

Hotel Management System

For Wolf Inns, a popular Hotel Chain

CSC 540 Database Systems

Project Report #2

Srilatha Bekkem

1) Database Schema

Checks for identifying BCNF :-

1. Each of the functional dependencies, the closure of the LHS is a superkey
2. A two attribute relation is always in BCNF

Note: Any relation that is in BCNF, has to be in 3NF.

Below are the schemas for the entire database system: -

1. location(city, costfactor);
2. hotel(hotel_id, Name, Phone_no, Address, City);
3. staff(staff_id, Name, Phone_no, Title, Pwd, Dob, Hotel_id);
4. catering_staff(staff_id);
5. service_staff(staff_id);
6. occupancy(max_limit);
7. category(Category_name);
8. room(room_no, hotel_id, Availability, max_limit, category_name, rate)
9. presidential(room_no, hotel_id, catering_staff_id, service_staff_id)
10. card(card_no, validity, name, max_limit, balance, cvv)
11. customer(customer_id, name, dob, Phone_number, Email, SSN, card_no, room_no, hotel_id, guest_count, check_in, check_out, start_date, end_date)
12. bill(Bill_id, customer_id, Bill_date, Amount, Discount, Total, Payment_type)
13. services(Service_name, Charges, Period)
14. serves(staff_id, customer_id);
15. cost(max_limit, category_name, costfactor)
16. offers(category_name, service_name)
17. uses(service_name, customer_id, amount)

From the above list, we find that each of the entities have unique IDs for identification. Therefore , the list of schemas must be in 3NF.

Now, let us analyze each of the schemas:

1. location(city, costfactor)

Solution: Since locations are uniquely identified by the city ids, therefore the following functional dependency(FD) obviously holds true:

city->costfactor

Since, this relation has only two attributes ,therefore this has to be in BCNF.

Since the functional dependency is in BCNF, therefore it has to be in 3NF.

2.hotel(hotel_id, Name, Phone_no, Address, City)

Solution: Since each hotel id uniquely identified by its hotel_id, therefore the below functional dependency holds true:

Hotel_id -> Name, Phone_no, Address, City

No other combination of name, phone_no, address or city will uniquely identify the details of the hotel.

Thus the LHS becomes a superkey. As a result the relation is in BCNF and therefore 3NF.

3.staff(staff_id, Name, Phone_no, Title, Pwd, Dob, Hotel_id)

Solution: Since the staff details are uniquely determined by staff_id, therefore the below functional dependency holds true.

Staff_id -> Name, Phone_no, Title, Pwd, Dob, Hotel_id

No other combination or subset of the RHS of the above functional dependency can uniquely determine unique staff details. It is pretty obvious that Name, Title, Pwd, Dob and hotel_id are details which many staffs can share. The delicate attribute is phone_no, the unique possibility of which can also be nullified by the below arguments:

- a) Staffs might stay in a dorm as residence and can use that phone number as the common phone number.
- b) Two or more staffs who are closely personally related may have shared same phone_no
- c) The old phone number of a staff may not have been updated in the system and this same phone number can be used by another staff when randomly assigned by the connection provider. System will throw no error, as we do not have any unique constraint on the phone number.

Thus, the functional dependency mentioned above is in BCNF and therefore in 3NF.

Name, DOB -> Phone#

In our declaration of the staff relation, name and DOB are not declared to be unique. Many staffs can share the same name as well as DOB. Thus the name, DOB combination may or may not determine a unique tuple. Thus this does not contribute.

Phone# -> Hotel_id

As mentioned in the bullet point 'c' and 'a' above, many staffs may have same phone number updated in the system. Thus Phone# alone does not provide a sufficient condition to determine a unique tuple.

4. catering_staff(staff_id)

Solution: The only functional dependency that holds here is:

staff_id->staff_id

This relation has to be in BCNF (therefore 3 NF) since it has only one attribute.

5.service_staff(staff_id):

Solution: The only functional dependency that holds here is:

staff_id->staff_id

This relation has to be in BCNF (therefore 3 NF) since it has only one attribute.

6.occupancy(max_limit):

Solution: The only functional dependency that holds here is:

max_limit->max_limit

This relation has to be in BCNF (therefore 3 NF) since it has only one attribute.

7.category(Category_name):

Solution: The only functional dependency that holds here is:

Category_name ->Category_name

This relation has to be in BCNF (therefore 3 NF) since it has only one attribute.

8. room(room_no, hotel_id, Availability, max_limit, category_name, rate)

Solution:

The below functional dependency holds true here:

Room_no, hotel_id -> Availability, max_limit, category_name, rate

We can see that:

1. Room_no -> Availability does not hold uniquely as there can be many "same room numbers" in different hotels, which can be available in not available in others. Similarly, room_no->max_limit or room_no -> category_name.

All the attributes Availability, max_limit,category_name are only unique as a combination when derived from the room_no,hotel_id combination. Therefore the LHS, Room_no,hotel_id is a superkey to the relation.

Therefore the relation is in BCNF and therefore in 3NF.

9. presidential(room_no, hotel_id, catering_staff_id,service_staff_id)

Solution:

The below functional dependency holds true:-

Room_no,hotel_id ->catering_staff_id,service_staff_id

Only this functional dependency holds. There can be no functional dependencies between staff_ids and the combination of room_no and hotel_id is a must to determine the unique id of a room as same room numbers can exist in different hotels.

Therefore , this FD is in BCNF and therefore in 3NF.

10. card(card_no, validity, name,max_limit,balance,cvv)

Solution:

The below functional dependency holds true:

Card_no -> validity, name,max_limit,balance,cvv

No other combination of attributes could uniquely determine card number. Therefore, this is in BCNF and therefore in 3NF.

11. customer(customer_id, name, dob, Phone_number, Email, SSN, card_no,room_no,hotel_id,guest_count,check_in,check_out, start_date, end_date)

Solution:

The below functional dependency holds true:

Customer_id -> name, dob, Phone_number, Email, SSN, card_no, room_no, hotel_id, guest_count, check_in, check_out, start_date, end_date

No other combination of the attributes could uniquely determine customer_id. Therefore this is in BCNF and therefore 3NF

12. bill(Bill_id, customer_id, Bill_date, Amount, Discount, Total, Payment_type)

Solution:

The below functional dependency holds true:

Bill_id -> customer_id, Bill_date, Amount, Discount, Total, Payment_type

No other combination of the attributes could uniquely determine Bill_id. Therefore this is in BCNF and therefore 3NF

13. services(Service_name, Charges, Period)

Solution:

The below functional dependency hold true:-

Service_name -> Charges, Period

The combination Charges -> Service_name or Period -> Service_name does not hold true as many services can have same charges and there can be different periods for the same service_names.

Thus other than, **Service_name -> Charges, Period**

No other combination of attribute will uniquely define the tuple. Therefore this is in BCNF and hence in 3NF.

14. serves(staff_id, customer_id)

Solution:

The below functional dependency hold true:-

Staff_id,customer_id -> Staff_id,customer_id

Any two attribute relation is in BCNF, therefore this is also in BCNF and therefore in 3NF.

15. cost(max_limit, category_name, costfactor)

Solution:

The below functional dependency hold true:-

Max_limit,category_name -costfactor

No other combination of attribute will uniquely define the tuple. Therefore this is in BCNF and hence in 3NF.

16. offers(category_name, service_name)

Solution:

The below functional dependency hold true:-

category_name,service_name -> category_name,service_name

Any two attribute relation is in BCNF, therefore this is also in BCNF and therefore in 3NF.

