict the registration process
Phone_number (NOT NULL) - Phone number of the customer to send account related notifications.

***Customer*(Customer_id, SSN)**
customer_id (Primary Key) - To uniquely identify customer.
SSN (Referential Integrity) - refers to other entities within the database (Customer_details)

***Check_in_info*(Hotel_id, Staff_id, Customer_Id, Room_no, Reservation_Id, Billing_Id)**
Hotel_id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (Room)
Staff_id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (Staff)
Customer_id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (Customer)
Room_no (Referential Integrity) - unique identifier and refers to other entities within the database (Room)
Reservation_id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (Reservation)
Billing_id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (Billing)

***Room*(Room_no, Hotel_id, Availability, Category, Capacity)**
Room_no (Primary Key) - unique identifier
Hotel_id (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (hotel)
Availability (NOT NULL) - required for checking availability of rooms
Capacity (NOT NULL) - required to make the appropriate number of reservations for a particular number of guests
Category (NOT NULL) - required to identify the category to which a room belongs to.

***Staff*(Staff_ID, Hotel_ID, Age, Address, Department, Job_title, Phone_number, Staff_name)**
Hotel_Id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (hotel)
Staff_id (Primary Key) - unique identifier.
Phone_number (NOT NULL) - required to contact the staff member.
Address (NOT NULL) - required to send important mail to the staff member.
Department (NOT NULL) - required to fetch/know the count of people in each department.
Job_title (NOT NULL) - required to fetch/know the people serving a particular role.
Staff_name (NOT NULL) - required to know who've served a customer during their stay.

***Services*(Service_name, Service_Price, Service_ID)**
Service_name (NOT NULL) - Used to print the service name on itemized receipt.
Service_price (NOT NULL) - Used to indicate the amount for the a single unit usage of the particular service.
Service_Id (Primary Key) - Used to identify each service uniquely.

***Services_Used*(Staff_ID, Hotel_ID, Service_Instance_Id, Reservation_ID, Service_ID)**
Service_Instance_Id (NOT NULL, Primary Key) - Required to uniquely identify each instance when a service was availed.
Staff_Id (Referential Integrity) - unique identifier and refers to other entities within the database (Staff)
Hotel_Id (Referential Integrity) - unique identifier and refers to other entities within the database (hotel)
Reservation_Id (Referential Integrity) - unique identifier and refers to other entities within the database (Reservation)
Service_Id (Referential Integrity) - unique identifier and refers to other entities within the database (Services)

***Reservation*(Reservation_id, Checkin_time, Hotel_id,Checkin_out)**
Reservation_id (Primary Key) - unique identifier
Hotel_id (Referential Integrity, Primary Key) - unique identifier and refers to other entities within the database (hotel)
Checkin_time (NOT NULL) - required to identify check-in time.
Checkout_time (NOT NULL) - required to identify checkout time.

***Payment_details*(Payment_Id, Payment_type, Billing_address, Ccc_details)**

Payment_id (Primary Key) - unique identifier
Payment_type (NOT NULL) - required to identify payment type used. Can take any of following values, cash or credit card or cheque or debit card
Billing_address (NOT NULL) - required to identify billing address
Ccc_details (NOT NULL) - required to identify cash/credit/cheque_number details

***Payment_relation*(Payment_Id, Billing_Id)**
Payment_id (Referential Integrity, Primary Key) - refers to other entities within the database (Payment)
Billing_id (Referential Integrity, Primary Key) - refers to other entities within the database (Billing)

***Billing_info*(Billing_id, Amount)**
Amount (NOT NULL) - required for storing the billing amount.
Billing_id (Primary Key) - unique identifier

***Executive_manager*(Manager_name)**
Manager_name (Primary Key) - unique identifier

***Manager_hotel_association*(Manager_name, Hotel_id)**
Manager_name (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (Executive_manager)
Hotel_id (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (hotel)

***FDR_staff*(Staff_Id, Hotel_Id)**
Hotel_Id (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (Staff)
Staff_Id (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (Staff)

***Service_staff*(Hotel_Id, Staff_Id, Serving_premium)**
Hotel_Id (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (Staff)
Staff_Id (Primary Key, Referential Integrity) - unique identifier and refers to other entities within the database (Staff)
Serving_premium (NOT NULL) - used to determine whether a staff member has been allocated to a presidential suite.

***Room_Price_Listing*(City, Category, Price)**
City (Primary Key, NOT NULL, Referential Integrity) - All the hotels that exist in a particular city will have a same price for a same room category.
Category (Primary Key, NOT NULL, Referential Integrity) - A room of the particular category must exist for its price to be determined.
Price (NOT NULL) - Used to calculate the nightly price for a room.

3) SQL Statements

Note : Used AUTO_INCREMENT in place of sequences since SEQUENCE is not supported by MariaDB version 5.5.56 provided by EOS server.

