**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(Manager\_Name) -**

Manager\_Name -> Manager\_Name

This relation is in BCNF (and thus 3NF) because it contains one attribute which is the key.

**Hotel(Hotel\_ID, 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(Staff\_ID, Hotel\_ID,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, Service\_ID)**

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, Service\_Instance\_Id, 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.

**Rooms\_Price\_Listing(City, Category, 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,

CONSTRAINT checkin\_Reservation\_fk FOREIGN KEY (Reservation\_id) references Reservation(Reservation\_id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT checkin\_billing\_fk FOREIGN KEY (Billing\_id) references Billing\_info(Billing\_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](mailto: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, 'Kimb[3@gamil.com](mailto:tim3@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, 'to[ml3@gamil.com](mailto:tim3@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](mailto:tim3@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, 'jred[3@gmail.com](mailto:tim3@gamil.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](mailto: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;

1. γh1.Hotel\_id,hotel\_name,Count(\*) →Total\_Occupancy,(γ((Count(\*)\*100)) ÷ (γCount(\*)((σh.Hotel\_id=h1.Hotel\_id)((ρr(Hotel\_id)(Room)▷◁ (ρh(Hotel\_id)(Hotel))) ))→Percent Occupied(σAvailability=’Occupied’(ρr(Hotel\_id)(Room) ▷◁ ρh1(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

γCity,Count(\*) →Occupancy,(γ((Count(City)\*100)) ÷ (γCount(\*)((σh1.City=h.City)((ρr1(Hotel\_id)(Room)▷◁ (ρh1(Hotel\_id,City)(Hotel))) ))→Percent Occupied(σAvailability=’Occupied’(ρr(Hotel\_id)(Room) ▷◁ ρh(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.