17. uses(service_name, customer_id, amount)

Solution:

The below functional dependency hold true:-

Service_name,customer_id ->amount

No other combination of attribute will uniquely define the tuple. Therefore this is in BCNF and hence in 3NF.

2) Design Decisions

We have used the mechanical approach to convert the relations to tables.

Location (city, cost_factor)

City (Primary Key) - unique identifier

cost_factor (NOT NULL) - to calculate the rate of room..every city has to have a costfactor associated with it.

hotel (hotel_id, name, phone#, address, city)

hotel_id (Primary Key) - unique identifier

name (NOT NULL) - Every hotel is required to have a hotel name

Phone_number - Hotel may or may not have a phone number but if there is a value in this field it has to be unique. No two hotels can have the same phone numbers.

Address - There is no restriction on availability of this value. Hotel may or may not have an address populated.

city (Referential integrity, NOT NULL) - refers to other entities within the database (location)

staff (staff_id, name, phone#, title, pwd, DOB, hotel_id)

staff_id (Primary Key) - unique identifier

name (NOT NULL) - Every staff is required to have a name

Phone_number - Staffs may or may not have phone number and also there is no unique constraint on the Phone_no values.

title (NOT NULL) - Each staff should have a role

pwd (NOT NULL) - Each staff should have a password to login

DOB - This can have null values. Staffs may or may not have date of birth updated in the system.

hotel_id (Referential integrity, NOT NULL) - This is the referential integrity constraint and it references the primary key hotel_id of the table hotel. This value cannot be null in the staff table. Each staff has to be assigned to a hotel.

catering_staff(staff_id)

staff_id (Referential integrity, Primary Key) - This is the unique identifier and Primary key to the catering_staff table. But it is actually a referential integrity constraint which references the Primary Key staff_id of the staff table.

service_staff(staff_id)

staff_id (Referential integrity, Primary Key) - This is the unique identifier and Primary key to the service_staff table. But it is actually a referential integrity constraint which references the Primary Key staff_id of the staff table.

occupancy(max_limit)

max_limit (Primary Key) - unique identifier

category(category_name)

category_name(Primary Key)- unique identifier

room(room_no, hotel_id, availability, max_limit, category_name,rate)

room_no(Primary Key) - unique identifier

hotel_id(Referential integrity ,Primary Key) - unique identifier. But hotel_id is also a referential integrity constraint and it references to the hotel_id attribute in the hotel table.

availability (NOT NULL) -Each room should have its availability status

max_limit(Referential integrity) - This is a multivalued referential integrity constraint which references to the max_limit field of the occupancy table.

category_name(Referential integrity) - This is a multivalued referential integrity constraint which references to the category_name field of the category table.

rate (NOT NULL) - each room should have a nightly rate to charge the customer using it

presidential(room_no, hotel_id , catering_staff_id, service_staff_id)

room_no(Referential integrity, Primary Key) - Unique identifier. But room_no is also a referential integrity constraint and it references to the room_no attribute in the room table.

hotel_id(Referential integrity, Primary Key)- unique identifier . But hotel_id is also a referential integrity constraint and it references to the hotel_id attribute in the hotel table.

catering_staff_id(Referential integrity, NOT NULL)-This is a referential integrity constraint which references the staff_id in the catering_staff table. and each presidential room should have a catering staff with id.

service_staff_id(Referential integrity, NOT NULL)- this is a referential integrity constraint which references the staff_id in the service_staff table. Each presidential room should have a service staff with id

customer (customer_id, name, DOB, phone_number, email, SSN, card_no, room_no, hotel_id, guest_count, check_in, check_out, start_date, end_date)

customer_id(Primary Key)- unique identifier

name(NOT NULL) - each customer should have name

DOB -This is non mandatory field for customer. Customers may or may not choose to disclose their date of birth. Thus DOB can have null value.

Phone_number - This is non mandatory field for customer. Customers may or may not choose to disclose their phone number. Thus Phone_number can have null value.

Email - This is non mandatory field for customer. Customers may or may not choose to disclose their email id. Thus email id can have null value.

SSN -This is non mandatory field for customer. Customers may or may not choose to disclose their SSN . ThusSSN can have null value.

Card_no (Referential integrity)- If there is a card_number associated with the customer, then, it is referenced from the card_no in the card table. Thus card_no is a referential integrity constraint.

Room_no (Referential integrity, NOT NULL) - refers to room_no within the database (room) and each customer should reside in a room

hotel_id(Referential integrity, NOT NULL) - refers to hotel_id within the database (room) and each customer should reside in a room that belongs to a hotel

guest_count(NOT NULL) - each customer should have count of guests to stay

check_in(NOT NULL) - each customer should have check in time

Check_out - This field can be null, If a customer is currently checked in, but once the customer checks out of the hotel, this should be populated with the check out time

start_date (NOT NULL)- each customer should have start date i.e. when they started using room

End_date - This field can be null, If a customer is currently checked in, but once the customer checks out of the hotel, this should be populated with the check out date.

card(cardno,validity,name,max_limit,balance,cvv)

cardno, (Primary Key)

Validity (NOT NULL)- Every card should have till what time card is valid

Name (NOT NULL)- Every card should be associated with a name of the holder

max_limit (NOT NULL) - Every card should have its limit of expenditure

balance(NOT NULL)- Every card should have balance

cvv(NOT NULL)- Every card should have a cvv number

bill(bill_id,customer_id,bill_date,amount,discount,total,payment_type)

Bill_id (Primary Key) - unique identifier

customer_id (Referential integrity, NOT NULL) -This is a referential integrity constraint and refers to the customer_id from the customer table. This cannot be null as bills have to be associated with customers. But since there can be multiple bills of the same customers, therefore the customer_id can have same values in multiple tuples.

Bill_date -This bill will have the date when the bills will be cleared by the customer. Until then it can have null value.

Amount - The bill amount will be stored in this attribute. This will have the amounts spent in various things

Discount - This will contain any discount that has to be applied to the customer.

Total - This has the bill total .

Payment_type - This will have the payment type of the customer to clear the bill

services(service_name,charges,period)

service_name (Primary Key) - unique identifier

charges(NOT NULL)- each service should have a charge for using it

period(NOT NULL)- each service should have time period, according to which charges are applied.

serves(staff_id,customer_id)

Staff_id (Referential integrity ,Primary Key)- unique identifier and refers staff_id within the database (staff)

customer_id (Referential integrity ,Primary Key)- unique identifier and refers to customer_id within the database (customer)

cost(limit,category_name,costfactor)

limit, (Referential integrity ,Primary Key)- unique identifier and refers to other entities within the database (occupancy)

category_name, (Referential integrity ,Primary Key)- unique identifier and refers to other entities within the database (category)

costfactor (NOT NULL)

offers(category_name,service_name)

category_name, (Referential integrity ,Primary Key)- unique identifier and refers to other entities within the database (category)

service_name (Referential integrity ,Primary Key)- unique identifier and refers to other entities within the database (services)

uses(service_name, customer_id, amount)

service_name (Referential integrity, PRIMARY KEY)- unique identifier. But service_name is also a referential integrity constraint and it references to the service_name attribute in the services table

customer_id (Referential integrity, PRIMARY KEY)- unique identifier . But customer_id is also a referential integrity constraint and it references to the customer_id attribute in the customer table.

amount (NOT NULL) - each customer using a service should be charged according to service rates