Create Tables

```
CREATE TABLE Hotel(
Hotel_id INT PRIMARY KEY AUTO_INCREMENT,
Hotel_name VARCHAR(40) NOT NULL,
Hotel_phone_no VARCHAR(10) NOT NULL,
City VARCHAR(30) NOT NULL,
Manager_Staff_id INT NOT NULL,
Street_name VARCHAR(128) NOT NULL
);

CREATE TABLE Customer_details(
SSN VARCHAR(10) PRIMARY KEY,
Customer_email VARCHAR(128) NOT NULL,
Customer_name VARCHAR(30) NOT NULL,
DOB VARCHAR(15) NOT NULL,
Phone_number VARCHAR(10) NOT NULL,
Address VARCHAR(50) NOT NULL
);

CREATE TABLE Customer(
Customer_id INT PRIMARY KEY AUTO_INCREMENT,
SSN VARCHAR(10),
CONSTRAINT ssn_fk FOREIGN KEY (SSN) REFERENCES Customer_details(SSN) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE Staff(
Staff_id INT AUTO_INCREMENT,
Hotel_id INT ,
Staff_name VARCHAR(50) NOT NULL,
Job_title VARCHAR(20) NULL,
Department VARCHAR(128) NOT NULL,
Age VARCHAR(30) NOT NULL,
Phone_number VARCHAR(10) NOT NULL,
Address VARCHAR(128) NOT NULL,
PRIMARY KEY(Staff_id,Hotel_id),
CONSTRAINT Staff_fk FOREIGN KEY(Hotel_id) REFERENCES Hotel(Hotel_id) ON DELETE CASCADE
);


CREATE TABLE Room(
Room_no INT,
Hotel_id INT NOT NULL,
Availability VARCHAR(15)NOT NULL,
```

```
Category VARCHAR(15) NOT NULL,
Capacity INT,
PRIMARY KEY(Room_no,Hotel_id),
CONSTRAINT room_fk FOREIGN KEY(Hotel_id) REFERENCES Hotel(Hotel_id) ON DELETE CASCADE
);

CREATE TABLE Reservation(
Reservation_id INT AUTO_INCREMENT,
Hotel_id INT NOT NULL,
Checkin_time TIMESTAMP NOT NULL,
Checkout_time TIMESTAMP NOT NULL,
PRIMARY KEY(Reservation_id,Hotel_id),
CONSTRAINT Reservation_fk FOREIGN KEY (Hotel_id) REFERENCES Hotel(Hotel_id) ON DELETE
CASCADE
);

CREATE TABLE Billing_info(
Billing_id INT PRIMARY KEY AUTO_INCREMENT,
Amount INT NOT NULL
);

CREATE TABLE Check_in_info(
Hotel_id INT,
Staff_id INT,
Customer_id INT,
Room_no INT,
Reservation_id INT,
Billing_id INT,
PRIMARY KEY(Hotel_id, Staff_id, Customer_id, Reservation_id, Billing_id),
CONSTRAINT checkin_room_fk FOREIGN KEY (Room_no,Hotel_id) REFERENCES
Room(Room_no,Hotel_id) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT checkin_Staff_fk FOREIGN KEY (Staff_id) REFERENCES Staff(Staff_id) ON DELETE
CASCADE ON UPDATE CASCADE,
CONSTRAINT checkin_Customer_fk FOREIGN KEY (Customer_id) REFERENCES
Customer(Customer_id) ON DELETE CASCADE ON UPDATE CASCADE,
);

CREATE TABLE Services(
Service_id INT PRIMARY KEY AUTO_INCREMENT,
Service_name VARCHAR(50) NOT NULL,
Service_price DECIMAL(6,2) NOT NULL
);

CREATE TABLE Services_used(
Hotel_id INT,
Staff_id INT,
Service_id INT,
Service_instance_id INT PRIMARY KEY AUTO_INCREMENT,
Reservation_id INT,
CONSTRAINT services_hotel_fk FOREIGN KEY(Hotel_id) REFERENCES Hotel(Hotel_id) ON DELETE
CASCADE,
CONSTRAINT services_staff_fk FOREIGN KEY(Staff_id) REFERENCES Staff(Staff_id) ON DELETE
CASCADE,
CONSTRAINT service_id_fk FOREIGN KEY(Service_id) REFERENCES Services(Service_id) ON
```

```
DELETE CASCADE,
CONSTRAINT reservation_id_fk FOREIGN KEY(reservation_id) REFERENCES
Reservation(reservation_id) ON DELETE CASCADE
);

CREATE TABLE Payment_details(
Payment_id INT PRIMARY KEY AUTO_INCREMENT,
Payment_type VARCHAR(20),
Billing_address VARCHAR(50),
Ccc_details VARCHAR(30)
);

CREATE TABLE Payment_relation(
Payment_id INT,
Billing_id INT,
PRIMARY KEY(Payment_id, Billing_id),
CONSTRAINT Payment_fk FOREIGN KEY(Payment_id) REFERENCES Payment_details(Payment_id) ON
DELETE CASCADE,
CONSTRAINT Billing_fk FOREIGN KEY(Billing_id) REFERENCES Billing_info(Billing_id) ON
DELETE CASCADE
);

CREATE TABLE Executive_manager(
Manager_name VARCHAR(20) PRIMARY KEY
);

CREATE TABLE Manager_hotel_association(
Manager_name VARCHAR(20),
Hotel_id INT,
PRIMARY KEY(Manager_name, Hotel_id),
CONSTRAINT Executive_manager_fk FOREIGN KEY(Manager_name) REFERENCES
Executive_manager(Manager_name) ON DELETE CASCADE,
CONSTRAINT Hotel_id_fk1 FOREIGN KEY(Hotel_id) REFERENCES Hotel(Hotel_id) ON DELETE
CASCADE
);

CREATE TABLE FDR_staff(
Hotel_id INT,
Staff_id INT,
PRIMARY KEY (Hotel_id, Staff_id),
CONSTRAINT fdr_staff_fk FOREIGN KEY(Staff_id,Hotel_id) REFERENCES
Staff(Staff_id,Hotel_id)
ON DELETE CASCADE
);

CREATE TABLE Service_staff(
Hotel_id INT,
Staff_id INT,
Serving_premium VARCHAR(1) NOT NULL,
PRIMARY KEY (Hotel_id, Staff_id),
CONSTRAINT service_staff_fk FOREIGN KEY(Staff_id,Hotel_id) REFERENCES
Staff(Staff_id,Hotel_id)
);
```

```
CREATE TABLE Rooms_price_listing(
City VARCHAR(20) ,
Category VARCHAR(20),
Price INT,
PRIMARY KEY(City,Category)
);
```

Inserts

Into Hotel

```
INSERT into Hotel(Hotel_name, Hotel_phone_no, City, Manager_Staff_id, Street_name)
values ('Wolfinn Raleigh', '9090908989', 'Raleigh', 1, '134 West Blvd');
INSERT into Hotel(Hotel_name, Hotel_phone_no, City, Manager_Staff_id, Street_name)
values ('Wolfinn Cary', '9090908989', 'Cary', 1, '122 East Hill');
INSERT into Hotel(Hotel_name, Hotel_phone_no, City, Manager_Staff_id, Street_name)
values ('Wolfinn Cary 1', '9090908765', 'Cary', 1, '123 North Hill');
INSERT into Hotel(Hotel_name, Hotel_phone_no, City, Manager_Staff_id, Street_name)
values ('Wolfinn Cary 2', '9090111989', 'Cary', 1, '127 South Hill');
```

Into Customer_details

```
INSERT into Customer_details(SSN,Customer_email,Customer_name,DOB,Phone_number,Address)
values (334556778, 'tim3@gamil.com', 'Tim Bink', '02-06-1987', '9944435667', 'Street 1,
Wst Blvd City 1');

INSERT into Customer_details(SSN,Customer_email,Customer_name,DOB,Phone_number,Address)
values (334556779, 'Kimb3@gamil.com', 'Kim Bank', '02-09-1987', '9944439667', 'Street 2,
Wst Blvd City 2');

INSERT into Customer_details(SSN,Customer_email,Customer_name,DOB,Phone_number,Address)
values (334556780, 'toml3@gamil.com', 'Tomas link', '02-12-1992', '9944435651','Street
3, Wst Blvd City 3');

INSERT into Customer_details (SSN,Customer_email, Customer_name, DOB, Phone_number,
Address) values (334556785, 'lsys@gamil.com', 'Logan Sys', '02-12-1991',
'9944431151','Street 90, Est Blvd City 3');
```

Into Customer

```
INSERT into Customer(SSN) values ('334556778');
INSERT into Customer(SSN) values ('334556779');
INSERT into Customer(SSN) values ('334556780');
```