3) SQL Statements**Drop Table**

```
DROP TABLE location CASCADE CONSTRAINTS;
DROP TABLE hotel CASCADE CONSTRAINTS;
DROP TABLE staff CASCADE CONSTRAINTS;
DROP TABLE catering_staff CASCADE CONSTRAINTS;
DROP TABLE room_service_staff CASCADE CONSTRAINTS;
DROP TABLE occupancy CASCADE CONSTRAINTS;
DROP TABLE category CASCADE CONSTRAINTS;
DROP TABLE room CASCADE CONSTRAINTS;
DROP TABLE presidential CASCADE CONSTRAINTS;
```

```

DROP TABLE customer CASCADE CONSTRAINTS;
DROP TABLE card CASCADE CONSTRAINTS;
DROP TABLE bill CASCADE CONSTRAINTS;
DROP TABLE services CASCADE CONSTRAINTS;
DROP TABLE serves CASCADE CONSTRAINTS;
DROP TABLE cost CASCADE CONSTRAINTS;
DROP TABLE offers CASCADE CONSTRAINTS;
DROP TABLE uses CASCADE CONSTRAINTS;

```

Create Tables

```

CREATE TABLE location(
city VARCHAR(20),
costfactor FLOAT(2) NOT NULL,
CONSTRAINT location_pk PRIMARY KEY(city)
);

```

```

CREATE TABLE hotel(
hotel_id INT(20) AUTO_INCREMENT,
Name VARCHAR(50) NOT NULL,
Phone_no BIGINT(10) UNIQUE,
Address VARCHAR(50),
City VARCHAR(20) NOT NULL,
CONSTRAINT hotel_pk PRIMARY KEY(hotel_id),
CONSTRAINT hotel_location_fk FOREIGN KEY(city) REFERENCES
location(city) ON DELETE CASCADE
);

```

```

CREATE TABLE staff(
staff_id INT(50) AUTO_INCREMENT,
Name VARCHAR(50) NOT NULL,
Phone_no BIGINT(10),
Title VARCHAR(50) NOT NULL,
Pwd VARCHAR(50) NOT NULL,
Dob DATE,
Hotel_id INT(20) NOT NULL,
CONSTRAINT staff_pk PRIMARY KEY(staff_id),
CONSTRAINT staff_hotel_fk FOREIGN KEY(hotel_id) REFERENCES
hotel(hotel_id) ON DELETE CASCADE
);

```

```

CREATE TABLE catering_staff(

```

```

Staff_id INT(50),
CONSTRAINT catering_staff_pk PRIMARY KEY(staff_id),
CONSTRAINT catering_staff_fk FOREIGN KEY(staff_id) REFERENCES
staff(staff_id)ON DELETE CASCADE
);

```

```

CREATE TABLE service_staff(
Staff_id INT(50),
CONSTRAINT room_service_pk PRIMARY KEY(staff_id),
CONSTRAINT room_staff_fk FOREIGN KEY(staff_id) REFERENCES
staff(staff_id)ON DELETE CASCADE
);

```

```

CREATE TABLE occupancy(
max_limit INT(2),
CONSTRAINT occupancy_pk PRIMARY KEY(max_limit)
);

```

```

CREATE TABLE category(
Category_name VARCHAR(50),
CONSTRAINT category_pk PRIMARY KEY(category_name)
);

```

```

CREATE TABLE room(
Room_no INT(10),
hotel_id INT(20),
Availability BIT NOT NULL,
max_limit INT(2),
category_name VARCHAR(50),
rate FLOAT(2) NOT NULL,
CONSTRAINT room_pk PRIMARY KEY(room_no, hotel_id),
CONSTRAINT room_hotel_fk FOREIGN KEY(hotel_id) REFERENCES
hotel(hotel_id)ON DELETE CASCADE,
CONSTRAINT room_occupancy_fk FOREIGN KEY(max_limit) REFERENCES
occupancy(max_limit)ON DELETE CASCADE,
CONSTRAINT room_category_fk FOREIGN KEY(category_name)
REFERENCES category(category_name)ON DELETE CASCADE
);

```

```

CREATE TABLE presidential(
Room_no INT(10),
Hotel_id INT(20),
Catering_staff_id INT(20) NOT NULL,
Service_staff_id INT(20) NOT NULL,
CONSTRAINT presidential_pk PRIMARY KEY(hotel_id,room_no),

```

```

CONSTRAINT presidential_room_fk FOREIGN KEY(room_no,hotel_id)
REFERENCES room(room_no, hotel_id)ON DELETE CASCADE,
CONSTRAINT presidential_catering_fk FOREIGN
KEY(catering_staff_id) REFERENCES catering_staff(staff_id)ON
DELETE CASCADE,
CONSTRAINT presidential_service_fk FOREIGN KEY(service_staff_id)
REFERENCES service_staff(staff_id)ON DELETE CASCADE
);

```

```

CREATE TABLE card(
Card_no BIGINT(16),
Validity DATE NOT NULL,
Name VARCHAR(20) NOT NULL,
max_limit FLOAT(10,2) NOT NULL,
Balance FLOAT(10,2) NOT NULL,
Cvv INT(3) NOT NULL,
CONSTRAINT card_pk PRIMARY KEY(card_no)
);

```

```

CREATE TABLE customer(
Customer_id INT(20) AUTO_INCREMENT,
Name VARCHAR(50) NOT NULL,
DOB DATE,
Phone_number BIGINT(10),
Email VARCHAR(30),
SSN INT(9),
Card_no BIGINT(16),
Room_no INT(10) NOT NULL,
Hotel_id INT(20) NOT NULL,
Guest_count INT(20) NOT NULL,
Check_in TIME NOT NULL,
Check_out TIME,
Start_date DATE NOT NULL,
End_date DATE,
CONSTRAINT customer_pk PRIMARY KEY(customer_id),
CONSTRAINT customer_room_fk FOREIGN KEY(room_no,hotel_id)
REFERENCES room(room_no,hotel_id)ON DELETE CASCADE,
CONSTRAINT customer_card_fk FOREIGN KEY(card_no) REFERENCES
card(card_no)ON DELETE CASCADE
);

```

```

CREATE TABLE bill(
Bill_id INT(20) AUTO_INCREMENT,
customer_id INT(20) NOT NULL,
Bill_date DATE,

```

```

Amount FLOAT(2),
Discount INT(3),
Total FLOAT(2),
Payment_type varchar(20),
CONSTRAINT bill_pk PRIMARY KEY(bill_id),
CONSTRAINT bill_customer_fk FOREIGN KEY(customer_id) REFERENCES
customer(customer_id) ON DELETE CASCADE
);

```

```

CREATE TABLE services(
Service_name VARCHAR(20),
Charges FLOAT(2),
Period varchar(20),
CONSTRAINT services_pk PRIMARY KEY(service_name)
);

```

```

CREATE TABLE serves(
Staff_id INT(20),
Customer_id INT(20),
CONSTRAINT serves_pk PRIMARY KEY(staff_id,customer_id),
CONSTRAINT serves_staff_fk FOREIGN KEY(staff_id) REFERENCES
staff(staff_id) ON DELETE CASCADE,
CONSTRAINT serves_customer_fk FOREIGN KEY(customer_id)
REFERENCES customer(customer_id) ON DELETE CASCADE
);

```

```

CREATE TABLE cost(
max_limit INT(2),
Category_name VARCHAR(20),
Costfactor FLOAT(2) NOT NULL,
CONSTRAINT cost_pk PRIMARY KEY(max_limit,category_name),
CONSTRAINT cost_occupancy_fk FOREIGN KEY(max_limit) REFERENCES
occupancy(max_limit) ON DELETE CASCADE,
CONSTRAINT cost_category_fk FOREIGN KEY(category_name)
REFERENCES category(category_name) ON DELETE CASCADE
);

```

```

CREATE TABLE offers(
Category_name VARCHAR(20),
Service_name VARCHAR(20),
CONSTRAINT offers_pk PRIMARY KEY(category_name,service_name),
CONSTRAINT offers_category_fk FOREIGN KEY(category_name)
REFERENCES category(category_name) ON DELETE CASCADE,
CONSTRAINT offers_service_fk FOREIGN KEY(service_name)
REFERENCES services(service_name) ON DELETE CASCADE

```

```
);
```

```
CREATE TABLE uses(
Service_name VARCHAR(20),
Customer_id INT,
Amount FLOAT(2) NOT NULL,
CONSTRAINT usagelog_pk PRIMARY KEY(service_name,customer_id),
CONSTRAINT usage_service_fk FOREIGN KEY(service_name) REFERENCES
services(service_name)ON DELETE CASCADE,
CONSTRAINT usage_customer_id FOREIGN KEY(customer_id) REFERENCES
customer(customer_id)ON DELETE CASCADE
);
```

Inserts

```
INSERT INTO location(city, costfactor) VALUES ('duhram', 50);
INSERT INTO location(city, costfactor) VALUES ('Los Angeles',
200);
INSERT INTO location(city, costfactor) VALUES ('Raleigh',
100.00);
```

```
INSERT INTO hotel(name, phone_no, address, city) VALUES
('wolfie1', 9192601578, '1690 Crest Road, block 1', 'Raleigh');
INSERT INTO hotel(name, phone_no, address, city) VALUES
('wolfie2', 9192601579, '1630 Avent ferry, block 4', 'Raleigh');
INSERT INTO hotel(name, phone_no, address, city) VALUES
('wollfie3', 9192601577, '238 Belmont road, Red block', 'Los
Angeles');
```

```
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Alex', 9193601234, 'manager', '123qwe', '1990-06-14',
1);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Clara', 9193249532, 'catering', 'fgj678',
'1989-08-15', 1);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Joe', 9145323456, 'service', 'fvmb056!', '1995-01-22',
1);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Simi', 9234578961, 'manager', '4f0fo03', '1994-09-02',
2);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Maria', 8763251738, 'catering', 'kw9320',
'1994-01-09', 2);
```



```

INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Arya', 4563920139, 'manager', 'sjfo!#', '1993-10-11',
3);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Deep', 4930257392, 'service', 'sjdff', '1985-12-30',
3);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'John', 5687920193, 'front desk', 'sjdq', '1987-03-05',
1);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Tim', 3475969409, 'front desk', 'wejf', '1995-07-16',
2);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Kelly', 9192378937, 'front desk', '123qejk',
'1995-09-10', 3);
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'Sal', 91928374689, 'service', 'qejf', '1992-02-28',
2);

INSERT INTO catering_staff
SELECT staff_id from staff where title="catering";

INSERT INTO service_staff
SELECT staff_id from staff where title="service";

INSERT INTO occupancy(max_limit) VALUES (1);
INSERT INTO occupancy(max_limit) VALUES (2);
INSERT INTO occupancy(max_limit) VALUES (3);
INSERT INTO occupancy(max_limit) VALUES (4);

INSERT INTO category(category_name) VALUES ("economy");
INSERT INTO category(category_name) VALUES ("executive");
INSERT INTO category(category_name) VALUES ("deluxe");
INSERT INTO category(category_name) VALUES ("presidential");

INSERT INTO card(card_no,validity,name,max_limit,balance,cvv)
VALUES (1234567812345678, '2022-09-30', 'jimmy', 1000, 600,
789);
INSERT INTO card(card_no,validity,name,max_limit,balance,cvv)
VALUES (1234567845678912, '2025-08-30', 'teddy', 2000, 800,
467);
INSERT INTO card(card_no,validity,name,max_limit,balance,cvv)
VALUES (1234567832165498, '2023-03-31', 'shree', 1500, 1500,
139);

```

```

INSERT INTO cost(max_limit,category_name,costfactor) VALUES (1,
'presidential', 100);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (1,
'executive', 90);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (1,
'deluxe', 75);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (1,
'economy', 50);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (2,
'presidential', 150);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (2,
'executive', 110);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (2,
'deluxe', 100);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (2,
'economy', 75);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (3,
'presidential', 200);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (3,
'executive', 150);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (3,
'deluxe', 120);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (3,
'economy', 100);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (4,
'presidential', 250);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (4,
'executive', 180);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (4,
'deluxe', 150);
INSERT INTO cost(max_limit,category_name,costfactor) VALUES (4,
'economy', 125);

```

```

INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT
1,1,1,1,'presidential',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 1 AND category_name = 'presidential' AND
city=(SELECT city from hotel where hotel_id= 1);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 2,1,1,2,'executive',cost.costfactor+location.costfactor
FROM cost, location

```

```

WHERE max_limit = 2 AND category_name = 'executive' AND
city=(SELECT city from hotel where hotel_id= 1);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 3,1,1,3, 'deluxe',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 3 AND category_name = 'deluxe' AND
city=(SELECT city from hotel where hotel_id= 1);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 4,1,1,4, 'economy',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 4 AND category_name = 'economy' AND
city=(SELECT city from hotel where hotel_id= 1);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT
101,2,1,1,'presidential',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 1 AND category_name = 'presidential' AND
city=(SELECT city from hotel where hotel_id= 2);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 102,2,1,2, 'deluxe',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 2 AND category_name = 'deluxe' AND
city=(SELECT city from hotel where hotel_id= 2);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 103,2,1,3,'economy',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 3 AND category_name = 'economy' AND
city=(SELECT city from hotel where hotel_id= 2);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 1,3,1,1,'executive',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 1 AND category_name = 'executive' AND
city=(SELECT city from hotel where hotel_id= 3);
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 2,3,1,1,'economy',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 1 AND category_name = 'economy' AND
city=(SELECT city from hotel where hotel_id= 3);

```

```

INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name,rate)
SELECT 3,3,1,2,'deluxe',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 2 AND category_name = 'deluxe' AND
city=(SELECT city from hotel where hotel_id= 3);

```

```

INSERT INTO
presidential(room_no,hotel_id,catering_staff_id,service_staff_id
) VALUES (1,1,2,3);
INSERT INTO
presidential(room_no,hotel_id,catering_staff_id,service_staff_id
) VALUES (101,2,5,11);

```

```

INSERT INTO customer (name, DOB, phone_number, email, SSN,
card_no, room_no, hotel_id,
guest_count, check_in, start_date) VALUES ("jimmy","1979-05-05",
6786975432,"jim@gmail.com",845869825,1234567812345678,1,1,1,"15:
30:00","2018-01-30");
INSERT INTO customer (name, DOB, phone_number, email, SSN,
card_no, room_no, hotel_id,
guest_count, check_in, start_date) VALUES ("teddy","1997-11-08",
3589746582,"ted@gmail.com",845869875,1234567845678912,4,1,4,"11:
00:00","2018-03-12");
INSERT INTO customer (name, DOB, phone_number, email, SSN,
room_no, hotel_id,
guest_count, check_in, start_date) VALUES ("sammy","1985-08-24",
9192375892,"sam@gmail.com",845869725,102,2,2,"20:15:00","2018-02
-11");
INSERT INTO customer (name, DOB, phone_number, email, SSN,
room_no, hotel_id,
guest_count, check_in, start_date) VALUES ("barry","1990-10-12",
9192347586,"barr@yahoo.com",845867825,103,2,3,"17:50:00","2017-1
0-09");
INSERT INTO customer (name, DOB, phone_number, email, SSN,
room_no, hotel_id,
guest_count, check_in, start_date) VALUES
("hannah","1987-01-12",
9847223828,"hannah@gmail.com",845769825,1,3,1,"22:00:00","018-03
-10");