Into Staff

```
INSERT into Staff(Hotel_id, Staff_name, Job_title, Department, Age, Phone_number,
Address) values (1, 'John Lint', 'Manager', 'Administration', 24, '9822337766', '124
South St, Raleigh');

INSERT into Staff(Hotel_id, Staff_name, Job_title, Department, Age, Phone_number,
```

```
Address) values (1, 'Thomas King', 'Front Desk', 'Front Desk Rep', 30, '9898987766',
'123 South St, Raleigh');

INSERT into Staff(Hotel_id, Staff_name, Job_title, Department, Age, Phone_number,
Address) values (1, 'Kim Ju', 'Service Staff', 'Catering', 25, '9844557766', '125 South
St, Raleigh');

INSERT into Staff(Hotel_id, Staff_name, Job_title, Department, Age, Phone_number,
Address) values (1, 'Gang Xu', 'Service Staff', 'Laundry', 27, '9888557766', '126 South
St, Raleigh');
```

Into Room

```
INSERT into Room(Room_no,Hotel_id, Availability, Category, Capacity) values (111, 1,
'Available', 'Deluxe', 3);

INSERT into Room(Room_no,Hotel_id, Availability, Category, Capacity)  values (112, 1,
'Available', 'Economy', 4);

INSERT into Room(Room_no,Hotel_id, Availability, Category, Capacity)  values (211, 1,
'Available', 'Deluxe', 3);

INSERT into Room(Room_no,Hotel_id, Availability, Category, Capacity)  values (212, 1,
'Available', 'Presidential', 5);

INSERT into Room(Room_no,Hotel_id, Availability, Category, Capacity)  values (311, 1,
'Available', 'Economy', 2);

INSERT into Room(Room_no,Hotel_id, Availability, Category, Capacity)  values (312, 1,
'Available', 'Executive', 3);
```

Into Reservation

```
INSERT into Reservation( Hotel_id, Checkin_time, Checkout_time) VALUES (1, '2018-01-01
12:00:01', '2018-01-10 11:59:59');

INSERT into Reservation( Hotel_id, Checkin_time, Checkout_time) VALUES ( 1, '2018-01-10
14:00:01', '2018-01-12 19:50:00');

INSERT into Reservation(Hotel_id, Checkin_time, Checkout_time) VALUES (1, '2018-01-15
17:30:00', '2018-01-31 12:30:00');
INSERT into Reservation(Hotel_id, Checkin_time, Checkout_time) VALUES (1, '2018-01-16
17:30:00', '2018-01-31 12:30:00');
INSERT into Reservation(Hotel_id, Checkin_time, Checkout_time) VALUES (1, '2018-01-17
17:30:00', '2018-01-31 12:30:00');
INSERT into Reservation(Hotel_id, Checkin_time, Checkout_time) VALUES (1, '2018-01-17
18:30:00', '2018-01-31 12:30:00');
```

Into Billing_info

```
INSERT into Billing_info(Amount) values (0);
```

```
INSERT into Billing_info(Amount) values (0);

INSERT into Billing_info(Amount) values (0);

INSERT into Billing_info(Amount) values (0);

INSERT into Billing_info(Amount) values (0);

INSERT into Billing_info(Amount) values (0);
```

Into Check_in_info

```
INSERT into Check_in_info(Hotel_id, Staff_id, Customer_id, Room_no,
Reservation_id,Billing_id) values (1, 2, 1, 211, 1, 2);

INSERT into Check_in_info(Hotel_id, Staff_id, Customer_id, Room_no,
Reservation_id,Billing_id) values (1, 2, 2, 312, 2, 1);

INSERT into Check_in_info(Hotel_id, Staff_id, Customer_id, Room_no,
Reservation_id,Billing_id) values (1, 2, 3, 112, 3, 3);
```

Into Services

```
INSERT into Services(Service_name, Service_price) values ('Laundry', 10);

INSERT into Services(Service_name, Service_price) values ('Wifi', 20);

INSERT into Services(Service_name, Service_price) values ('Special Service', 20);

INSERT into Services(Service_name, Service_price) values ('Catering', 40);
```

Into Services_Used

```
INSERT into Services_used(Hotel_Id,Staff_Id,Service_Id,Reservation_Id) values(1,3,1,1);

INSERT into Services_used(Hotel_Id,Staff_Id,Service_Id,Reservation_Id) values(1,4,3,1);

INSERT into Services_used(Hotel_Id,Staff_Id,Service_Id,Reservation_Id) values(1,3,2,1);

INSERT into Services_used(Hotel_Id,Staff_Id,Service_Id,Reservation_Id) values(1,3,1,2);

INSERT into Services_used(Hotel_Id,Staff_Id,Service_Id,Reservation_Id) values(1,4,3,3);

INSERT into Services_used(Hotel_Id,Staff_Id,Service_Id,Reservation_Id) values(1,3,2,3);
```

Into Payment_details

```
INSERT into Payment_details(Payment_type,Billing_address,Ccc_details) values ('Card',
'123 Wr St Hill', '23343438787585');

INSERT into Payment_details(Payment_type,Billing_address,Ccc_details) values ('Card',
'121 Wr St Hill', '2334343870000');
```

```
INSERT into Payment_details (Payment_type,Billing_address,Ccc_details) values ('Cheque',
'123 South Bnk Hill', '23343433245235');
```

Into Payment_relation

```
INSERT into Payment_relation(Payment_Id, Billing_Id) values (1,3);
INSERT into Payment_relation(Payment_Id, Billing_Id) values (2,2);
INSERT into Payment_relation(Payment_Id, Billing_Id) values (3,1);
```

Into Executive_manager

```
INSERT into Executive_manager values ('Executive_Manager_Default');
```

Into Rooms_price_listing

```
INSERT into Rooms_price_listing values ('Raleigh','Economy',50);
INSERT into Rooms_price_listing values ('Raleigh','Deluxe',100);
INSERT into Rooms_price_listing values ('Raleigh','Executive',150);
INSERT into Rooms_price_listing values ('Raleigh','Presidential',200);

INSERT into Rooms_price_listing values ('Cary','Economy',40);
INSERT into Rooms_price_listing values ('Cary','Deluxe',80);
INSERT into Rooms_price_listing values ('Cary','Executive',120);
INSERT into Rooms_price_listing values ('Cary','Presidential',160);
```

Into Manager_hotel_association

```
INSERT into Manager_hotel_association values ('Executive_manager_Default',1);
INSERT into Manager_hotel_association values ('Executive_manager_Default',2);
INSERT into Manager_hotel_association values ('Executive_manager_Default',3);
INSERT into Manager_hotel_association values ('Executive_manager_Default',4);
```

Selects

Note: All result sets from select statements have been formatted for readability.

```
SQL> Select * from Hotel;
| Hotel_id | Hotel_name      | Hotel_phone_no | City    |
Manager_Staff_id | Street_name    |
+----------+-----------------+----------------+---------+-------------
----+----------------+
|        1 | Wolfinn Raleigh | 9090908989     | Raleigh |
1 | 134 West Blvd  |
|        2 | Wolfinn Cary    | 9090908989     | Cary    |
1 | 122 East Hill  |
|        3 | Wolfinn Cary 1  | 9090908765     | Cary    |
1 | 123 North Hill |
|        4 | Wolfinn Cary 2  | 9090111989     | Cary    |
1 | 127 South Hill |
```

```
+----------+----------------+---------------+--------+-------------
----+----------------+
4 rows in set (0.00 sec)

SQL> Select * from Room;
+---------+----------+--------------+--------------+----------+
| Room_no | Hotel_id | Availability | Category     | Capacity |
+---------+----------+--------------+--------------+----------+
|     111 |        1 | Available    | Deluxe       |        3 |
|     112 |        1 | Available    | Economy      |        4 |
|     211 |        1 | Available    | Deluxe       |        3 |
|     212 |        1 | Available    | Presidential |        5 |
|     311 |        1 | Available    | Economy      |        2 |
|     312 |        1 | Available    | Executive    |        3 |
+---------+----------+--------------+--------------+----------+
6 rows in set (0.01 sec)

SQL> Select * from Staff;
| Staff_id | Hotel_id | Staff_name  | Job_title     | Department     |
Age | Phone_number | Address               |
+----------+----------+-------------+---------------+----------------+-
----+--------------+-----------------------+
|        1 |        1 | John Lint   | Manager       | Administration |
24  | 9822337766   | 124 South St, Raleigh |
|        2 |        1 | Thomas King | Front Desk    | Front Desk Rep |
30  | 9898987766   | 123 South St, Raleigh |
|        3 |        1 | Kim Ju      | Service Staff | Catering       |
25  | 9844557766   | 125 South St, Raleigh |
|        4 |        1 | Gang Xu     | Service Staff | Laundry        |
27  | 9888557766   | 126 South St, Raleigh |
+----------+----------+-------------+---------------+----------------+-
----+--------------+-----------------------+
4 rows in set (0.00 sec)

SQL> Select * from Customer;
+-------------+-----------+
| Customer_id | SSN       |
+-------------+-----------+
|           1 | 334556778 |
|           2 | 334556779 |
|           3 | 334556780 |
+-------------+-----------+
3 rows in set (0.00 sec)

SQL> Select * from Customer_details;
```

```
| 334556778 | tim3@gamil.com  | Tim Bink      | 02-06-1987 | 9944435667
| Street 1, Wst Blvd City 1  |
| 334556779 | Kimb3@gamil.com | Kim Bank      | 02-09-1987 | 9944439667
| Street 2, Wst Blvd City 2  |
| 334556780 | toml3@gamil.com | Tomas link    | 02-12-1992 | 9944435651
| Street 3, Wst Blvd City 3  |
| 334556785 | lsys@gamil.com  | Logan Sys     | 02-12-1991 | 9944431151
| Street 90, Est Blvd City 3 |
+-----------+-----------------+---------------+------------+-----------
---+----------------------------+
4 rows in set (0.00 sec)

SQL> Select * from Reservation;
+----------------+----------+---------------------+--------------------
-+
| Reservation_id | Hotel_id | Checkin_time        | Checkout_time
|
+----------------+----------+---------------------+--------------------
-+
|              1 |        1 | 2018-01-01 12:00:01 | 2018-01-10 11:59:59
|
|              2 |        1 | 2018-01-10 14:00:01 | 2018-01-12 19:50:00
|
|              3 |        1 | 2018-01-15 17:30:00 | 2018-01-31 12:30:00
|
|              4 |        1 | 2018-01-16 17:30:00 | 2018-01-31 12:30:00
|
|              5 |        1 | 2018-01-17 17:30:00 | 2018-01-31 12:30:00
|
|              6 |        1 | 2018-01-17 18:30:00 | 2018-01-31 12:30:00
|
+----------------+----------+---------------------+--------------------
-+
6 rows in set (0.00 sec)

SQL> Select * from Check_in_info;
+----------+----------+-------------+---------+----------------+-------
-----+
| Hotel_id | Staff_id | Customer_id | Room_no | Reservation_id |
Billing_id |
+----------+----------+-------------+---------+----------------+-------
-----+
|        1 |        2 |           3 |     112 |              3 |
3 |
|        1 |        2 |           1 |     211 |              1 |
```

```
2 |
|        1 |         2 |           2 |     312 |               2 |
1 |
+----------+----------+-------------+---------+-----------------+-------
-----+
3 rows in set (0.00 sec)
SQL> Select * from Services;
+------------+-----------------+---------------+
| Service_id | Service_name    | Service_price |
+------------+-----------------+---------------+
|          1 | Laundry         |         10.00 |
|          2 | Wifi            |         20.00 |
|          3 | Special Service |         20.00 |
|          4 | Catering        |         40.00 |
+------------+-----------------+---------------+
4 rows in set (0.00 sec)
SQL> Select * from Services_used;
+------------+-----------------+---------------+
| Service_id | Service_name    | Service_price |
+------------+-----------------+---------------+
|          1 | Laundry         |         10.00 |
|          2 | Wifi            |         20.00 |
|          3 | Special Service |         20.00 |
|          4 | Catering        |         40.00 |
+------------+-----------------+---------------+
4 rows in set (0.00 sec)
SQL> Select * from Payment_details;
+------------+--------------+--------------------+----------------+
| Payment_id | Payment_type | Billing_address    | Ccc_details    |
+------------+--------------+--------------------+----------------+
|          1 | Card         | 123 Wr St Hill     | 23343438787585 |
|          2 | Card         | 121 Wr St Hill     | 2334343870000  |
|          3 | Cheque       | 123 South Bnk Hill | 23343433245235 |
+------------+--------------+--------------------+----------------+
3 rows in set (0.00 sec)
SQL> Select * from Payment_relation;
+------------+------------+
| Payment_id | Billing_id |
+------------+------------+
|          1 |          3 |
|          2 |          2 |
|          3 |          1 |
+------------+------------+
3 rows in set (0.00 sec)
SQL> Select * from Billing_info;
```

```
+------------+--------+
| Billing_id | Amount |
+------------+--------+
|          1 |      0 |
|          2 |      0 |
|          3 |      0 |
|          4 |      0 |
|          5 |      0 |
|          6 |      0 |
+------------+--------+
6 rows in set (0.01 sec)
SQL> Select * from Rooms_price_listing;
+---------+--------------+-------+
| City    | Category     | Price |
+---------+--------------+-------+
| Cary    | Deluxe       |    80 |
| Cary    | Economy      |    40 |
| Cary    | Executive    |   120 |
| Cary    | Presidential |   160 |
| Raleigh | Deluxe       |   100 |
| Raleigh | Economy      |    50 |
| Raleigh | Executive    |   150 |
| Raleigh | Presidential |   200 |
+---------+--------------+-------+
8 rows in set (0.00 sec)
SQL> Select * from Executive_manager;
+----------------------+
| Manager_name         |
+----------------------+
| Executive_Manager_De |
+----------------------+
1 row in set (0.01 sec)
SQL> Select * from Manager_hotel_association;
+----------------------+----------+
| Manager_name         | Hotel_id |
+----------------------+----------+
| Executive_manager_De |        1 |
| Executive_manager_De |        2 |
| Executive_manager_De |        3 |
| Executive_manager_De |        4 |
+----------------------+----------+
4 rows in set (0.00 sec)
```