```

```

INSERT INTO bill(customer_id) VALUES (1);

```

```

INSERT INTO bill(customer_id) VALUES (2);
INSERT INTO bill(customer_id) VALUES (3);
INSERT INTO bill(customer_id) VALUES (4);
INSERT INTO bill(customer_id) VALUES (5);

INSERT INTO services(service_name,charges,period) VALUES
('phone', 2, 'minute');
INSERT INTO services(service_name,charges,period) VALUES ('dry
cleaning', 5, 'load');
INSERT INTO services(service_name,charges,period) VALUES ('gym',
20, 'hour');
INSERT INTO services(service_name,charges,period) VALUES ('spa',
50, 'session');
INSERT INTO services(service_name,charges,period) VALUES
('pool', 15, 'hour');
INSERT INTO services(service_name,charges,period) VALUES ('room
service', 7, 'order');
INSERT INTO services(service_name,charges,period) VALUES ('cater
service', 10, 'order');
INSERT INTO services(service_name,charges,period) VALUES
('food', 13, 'meal');

INSERT INTO serves(staff_id,customer_id) VALUES (8,2);
INSERT INTO serves(staff_id,customer_id) VALUES (8,1);
INSERT INTO serves(staff_id,customer_id) VALUES (9,3);
INSERT INTO serves(staff_id,customer_id) VALUES (10,5);
INSERT INTO serves(staff_id,customer_id) VALUES (9,4);

INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'phone');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'dry cleaning');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'gym');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'spa');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'pool');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'room service');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'cater service');
INSERT INTO offers(category_name,service_name) VALUES
('presidential', 'food');

```

```

INSERT INTO offers(category_name,service_name) VALUES
('executive', 'phone');
INSERT INTO offers(category_name,service_name) VALUES
('executive', 'dry cleaning');
INSERT INTO offers(category_name,service_name) VALUES
('executive', 'gym');
INSERT INTO offers(category_name,service_name) VALUES
('executive', 'pool');
INSERT INTO offers(category_name,service_name) VALUES
('executive', 'food');
INSERT INTO offers(category_name,service_name) VALUES ('deluxe',
'phone');
INSERT INTO offers(category_name,service_name) VALUES ('deluxe',
'dry cleaning');
INSERT INTO offers(category_name,service_name) VALUES ('deluxe',
'food');

INSERT INTO uses(service_name, customer_id,amount) VALUES
("pool",2,30);
INSERT INTO uses(service_name, customer_id,amount) VALUES
("food",2,26);
INSERT INTO uses(service_name, customer_id,amount) VALUES ("room
service",3,14);
INSERT INTO uses(service_name, customer_id,amount) VALUES
("gym",3,20);
INSERT INTO uses(service_name, customer_id,amount) VALUES
("cater service",5,20);

```

Selects

select * from location;

| city | costfactor |
|-------------|------------|
| duhram | 50 |
| Los Angeles | 200 |
| Raleigh | 100 |

3 rows in set (0.01 sec)

select * from hotel;

| hotel_id | Name | Phone_no | Address | City |
|----------|----------|------------|-----------------------------|-------------|
| 1 | wolfie1 | 9192601578 | 1690 Crest Road, block 1 | Raleigh |
| 2 | wolfie2 | 9192601579 | 1630 Avent ferry, block 4 | Raleigh |
| 3 | wollfie3 | 9192601577 | 238 Belmont road, Red block | Los Angeles |

3 rows in set (0.01 sec)

select * from staff;

| staff_id | Name | Phone_no | Title | Pwd | Dob | Hotel_id |
|----------|-------|-------------|------------|----------|------------|----------|
| 1 | Alex | 9193601234 | manager | 123qwe | 1990-06-14 | 1 |
| 2 | Clara | 9193249532 | catering | fgj678 | 1989-08-15 | 1 |
| 3 | Joe | 9145323456 | service | fvmbo56! | 1995-01-22 | 1 |
| 4 | Simi | 9234578961 | manager | 4f0fo03 | 1994-09-02 | 2 |
| 5 | Maria | 8763251738 | catering | kw9320 | 1994-01-09 | 2 |
| 6 | Arya | 4563920139 | manager | sjfo!# | 1993-10-11 | 3 |
| 7 | Deep | 4930257392 | service | sjdff | 1985-12-30 | 3 |
| 8 | John | 5687920193 | front desk | sjdq | 1987-03-05 | 1 |
| 9 | Tim | 3475969409 | front desk | wejf | 1995-07-16 | 2 |
| 10 | Kelly | 9192378937 | front desk | 123qejk | 1995-09-10 | 3 |
| 11 | Sal | 91928374689 | service | qejf | 1992-02-28 | 2 |

11 rows in set (0.01 sec)

select * from catering_staff;

| Staff_id |
|----------|
| 2 |
| 5 |

2 rows in set (0.01 sec)

select * from service_staff;

| Staff_id |
|----------|
| 3 |
| 7 |
| 11 |

3 rows in set (0.01 sec)

select * from occupancy;

| max_limit |
|-----------|
| 1 |
| 2 |
| 3 |
| 4 |

4 rows in set (0.01 sec)

select * from category;

| Category_name |
|---------------|
| deluxe |
| economy |
| executive |
| presidential |

4 rows in set (0.01 sec)

select * from room;

| Room_no | hotel_id | Availability | max_limit | category_name | rate |
|---------|----------|--------------|-----------|---------------|------|
| 1 | 1 | ⊗ | 1 | presidential | 200 |
| 1 | 3 | ⊗ | 1 | executive | 290 |
| 2 | 1 | ⊗ | 2 | executive | 210 |
| 2 | 3 | ⊗ | 1 | economy | 250 |
| 3 | 3 | ⊗ | 2 | deluxe | 300 |
| 13 | 1 | ⊗ | 3 | deluxe | 220 |
| 14 | 1 | ⊗ | 4 | economy | 225 |
| 101 | 2 | ⊗ | 1 | presidential | 200 |
| 102 | 2 | ⊗ | 2 | deluxe | 200 |
| 103 | 2 | ⊗ | 3 | economy | 200 |

10 rows in set (0.01 sec)

select * from presidential;

| Room_no | Hotel_id | Catering_staff_id | Service_staff_id |
|---------|----------|-------------------|------------------|
| 1 | 1 | 2 | 3 |
| 101 | 2 | 5 | 11 |

2 rows in set (0.00 sec)

select * from card;

| Card_no | Validity | Name | max_limit | Balance | Cvv |
|------------------|------------|-------|-----------|---------|-----|
| 1234567812345678 | 2022-09-30 | jimmy | 1000.00 | 600.00 | 789 |
| 1234567832165498 | 2023-03-31 | shree | 1500.00 | 1500.00 | 139 |
| 1234567845678912 | 2025-08-30 | teddy | 2000.00 | 800.00 | 467 |

3 rows in set (0.01 sec)

select * from cost;

| max_Limit | Category_name | Costfactor |
|-----------|---------------|------------|
| 1 | deluxe | 75 |
| 1 | economy | 50 |
| 1 | executive | 90 |
| 1 | presidential | 100 |
| 2 | deluxe | 100 |
| 2 | economy | 75 |
| 2 | executive | 110 |
| 2 | presidential | 150 |
| 3 | deluxe | 120 |
| 3 | economy | 100 |
| 3 | executive | 150 |
| 3 | presidential | 200 |
| 4 | deluxe | 150 |
| 4 | economy | 125 |
| 4 | executive | 180 |
| 4 | presidential | 250 |