4) Interactive SQL Queries

Note: All result sets from select statements have been formatted for readability.

4.1 Queries for tasks and operations

1. Information Processing

Enter Hotel information

```
SQL> INSERT into Hotel(Hotel_name, Hotel_phone_no, City,
Manager_Staff_id, Street_name) values ('Wolfinn 2
Raleigh','9090908945','Raleigh', 1, '134 West Blvd');

Query OK, 1 row affected (0.00 sec)
```

Update information about hotel

```
SQL> Update Hotel set Hotel_Name='Maria warrior Hotel', city='Durham',
street_Name='104 ligon drive', hotel_phone_no='9193847654',
manager_Staff_id=1 where hotel_Id=3;

Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete information about hotel

```
SQL> Delete from Hotel where Hotel_id=2;

Query OK, 1 row affected (0.00 sec)
```

Enter information about room

```
SQL> INSERT into Room(Room_no,Hotel_id, Availability, Category,
Capacity) values (113, 1, 'Available', 'Deluxe', 2);

Query OK, 1 row affected (0.00 sec)
```

Update information about room

```
SQL> update Room set capacity=3, category='Executive' where hotel_Id=1
and room_no =113;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete information about room

```
SQL> delete from Room where hotel_Id=1 and room_No=113;

Query OK, 1 row affected (0.00 sec)
```

Enter information about Staff

```
SQL> INSERT into Staff(Hotel_id, Staff_name, Job_title, Department,
Age, Phone_number, Address) values (1, 'Jathin Lint', 'Service Staff',
'Dry Cleaning', 46, '9822337766', '125 hypothetical St, Raleigh');

Query OK, 1 row affected (0.00 sec)
```

Update information about Staff

```
SQL> UPDATE Staff set staff_name = 'Jay Shetty', age =46 where
hotel_id=1 and staff_id=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete information about Staff

```
SQL> delete from Staff where staff_id = 4 and Hotel_Id=1;

Query OK, 1 row affected (0.00 sec)
```

Enter information about Customer

```
SQL> INSERT into Customer_details (SSN, Customer_email, Customer_name,
DOB, Phone_number, Address) values (339956781, 'jred3@gmail.com', 'Jill
Red', '12-06-1982', '9944995667', 'Street 5, Wst Blvd City 4');

Query OK, 1 row affected (0.01 sec)

SQL>INSERT into Customer(SSN) values ('339956781');

Query OK, 1 row affected (0.01 sec)
```

Update information about Customer

```
SQL> UPDATE Customer_details set customer_email = 'njred@gmail.com'
where SSN='339956781';

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete information about Customer

```
SQL> delete from Customer where customer_id = '4';
```

```
Query OK, 1 row affected (0.17 sec)
```

Check Rooms availability

```
SQL> SELECT * from Room where availability='Available' AND
category='Deluxe' AND Hotel_Id=1;

+---------+----------+--------------+----------+----------+
| Room_no | Hotel_id | Availability | Category | Capacity |
+---------+----------+--------------+----------+----------+
|    111  |    1     | Available    | Deluxe   |       3  |
|    211  |    1     | Available    | Deluxe   |       3  |
+---------+----------+--------------+----------+----------+
2 rows in set (0.00 sec)
```

Assign Rooms to customer according to their request and availability

```
SQL> UPDATE Room SET Availability='Occupied' where room_no=112 and
hotel_id=1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

SQL> INSERT into Reservation(Hotel_id, Checkin_time, Checkout_time)
values (1,'2018-02-16 12:30:00', '2018-03-16 14:20:00');

Query OK, 1 row affected (0.00 sec)

SQL> INSERT into Billing_info(Amount) values (0);

Query OK, 1 row affected (0.01 sec)


SQL> INSERT into Check_in_info(Hotel_id, Staff_id, Customer_id,
Room_no, Reservation_id,Billing_id) values (1,2,3,112,4,4);

Query OK, 1 row affected (0.01 sec)

SQL> SELECT * from Reservation where reservation_id=4;

+----------------+----------+---------------------+---------------------
-+
| Reservation_id | Hotel_id | Checkin_time        | Checkout_time     |
+----------------+----------+---------------------+---------------------
```

```
-+
|              4 |       1 | 2018-02-16 12:30:00  | 2018-03-16 14:20:00
|
+----------------+---------+----------------------+--------------------
-+
1 row in set (0.00 sec)
```

Release rooms

```
SQL> UPDATE Room SET availability='Available' where hotel_Id=1 and
room_no =112;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Enter Information about Services

```
SQL> INSERT into Services(Service_name, Service_price) values ('Gym',
15);
Query OK, 1 row affected (0.01 sec)
```

Enter Information about Payment Details

```
SQL> INSERT into Payment_details(Payment_type, Billing_address,
Ccc_details) values ('Card', '311 Crest St Rd 3l', '2994343870000');

Query OK, 1 row affected (0.01 sec)


SQL> INSERT into Payment_relation(Payment_id, Billing_id) values(4,4);

Query OK, 1 row affected (0.01 sec)
```

2. Maintaining Service Records

Enter service records for services availed by customers

```
SQL> INSERT into Services_used(Hotel_Id, Staff_Id, Service_Id,
Reservation_Id) values(1,3,1,2);

Query OK, 1 row affected (0.01 sec)
```

Update service records for services availed by customers

```
SQL> UPDATE Services_used set Service_Id=2 where Service_Instance_Id=7;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

3. *Maintaining Billing Accounts*

Generate Bills

i) Firstly we determine payment type :

```
SQL> select payment_type from Payment_details pd join Payment_relation
pr on pd.Payment_id=pr.Payment_id join Check_in_info cii on
cii.Billing_id=pr.Billing_id;
```

ii)
For non-card payment options ->

```
SQL> select final.* from (select 'Total_payable_amount' as 'Itemized
list',sum(ab.Amount_charged) as 'Amount_charged' from((select 'Room
charges' as 'Itemized list'
,((checkout_time-checkin_time)/1000000)*price as 'Amount_charged' from
Reservation r join Check_in_info cii on
r.reservation_id=cii.reservation_id join Room ro on
cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join
Rooms_price_listing rpl on h.city=rpl.city and rpl.category=ro.category
and r.reservation_id=1)
union (select service_name as 'Itemized list', sum(service_price) as
'Amount_charged' from Services s join Services_used su on
s.service_id=su.service_id join Reservation r on
r.reservation_id=su.reservation_id where r.reservation_id=1 group by
service_name)) ab
union (select 'Room charges' as 'Itemized list'
,((checkout_time-checkin_time)/1000000)*price as 'Amount_charged' from
Reservation r join Check_in_info cii on
r.reservation_id=cii.reservation_id join Room ro on
cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join
Rooms_price_listing rpl on h.city=rpl.city and rpl.category=ro.category
and r.reservation_id=1)
union (select service_name as 'Itemized list', sum(service_price) as
'Amount_charged' from Services s join Services_used su on
s.service_id=su.service_id join Reservation r on
r.reservation_id=su.reservation_id where r.reservation_id=1 group by
service_name)) final order by final.Amount_charged;
```

```
+----------------------+----------------+
| Itemized list        | Amount_charged |
+----------------------  +----------------+
| Laundry              |         10.0000 |
| Wifi                 |         20.0000 |
| Room charges         |        899.5958 |
| Total_payable_amount |        929.5958 |
+----------------------+----------------+
4 rows in set (0.00 sec)
```

For card payment options->

```
SQL> select final.* from (select 'Total_payable_amount' as
'Itemized_list',sum(ab.Amount_charged) as 'Amount_charged' from((select
'Room_charges' as 'Itemized_list'
,((checkout_time-checkin_time)/1000000)*price as 'Amount_charged' from
Reservation r join Check_in_info cii on
r.reservation_id=cii.reservation_id join Room ro on
cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join
Rooms_price_listing rpl on h.city=rpl.city and rpl.category=ro.category
and r.reservation_id=1)
union (select service_name as 'Itemized_list', sum(service_price) as
'Amount_charged' from Services s join Services_used su on
s.service_id=su.service_id join Reservation r on
r.reservation_id=su.reservation_id where r.reservation_id=1 group by
service_name) union (select 'Discount' as 'Itemized_list'
,((checkout_time-checkin_time)/1000000)*price*(-0.05) as
'Amount_charged' from Reservation r join Check_in_info cii on
r.reservation_id=cii.reservation_id join Room ro on
cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join
Rooms_price_listing rpl on h.city=rpl.city and rpl.category=ro.category
and r.reservation_id=1)) ab
union (select 'Room_charges' as 'Itemized_list'
,((checkout_time-checkin_time)/1000000)*price as 'Amount_charged' from
Reservation r join Check_in_info cii on
r.reservation_id=cii.reservation_id join Room ro on
cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join
Rooms_price_listing rpl on h.city=rpl.city and rpl.category=ro.category
and r.reservation_id=1)
union (select service_name as 'Itemized_list', sum(service_price) as
'Amount_charged' from Services s join Services_used su on
s.service_id=su.service_id join Reservation r on
r.reservation_id=su.reservation_id where r.reservation_id=1 group by
```

```
service_name) union (select 'Discount' as 'Itemized_list'
,((checkout_time-checkin_time)/1000000)*price*(-0.05) as
'Amount_charged' from Reservation r join Check_in_info cii on
r.reservation_id=cii.reservation_id join Room ro on
cii.room_no=ro.room_no join Hotel h on h.hotel_id=cii.hotel_id join
Rooms_price_listing rpl on h.city=rpl.city and rpl.category=ro.category
and r.reservation_id=1) ) final order by CASE When
final.Itemized_list='Laundry' then '1'
 When final.Itemized_list='Laundry' then 1
 When final.Itemized_list='Wifi' then 2
 When final.Itemized_list='Catering' then 3
 When final.Itemized_list='Gym' then 4
 When final.Itemized_list='Room_charges' then 5
 When final.Itemized_list='Discount' then 6
 When final.Itemized_list='Total_payable_amount' then 7
 Else final.Amount_charged
  END;
```

```
+----------------------+----------------+
| Itemized_list      | Amount_charged |
+----------------------+----------------+
| Laundry            |    10.000000      |
| Wifi               |    20.000000      |
| Room_charges       |    899.595800     |
| Discount           |    -44.979790     |
| Total_payable_amount |      884.616010 |
+----------------------+----------------+
5 rows in set (0.00 sec)
```

4. Reports

Occupancy Report by Hotel

```
SQL> select h1.Hotel_id,hotel_name,count(*) as
Total_occupancy,(count(*)*100/(select count(*) from Room r join Hotel h
on r.hotel_id=h.hotel_id where h.Hotel_id = h1.hotel_id)) as
Percent_occupied from Room r join Hotel h1 on r.hotel_id=h1.hotel_id
where Availability='Occupied' group by h1.Hotel_id;
```

```
+----------+-----------------+-----------------+------------------+
| Hotel_id | hotel_name           | Total_occupancy | Percent_occupied
|
+----------+-----------------+-----------------+------------------+
```

```
|         1| Wolfinn Raleigh |                3 |       50.0000    |
+----------+-----------------+------------------+------------------+
1 row in set (0.00 sec)
```

Occupancy Report by Room Type

```
SQL> select category as Room_type, count(*) as 'Occupancy',
(count(*)*100/(select count(*) from Room r2 where
r2.category=r1.category)) as 'Percent occupied' from Room r1 where
availability='Occupied' group by category;

(Some tuples have been modified for obtaining the results below)

+--------------+-----------+------------------+
| Room_type    | Occupancy | Percent occupied |
+--------------+-----------+------------------+
| Deluxe       |         2 |         100.0000 |
| Economy      |         1 |          50.0000 |
| Executive    |         1 |         100.0000 |
| Presidential |         1 |         100.0000 |
+--------------+-----------+------------------+
4 rows in set (0.01 sec)
```