16 rows in set (0.01 sec)

select * from customer;

| Customer_id | Name | DOB | Phone_number | Email | SSN |
|-------------|--------|------------|--------------|------------------|-----------|
| 1 | jimmy | 1979-05-05 | 6786975432 | jim@gmail.com | 845869825 |
| 2 | teddy | 1997-11-08 | 3589746582 | ted@gmail.com | 845869875 |
| 3 | sammy | 1985-08-24 | 9192375892 | sam@gmail.com | 845869725 |
| 4 | barry | 1990-10-12 | 9192347586 | barr@yahoo.com | 845867825 |
| 5 | hannah | 1987-01-12 | 9847223828 | hannah@gmail.com | 845769825 |

5 rows in set (0.01 sec)

| Card_no | Room_no | Hotel_id | Guest_count | Check_in | Check_out | Start_date | End_date |
|------------------|---------|----------|-------------|----------|-----------|------------|----------|
| 1234567812345678 | 1 | 1 | 1 | 15:30:00 | NULL | 2018-01-30 | NULL |
| 1234567845678912 | 4 | 1 | 4 | 11:00:00 | NULL | 2018-03-12 | NULL |
| NULL | 102 | 2 | 2 | 20:15:00 | NULL | 2018-02-11 | NULL |
| NULL | 103 | 2 | 3 | 17:50:00 | NULL | 2017-10-09 | NULL |
| NULL | 1 | 3 | 1 | 22:00:00 | NULL | 0018-03-10 | NULL |

select * from bill;

| Bill_id | customer_id | Bill_date | Amount | Discount | Total | Payment_type |
|---------|-------------|-----------|--------|----------|-------|--------------|
| 1 | 1 | NULL | NULL | NULL | NULL | NULL |
| 2 | 2 | NULL | NULL | NULL | NULL | NULL |
| 3 | 3 | NULL | NULL | NULL | NULL | NULL |
| 4 | 4 | NULL | NULL | NULL | NULL | NULL |
| 5 | 5 | NULL | NULL | NULL | NULL | NULL |

5 rows in set (0.04 sec)

select * from services;

| Service_name | Charges | Period |
|---------------|---------|---------|
| cater service | 10 | order |
| dry cleaning | 5 | load |
| food | 13 | meal |
| gym | 20 | hour |
| phone | 2 | minute |
| pool | 15 | hour |
| room service | 7 | order |
| spa | 50 | session |

8 rows in set (0.01 sec)

select * from serves;

| Staff_id | Customer_id |
|----------|-------------|
| 8 | 1 |
| 8 | 2 |
| 9 | 3 |
| 9 | 4 |
| 10 | 5 |

5 rows in set (0.01 sec)

select * from offers;

| Category_name | Service_name |
|---------------|---------------|
| presidential | cater service |
| deluxe | dry cleaning |
| executive | dry cleaning |
| presidential | dry cleaning |
| deluxe | food |
| executive | food |
| presidential | food |
| executive | gym |
| presidential | gym |
| deluxe | phone |
| executive | phone |
| presidential | phone |
| executive | pool |
| presidential | pool |
| presidential | room service |
| presidential | spa |

16 rows in set (0.01 sec)

select * from uses;

| Service_name | Customer_id | Amount |
|---------------|-------------|--------|
| cater service | 5 | 20 |
| food | 2 | 26 |
| gym | 3 | 20 |
| pool | 2 | 30 |
| room service | 3 | 14 |

5 rows in set (0.01 sec)

5) Application Program Interfaces

Information Processing

Enter a new location

```
INSERT INTO location(city, costfactor) VALUES ('Raleigh',
100.00);
```

1 row(s) affected

Update Location Cost

```
UPDATE location SET costfactor=125 WHERE city='Raleigh';
```

Query OK, 1 rows affected (0.10 sec)

Rows matched: 1 Changed: 1 Warnings: 0

Delete Location

```
DELETE from location WHERE city='Raleigh';
```

```
1 row(s) affected
```

Enter a hotel

```
INSERT INTO hotel(name, phone_no, address, city) VALUES
('wolfiel', 9192601578, '1690 Crest Road, block 1', 'Raleigh');
```

```
1 row(s) affected
```

Update hotel

```
UPDATE hotel SET name='wolfiel', phone_no=9192601575,
address='11690 Crest Road, block 1 NC' WHERE hotel_id=1;
```

```
Query OK, 1 rows affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete hotel

```
DELETE from hotel WHERE hotel_id=1;
```

```
1 row(s) affected
```

Enter a staff

```
INSERT INTO staff( name, phone_no, title, pwd, DOB, hotel_id)
VALUES ( 'John', 5687920193, 'front desk', 'sjdq', '1987-03-05',
1);
```

```
1 row(s) affected
```

Update staff

```
UPDATE staff SET name='John Mathew', title='service',
pwd='fdesk2453' WHERE staff_id=8;
```

```
Query OK, 1 rows affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete staff

```
DELETE from staff WHERE staff_id=8;
```

1 row(s) affected

Enter a customer

```
INSERT INTO customer (name, DOB, phone_number, email, SSN,
card_no, room_no, hotel_id,
guest_count, check_in, start_date) VALUES ("teddy", "1997-11-08",
3589746582, "ted@gmail.com", 845869875, 1234567845678912, 4, 1, 4, "11:
00:00", "2018-03-12");
```

1 row(s) affected

Update customer

```
UPDATE customer SET name='teddy', DOB='1997-02-08',
phone_number=3589746582, email="ted@gmail.com",
SSN=845869875 WHERE customer_id= 2;
```

Query OK, 1 rows affected (0.10 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Delete customer

```
DELETE from customer WHERE customer_id=2;
```

1 row(s) affected

Enter a room

```
INSERT INTO room(room_no, hotel_id, availability, max_limit,
category_name, rate)
SELECT
1,1,1,1,'presidential',cost.costfactor+location.costfactor
FROM cost, location
WHERE max_limit = 1 AND category_name = 'presidential' AND
city=(SELECT city from hotel where hotel_id= 1);
```

1 row(s) affected

Update room

```
UPDATE room,
(SELECT cost.costfactor+location.costfactor as rate
FROM cost, location
WHERE max_limit = 4 AND category_name = 'economy' AND
city=(SELECT city from hotel where hotel_id= 1)) as a
SET max_limit= 4, category_name='economy',room.rate=a.rate
WHERE room_no=103 AND hotel_id=2;
```

Query OK, 1 rows affected (0.10 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Delete room

```
DELETE from room WHERE room_no=103;
```

1 row(s) affected

Room Available

```
SELECT room_no, max_limit, category_name, rate FROM room where
hotel_id=1 AND availability=1;
```

| room_no | max_limit | category_name | rate |
|---------|-----------|---------------|------|
| 1 | 1 | presidential | 200 |
| 2 | 2 | executive | 210 |
| 3 | 3 | deluxe | 220 |
| 4 | 4 | economy | 225 |

4 rows in set (0.01 sec)

Requested Room Available

```
SELECT * FROM room where hotel_id=2 AND availability=1 AND
category_name='presidential' AND max_limit=1;
```

| Room_no | hotel_id | Availability | max_limit | category_name | rate |
|---------|----------|--------------|-----------|---------------|------|
| 101 | 2 | ⊗ | 1 | presidential | 225 |

1 row in set (0.00 sec)

release room

```
UPDATE room SET availability=1 WHERE room_no=1 AND hotel_id=1;
```

```
Query OK, 1 rows affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Maintaining Service Records:

Enter a new service

```
INSERT INTO services(service_name,charges,period) VALUES
('gym',10,'hour');
```