Occupancy Report by Date Range

```
SQL> SELECT '2018-01-01 12:00:01' as 'Start_date','2018-01-31 12:30:00'
as 'End_date', count(*) AS 'Occupancy', (count(*)*100/(SELECT count(*)
FROM (SELECT distinct r1.room_no, r1.hotel_id FROM Room r1) as t2)) as
'Percent occupied' FROM
(SELECT distinct c.room_no, c.hotel_id FROM Check_in_info c INNER JOIN
(SELECT reservation_id FROM Reservation WHERE (checkin_time>
'2018-01-01 12:00:01' AND checkin_time<'2018-01-31 12:30:00') OR
(checkout_time> '2018-01-01 12:00:01'AND checkout_time <'2018-01-31
12:30:00')  OR (checkin_time<'2018-01-01 12:00:01' AND
checkout_time>'2018-01-31 12:30:00')) AS d
WHERE d.reservation_id = c.reservation_id) AS tab1;
+---------------------+---------------------+-----------+--------------
----+
| Start_date          | End_date            | Occupancy | Percent
occupied |
+---------------------+---------------------+-----------+--------------
----+
| 2018-01-01 12:00:01 | 2018-01-31 12:30:00 |         2 | 33.3333    |
+---------------------+---------------------+-----------+--------------
----+
```

```
1 row in set (0.00 sec)
```

Occupancy Report by City

```
SQL> select city, count(*) as 'Occupancy',(count(city)*100/
(select count(*) from Room r1 JOIN Hotel h1 on r1.hotel_id =
h1.hotel_id where h1.city = h.City) ) as 'Percent Occupied' from Room r
join Hotel h on r.Hotel_id = h.Hotel_id where Availability='Occupied'
group by city;

+---------+-----------+------------------+
| city    | Occupancy | Percent Occupied |
+---------+-----------+------------------+
| Cary    |         1 |         100.0000 |
| Raleigh |         5 |          83.3333 |
+---------+-----------+------------------+
2 rows in set (0.00 sec)
```

Information on staff grouped by their role

```
SQL>  select * from Staff s where s.hotel_id=1 order by job_title;

+----------+----------+-------------+---------------+----------------+-
----+--------------+-----------------------+
| Staff_id | Hotel_id | Staff_name  | Job_title     | Department     |
Age | Phone_number | Address               |
+----------+----------+-------------+---------------+----------------+-
----+--------------+-----------------------+
|        2 |        1 | Thomas King | Front Desk    | Front Desk Rep |
30  | 9898987766   | 123 South St, Raleigh |
|        1 |        1 | John Lint   | Manager       | Administration |
24  | 9822337766   | 124 South St, Raleigh |
|        3 |        1 | Kim Ju      | Service Staff | Catering       |
25  | 9844557766   | 125 South St, Raleigh |
|        4 |        1 | Gang Xu     | Service Staff | Laundry        |
27  | 9888557766   | 126 South St, Raleigh |
+----------+----------+-------------+---------------+----------------+-
----+--------------+-----------------------+
4 rows in set (0.00 sec)
```

Information on all staff members serving the customer during the stay

```
SQL>  (select Staff_name,Job_title,s.Staff_id,address from Reservation
r join Services_used su on r.reservation_id=su.reservation_id join
Staff s on su.staff_id=s.staff_id where r.reservation_id=1) union
```

```
(select Staff_name,Job_title,s.staff_id,address from Reservation r join
Check_in_info cii on r.reservation_id=cii.reservation_id join Staff s
on s.staff_id=cii.staff_id where r.reservation_id=1);

+-------------+--------------+----------+----------------------+
| Staff_name  | Job_title    | Staff_id | address              |
+-------------+--------------+----------+----------------------+
| Kim Ju      | Service Staff |        3 | 125 South St, Raleigh |
| Gang Xu     | Service Staff |        4 | 126 South St, Raleigh |
| Thomas King | Front Desk    |        2 | 123 South St, Raleigh |
+-------------+--------------+----------+----------------------+
3 rows in set (0.00 sec)
```

Revenue report for a Hotel in a date range

```
SQL> select '2018-01-01 12:00:01' as 'Start date', '2018-01-31
12:30:00' as 'End date',cii.Hotel_id, sum(amount) as Revenue from
Billing_info bi join Check_in_info cii on bi.billing_id=cii.billing_id
join Reservation r on r.reservation_id = cii.reservation_id where
cii.hotel_id=1 and ((r.checkin_time> '2018-01-01 12:00:01' AND
r.checkin_time<'2018-01-31 12:30:00') OR (r.checkout_time> '2018-01-01
12:00:01' AND r.checkout_time <'2018-01-31 12:30:00')  OR
(r.checkin_time<'2018-01-01 12:00:01' AND r.checkout_time>'2018-01-31
12:30:00'));

+---------------------+---------------------+----------+---------+
| Start date          | End date            | Hotel_id | Revenue |
+---------------------+---------------------+----------+---------+
| 2018-01-01 12:00:01 | 2018-01-31 12:30:00 |        1 |  338 |
+---------------------+---------------------+----------+---------+
1 row in set (0.00 sec)
```

4.2 Explain directive for two queries

Let's consider the report occupancy by room type query.

```
SQL> Explain select category as Room_type, count(*) as 'Occupancy',
(count(*)*100/(select count(*) from Room r2 where
r2.category=r1.category)) as 'Percent occupied' from Room r1 where
availability='Occupied' group by category;
+------+--------------------+------+------+--------------+------+----
-----+------+------+-------------------------------------------------+
```

```
| id   | select_type        | table | type | possible_keys | key  |
key_len | ref  | rows | Extra                                          |
+------+--------------------+-------+------+---------------+------+----
-----+------+------+------------------------------------------------+
|    1 | PRIMARY            | r1    | ALL  | NULL          | NULL |
NULL    | NULL |    7 | Using where; Using temporary; Using filesort |
|    2 | DEPENDENT SUBQUERY | r2    | ALL  | NULL          | NULL |
NULL    | NULL |    7 | Using where                                    |
+------+--------------------+-------+------+---------------+------+----
-----+------+------+------------------------------------------------+
2 rows in set (0.01 sec)

SQL> Create index room_cat on Room(category);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

SQL> Explain select category as Room_type, count(*) as 'Occupancy',
(count(*)*100/(select count(*) from Room r2 where
r2.category=r1.category)) as 'Percent occupied' from Room r1 where
availability='Occupied' group by category;
+------+--------------------+-------+-------+---------------+---------
+---------+------+------+-------------+
| id   | select_type        | table | type  | possible_keys | key
| key_len | ref  | rows | Extra       |
+------+--------------------+-------+-------+---------------+---------
+---------+------+------+-------------+
|    1 | PRIMARY            | r1    | index | NULL          | room_cat
| 17      | NULL |    7 | Using where |
|    2 | DEPENDENT SUBQUERY | r2    | ref   | room_cat      | room_cat
| 17      | func |    1 | Using index |
+------+--------------------+-------+-------+---------------+---------
+---------+------+------+-------------+
2 rows in set (0.00 sec)
```