```
1 row(s) affected
```

Update charges

```
UPDATE services SET charges=12 WHERE service_name = 'gym';
```

```
Query OK, 1 rows affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete service

```
DELETE from services WHERE service_name ='gym';
```

```
1 row(s) affected
```

roomServiceOffered

```
SELECT count(*) from offers WHERE category_name =
'presidential' AND service_name = 'gym' ;
```

| count(*) |
|----------|
| 1 |

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

enterServiceUsed

```
INSERT INTO uses(service_name, customer_id,amount)
SELECT 'pool',2,charges
FROM services
WHERE service_name='pool';
```

```
1 row(s) affected
```

Update service used

```
UPDATE uses,(SELECT (charges*2) as amount FROM services WHERE
service_name='pool') as a
SET uses.amount= a.amount+uses.amount
where uses.service_name='pool' AND customer_id=2;
```

Query OK, 1 rows affected (0.10 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Generate bill

```
INSERT INTO bill(customer_id) VALUES (1);
```

1 row(s) affected

Update bill

```
UPDATE bill
SET bill.amount= (select rate*(datediff(End_date,Start_date))
from customer natural join room where
customer_id=2)+(select coalesce(sum(amount),0) from uses where
customer_id=2), bill_date=(select end_date
from customer where customer_id=2)
WHERE customer_id =2;
```

Query OK, 1 rows affected (0.10 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Update bill if paying through hotel credit card

```
UPDATE bill,card
SET bill.discount=5, bill.Payment_type='hotel credit card',
bill.total= bill.amount-(bill.amount*5/100),
card.balance= card.Balance-(bill.amount-(bill.amount*5/100))
where bill.customer_id=2 and card_no=(select card_no from
customer where customer_id=2);
```

Query OK, 1 rows affected (0.10 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Update bill if not paying through hotel credit card

```
UPDATE bill
SET discount=0,payment_type='debit', total=amount
where customer_id=4;
```

Query OK, 1 rows affected (0.10 sec)
 Rows matched: 1 Changed: 1 Warnings: 0

printBill(cid,bid)
 outputs cost of room for number of days of stay, cost of
 services used within the stay, discount if hotel credit card
 available, total bill to be paid.

Print itemized bill

```
(select 'room price' as
description,rate*(datediff(End_date,Start_date)) as amount from
customer natural join room where customer_id=2)
UNION
(select service_name ,amount from uses where customer_id=2)
UNION
(select 'amount',round(amount,2) from bill where customer_id=2)
UNION
(select 'discount%',coalesce(discount,0) from bill where
customer_id=2)
UNION
(select 'total',round(total,2) from bill where customer_id=2);
```

| description | amount |
|-------------|--------|
| room price | 750 |
| food | 26 |
| pool | 90 |
| amount | 866 |
| discount% | 5 |
| total | 822.7 |

6 rows in set (0.01 sec)

Reports

Report occupancy by city:

```
select b.city,coalesce(booked,0) as booked , total,
       cast((coalesce(booked, 0)/total* 100)as decimal(10,0)) as
       occupancy_percentage from (select city, count(Availability)
       as booked from hotel  natural join room where Availability
       =0 group by city) as a  natural right outer join (select
       city,count(*) as total from hotel natural join room group
       by city)as
b order by b.city;
```

| city | booked | total | occupancy_percentage |
|-------------|--------|-------|----------------------|
| los angeles | 1 | 7 | 14 |
| raleigh | 3 | 10 | 30 |

2 rows in set (0.04 sec)

Report occupancy by hotel:

```
select b.hotel_id,coalesce(booked,0) as booked , total,
       cast((coalesce(booked, 0)/total* 100)as decimal(10,0)) as
       occupancy_percentage from (select
       hotel_id,COUNT(Availability) as booked from hotel natural
       join room where Availability =0 group by hotel_id) as a
       natural right outer join (select hotel_id,count(*) as total
       from hotel natural join room group by hotel_id)as order by
       b.hotel_id;
```

| hotel_id | booked | total | occupancy_percentage |
|----------|--------|-------|----------------------|
| 1 | 3 | 7 | 43 |
| 2 | 0 | 3 | 0 |
| 3 | 0 | 3 | 0 |
| 4 | 1 | 4 | 25 |

4 rows in set (0.00 sec)

Report occupancy by room type:

```
select b.category_name,coalesce(booked,0) as booked , total,
       cast((coalesce(booked, 0)/total* 100)as decimal(10,0)) as
       occupancy_percentage from (select
       category_name,COUNT(Availability) as booked from room
       where Availability =0 group by category_name) as a natural
       right outer join (select category_name,count(*) as total
       from room group by category_name)as b order by
       b.category_name;
```

| category_name | booked | total | occupancy_percentage |
|---------------|--------|-------|----------------------|
| deluxe | 2 | 6 | 33 |
| economy | 0 | 4 | 0 |
| executive | 1 | 2 | 50 |
| presidential | 1 | 5 | 20 |

4 rows in set (0.00 sec)

Report occupancy by date range:

```
select start_date, end_date,count(*) as booked, a.total,
       round(count(*)/a.total*100 ,2) as occupancy_percentage from
       customer,(select count(*) as total from room)as a group by
       start_date ,end_date;
```

| start_date | end_date | booked | total | occupancy_percentage |
|------------|------------|--------|-------|----------------------|
| 2018-02-15 | 2018-02-17 | 1 | 17 | 5.88 |
| 2018-02-24 | 2018-02-28 | 2 | 17 | 11.76 |
| 2018-02-25 | 2018-03-01 | 1 | 17 | 5.88 |
| 2018-03-12 | 2018-03-15 | 1 | 17 | 5.88 |

4 rows in set (0.00 sec)

Total Occupancy

```
select (select count(*) from room where Availability=0) as
rooms_occupied,
(select count(*) from room) as total_rooms;
```

| rooms_occupied | total_rooms |
|----------------|-------------|
| 5 | 10 |

1 row in set (0.00 sec)

percentage of rooms occupied

```
select category_name, coalesce(booked, 0) as booked, total,
cast((coalesce(booked, 0)/total* 100)as decimal(10,0)) as
occupancy_percentage from
(select category_name,COUNT(Availability) as booked from hotel
natural join room where Availability =0 group by
```

| category_name | booked | total | occupancy_percentage |
|---------------|--------|-------|----------------------|
| deluxe | 1 | 3 | 33 |
| economy | 3 | 3 | 100 |
| executive | 1 | 2 | 50 |
| presidential | 0 | 2 | 0 |

4 rows in set (0.01 sec)

staffRole

```
SELECT title,staff_id,name from staff order by title;
```

| title | staff_id | name |
|------------|----------|-------------|
| catering | 2 | Clara |
| catering | 5 | Maria |
| front desk | 9 | Tim |
| front desk | 10 | Kelly |
| manager | 1 | Alex |
| manager | 4 | Simi |
| manager | 6 | Arya |
| service | 3 | Joe |
| service | 7 | Deep |
| service | 8 | John Mathew |
| service | 11 | Sal |

11 rows in set (0.00 sec)

staffAtCustomerService

```

select name, title from serves natural join staff where
    customer_id = 1
UNION
select name, title from staff where staff_id in
(select catering_staff_id from presidential natural join
    customer where customer_id=1
UNION
select service_staff_id from presidential natural join customer
    where customer_id=1);

```

| name | title |
|-------------|----------|
| John Mathew | service |
| Clara | catering |
| Joe | service |

3 rows in set (0.01 sec)

revenue

```

select round(sum(total),2) from bill where customer_id in(select
customer_id from customer where hotel_id=1)
AND bill_date between '2017-05-30' and '2018-03-18';