Let's consider the query checking whether rooms of a particular category are available.

```
SQL> Explain select * from Room where Category="Economy" and
Hotel_id=1;
+------+-------------+-------+------+---------------+------+
---------+------+------+-------------+
| id   | select_type | table | type | possible_keys | key  |
key_len | ref  | rows | Extra       |
+------+-------------+-------+------+---------------+------+
---------+------+------+-------------+
```

```
|    1 | SIMPLE      | Room  | ALL  | room_fk       | NULL |
NULL    | NULL |    7 | Using where |
+------+-------------+-------+------+---------------+------+
--------+------+------+-------------+
1 row in set (0.00 sec)
```

Creating an index on category will help us filter out the rooms faster as we don't have to go through all the tuples comparing the values.

```
SQL> create Index roomcat on Room(Category);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Let's execute the previous query again and observe the results

```
SQL> explain select * from Room where Category="Economy" and
Hotel_id=1;
+------+-------------+-------+------+-----------------+-----
----+---------+-------+------+----------------------+
| id   | select_type | table | type | possible_keys   | key
| key_len | ref   | rows | Extra                  |
+------+-------------+-------+------+-----------------+-----
----+---------+-------+------+----------------------+
|    1 | SIMPLE      | Room  | ref  | room_fk,roomcat |
roomcat | 17      | const |    2 | Using index condition |
+------+-------------+-------+------+-----------------+-----
----+---------+-------+------+----------------------+
1 row in set (0.00 sec)
```

Now, instead of checking all the 7 tuples as there is an index on Category, only the tuples belonging to the particular category i.e 2 tuples of "Economy" category are checked/filtered using the index.

4.3 Query correctness proofs

1. Occupancy Report by Hotel

```
select h1.Hotel_id,hotel_name,count(*) as
Total_occupancy,(count(*)*100/(select count(*) from Room r join
Hotel h on r.hotel_id=h.hotel_id where h.Hotel_id = h1.hotel_id))
as Percent_occupied from Room r join Hotel h1 on
```

```
r.hotel_id=h1.hotel_id where Availability='Occupied' group by
h1.Hotel_id;
```

a) $\gamma_{h1.Hotel\_id,hotel\_name,Count(*) \rightarrow Total\_Occupancy,(\gamma_{((Count(*)*100)) \div (\gamma_{Count(*)}((\sigma_{h.Hotel\_id=h1.Hotel\_id})((\rho_{r(Hotel\_id)}(Room) \triangleright\triangleleft (\rho_{h(Hotel\_id)}(Hotel))_{)\ ))\rightarrow Percent\ Occupied}(\sigma_{Availability='Occupied'}(\rho_{r(Hotel\_id)}(Room) \triangleright\triangleleft \rho_{h1(Hotel\_id)}(Hotel)))$

b) Let's say h1 is any tuple in the Hotel relation ,r is any tuple in the Room relation,Percent Occupied is any tuple in the nested subquery involving result of division of count of occupied rooms with the tuple h1 of Hotel and tuple r of Room total rooms available with r1.hotel_id has same value as h1.hotel_id with records filtered out for same value of Hotel in hotels and Rooms, such that the value of r.hotel_id is same as h.hotel_id .gives us the list of rooms joined to the hotel tuples to which they belong.Each tuple of (h1,r,Percent_Occupied(h1,r)) gives the information about hotels,counts of rooms occupied and percentage of rooms occupied for that hotel. For each such combination (h,r,Percent_Occupied(h1,r1)) the query returns the value of Hotel_id(the id of Hotel),Count(*)(the total number of occupied rooms in particular Hotel) ,Percentage_Occupied(percentage of rooms occupied in that hotel or number of rooms occupied divided by total number of available (occupied and available) rooms for the particular Hotel) . By calculating percentage_occupied,the subquery does something similar, it does the join but doesn't discard room tuples which are occupied. It then filters out the tuples by the Hotel_id returned in the outer query, i.e. for each Hotel_id of the outer tuple, the inner query calculates the total number of available (occupied and available) rooms for the particular Hotel_id. Let's say h1 is any tuple in the hotel relation, r1 is any tuple in the room relation such that r1.hotel_id=h1.hotel_id and h1.city=h.city gives us the list of all room tuples which belong to hotels residing in the same Hotel as h.
By filtering the tuples for the value of Availability,we limit the Hotel_id and corresponding count of rooms to the rooms which are occupied. But this is exactly what our query should return; see the specification.

2. Occupancy Report by City

$\gamma_{City,Count(*) \rightarrow Occupancy,(\gamma_{(((Count(City)*100)) \div (\gamma_{Count(*)}((\sigma_{h1.City=h.City})((\rho_{r1(Hotel\_id)}(Room) \triangleright\triangleleft (\rho_{h1(Hotel\_id,City)}(Hotel))_{)\ ))\rightarrow Percent\ Occupied}(\sigma_{Availability='Occupied'}(\rho_{r(Hotel\_id)}(Room) \triangleright\triangleleft \rho_{h(Hotel\_id)}(Hotel)))$

```
select city, count(*) as 'Occupancy',(count(city)*100/
```

```
(select count(*) from Room r1 JOIN Hotel h1 on r1.hotel_id =
h1.hotel_id where h1.city = h.City) ) as 'Percent Occupied' from Room r
join Hotel h on r.Hotel_id = h.Hotel_id where Availability='Occupied'
group by city;
```

c) Let's say h is any tuple in the Hotel relation ,r is any tuple in the Room relation,Percent Occupied is any tuple in the nested subquery involving result of division of count of occupied rooms with the tuple h1 of Hotel and tuple r1 of Room total rooms available with r1.hotel_id has same value as h1.hotel_id with records filtered out for same value of city in hotels and Rooms, such that the value of r.hotel_id is same as h.hotel_id .gives us the list of rooms joined to the hotel tuples to which they belong.Each tuple of (h,r,Percent_Occupied(h1,r1)) gives the information about cities,counts of rooms occupied and percentage of rooms occupied for that city. For each such combination (h,r,Percent_Occupied(h1,r1)) the query returns the value of City(the name of city),Count(*)(the total number of occupied rooms in particular city) ,Percentage_Occupied(percentage of rooms occupied in that city or number of rooms occupied divided by total number of available (occupied and available) rooms for the particular city) . By calculating percentage_occupied,the subquery does something similar, it does the join but doesn't discard room tuples which are occupied. It then filters out the tuples by the city returned in the outer query, i.e. for each city of the outer tuple, the inner query calculates the total number of available (occupied and available) rooms for the particular city. Let's say h1 is any tuple in the hotel relation, r1 is any tuple in the room relation such that r1.hotel_id=h1.hotel_id and h1.city=h.city gives us the list of all room tuples which belong to hotels residing in the same city as h.
By filtering the tuples for the value of Availability,we limit the city names and count of rooms to the rooms which are occupied. But this is exactly what our query should return; see the specification.