```

| round(sum(total),2) |
|---------------------|
| 822.70 |

1 row in set (0.00 sec)

6. EXPLAIN, Indexes for two queries

Update bills:

```

UPDATE bill
SET bill.amount= (select rate*(datediff(End_date,Start_date))
from customer natural join room where
customer_id=2)+(select coalesce(sum(amount),0) from uses where
customer_id=2), bill_date=(select end_date
from customer where customer_id=2)
WHERE customer_id =2;

```

Explain output:

```

sql> EXPLAIN UPDATE bill
-> SET bill.amount= (select rate*(datediff(End_date,Start_date))
from customer natural join room where

```

```

-> customer_id=2)+(select coalesce(sum(amount),0) from uses where
customer_id=2), bill_date=(select end_date
-> from customer where customer_id=2)
-> WHERE customer_id =2;

```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|----------|------------|-------|--------------------------|------------------|---------|-------------|------|----------|-------------|
| 1 | UPDATE | bill | NULL | range | bill_customer_fk | bill_customer_fk | 4 | const | 1 | 100.00 | Using where |
| 4 | SUBQUERY | customer | NULL | const | PRIMARY | PRIMARY | 4 | const | 1 | 100.00 | NULL |
| 3 | SUBQUERY | uses | NULL | ALL | usage_customer_id | NULL | NULL | NULL | 5 | 100.00 | Using where |
| 2 | SUBQUERY | customer | NULL | const | PRIMARY,customer_room_fk | PRIMARY | 4 | const | 1 | 100.00 | NULL |
| 2 | SUBQUERY | room | NULL | const | PRIMARY,room_hotel_fk | PRIMARY | 8 | const,const | 1 | 100.00 | NULL |

5 rows in set (0.10 sec)

Since the full scan happened on the uses table, therefore we will create index on the uses table;

Query for indexing:-

```
CREATE INDEX amt_index ON uses(amount);
```

Now the same **EXPLAIN** query is run again:

```
sql> EXPLAIN UPDATE bill
```

```

-> SET bill.amount= (select rate*(datediff(End_date,Start_date))
from customer natural join room where
-> customer_id=2)+(select coalesce(sum(amount),0) from uses where
customer_id=2), bill_date=(select end_date
-> from customer where customer_id=2)
-> WHERE customer_id =2;

```

EXPLAIN output after indexing:

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|----------|------------|-------|--------------------------|------------------|---------|-------------|------|----------|-------------|
| 1 | UPDATE | bill | NULL | range | bill_customer_fk | bill_customer_fk | 4 | const | 1 | 100.00 | Using where |
| 4 | SUBQUERY | customer | NULL | const | PRIMARY | PRIMARY | 4 | const | 1 | 100.00 | NULL |
| 3 | SUBQUERY | uses | NULL | ref | cust_index | cust_index | 4 | const | 5 | 100.00 | NULL |
| 2 | SUBQUERY | customer | NULL | const | PRIMARY,customer_room_fk | PRIMARY | 4 | const | 1 | 100.00 | NULL |
| 2 | SUBQUERY | room | NULL | const | PRIMARY,room_hotel_fk | PRIMARY | 8 | const,const | 1 | 100.00 | NULL |

Now as seen above, the uses table is now being accessed by type “ref” which means the rows are now being accessed by indexes.

Another Index:

Query Used:

```

select start_date, end_date,count(*) as booked, a.total1 as total,
(count(*)/a.total1*100) as occupancy_percentage from customer,

```

```
(select count(*) as total1 from room) as a group by start_date
,end_date;
```

Explain Query:

```
sql> EXPLAIN select start_date, end_date, count(*) as booked, a.total1
as total,
-> (count(*)/a.total1*100) as occupancy_percentage from customer,
-> (select count(*) as total1 from room) as a group by start_date
,end_date;
```

Explain output before indexing:

Query for indexing:

```
mysql> CREATE INDEX date_index on CUSTOMER(start_date,end_date);
```

Now the same EXPLAIN query is run again:

```
sql> EXPLAIN select start_date, end_date, count(*) as booked, a.total1
as total,
-> (count(*)/a.total1*100) as occupancy_percentage from customer,
-> (select count(*) as total1 from room) as a group by start_date
,end_date;
```

EXPLAIN query after indexing:

As seen, earlier the table Customer was not being accessed using any index but now it is being accessed by using the index date_index.

4.3 Query correctness proofs

```
select name, title from serves natural join staff where
customer_id = 1
UNION
select
name, title from staff where staff_id in
(select
catering_staff_id from presidential natural join customer where
customer_id=1
UNION ALL
```

```
select service_staff_id from presidential
natural join customer where customer_id=1);
```

$$\Pi_{\text{name, title}}(\sigma_{\text{customer_id}=1}(\text{services} \bowtie \text{staff})) \cup \Pi_{\text{name, title}}(\sigma_{\text{staff_id}} \text{staff} \text{ IN } (\Pi_{\text{catering_staff_id}}(\sigma_{\text{customer_id}=1}(\text{presidential} \bowtie \text{customer})))) \cup \Pi_{\text{service_staff_id}}(\sigma_{\text{customer_id}=1}(\text{presidential} \bowtie \text{customer})))$$

We basically have two subparts of this query and the second subquery further has two parts. In the first subpart we select the tuples from the result which we get after doing natural join of serves and staff. Then we put projection over the result with attributes name and title.

In the second subpart, firstly we find presidential natural join customer then we put selection over it on the condition $\text{customer_id} = 1$. After that we take projection on that result once for the attribute service_staff_id and second time for the catering_staff_id .

After that we get those tuples of staff which have staff id in the above calculated result then we union the results.

2.

```
(select 'room price' as description, rate*(datediff(End_date, Start_date)) as amount from
customer natural join room where customer_id=2)
```

```
UNION
```

```
(select service_name , amount from uses where customer_id=2)
```

```
UNION
```

```
(select 'amount', round(amount, 2) from bill where customer_id=2)
```

```
UNION
```

```
(select 'discount%', coalesce(discount, 0) from bill where customer_id=2)
```

```
UNION
```

```
(select 'total', round(total, 2) from bill where customer_id=2);
```

$$\rho_{(\text{description}, \text{amount})} (\pi_{(\text{roomprice}, \text{rate} * (\text{end_date} - \text{start_date}))}(\sigma_{(\text{customer_id}=2)}((\text{customer}) \bowtie (\text{room}))) \cup (\pi_{(\text{service_name}, \text{amount})}(\sigma_{(\text{customer_id}=2)}((\text{customer}))) \cup (\pi_{(\text{"amount"}, \text{amount})}(\sigma_{(\text{customer_id}=2)}((\text{bill}))) \cup (\pi_{(\text{"discount"}, \text{discount})}(\sigma_{(\text{customer_id}=2)}((\text{bill}))) \cup (\pi_{(\text{"total"}, \text{total})}(\sigma_{(\text{customer_id}=2)}((\text{bill})))$$

Explanation :

The first select query natural joins the tuples of customer and room table selects the tuple from that join where customer id = 2. It then finds the room price by the operation $\text{rate} * (\text{diff between the end date and the start date})$. This brings out one tuple in the final

output tuple.

The second select query selects the tuple from the table uses where customer_id = 2 and then projects on the service name, amount column from that and contributes to the final output table.

The third select query selects the tuple from the table bill where customer_id = 2 and makes amount in the output table.

The fourth select query selects the tuple from the table bill where customer_id = 2, finds discount from the table by coalescing discount and 0 and puts the discount and its value in the output table.

The fifth select query selects the tuple from bill from the table bill where customer_id = 2 , finds total bill amount and puts the tuple in the final output.