**1 Implementation of Database**

**1.1 Creation of Database with SQL Statements**

For this part, we are going to use SQL for creating tables.

CREATE TABLE EMPLOYEE (

employee\_id decimal(9, 0) not null,

name varchar(40) not null,

age int check(age > 0)

city varchar(20),

state varchar(20),

zip\_code varchar(10),

street\_number varchar(10),

salary\_rate double,

job\_type varchar(20),

PRIMARY KEY (employee\_id)

);

CREATE TABLE EVENT\_STAFF (

employee\_id decimal(9, 0) not null,

on\_call\_number decimal(4, 0),

PRIMARY KEY (employee\_id)

);

CREATE TABLE MANAGER (

employee\_id decimal(9, 0) not null,

title varchar(20),

PRIMARY KEY (employee\_id)

);

CREATE TABLE DINING\_STAFF (

employee\_id decimal(9, 0) not null,

shift varchar(20),

dining\_type varchar(20),

PRIMARY KEY (employee\_id)

);

CREATE TABLE TECH\_SUPPORT (

employee\_id decimal(9, 0) not null,

PRIMARY KEY (employee\_id)

);

CREATE TABLE ACCOUNTANT (

employee\_id decimal(9, 0) not null,

PRIMARY KEY (employee\_id)

);

CREATE TABLE CONCIERGE (

employee\_id decimal(9, 0) not null,

year\_of\_experience int check(year\_of\_experience >= 0 and year\_of\_experience <= 50),

PRIMARY KEY (employee\_id)

);

CREATE TABLE RECEPTIONIST (

employee\_id decimal(9, 0) not null,

language varchar(20),

PRIMARY KEY (employee\_id)

);

CREATE TABLE HOUSEKEEPER (

employee\_id decimal(9, 0) not null,

year\_of\_experience int check(year\_of\_experience >= 0 and year\_of\_experience <= 50),

PRIMARY KEY (employee\_id)

);

CREATE TABLE TECH\_SUPPORT\_LICENSE (

employee\_id decimal(9, 0) not null,

license varchar(60) not null,

PRIMARY KEY (employee\_id, license)

);

CREATE TABLE ACCOUNTANT\_LICENSE (

employee\_id decimal(9, 0) not null,

license varchar(60) not null,

PRIMARY KEY (employee\_id, license)

);

CREATE TABLE CLEAN (

employee\_id: integer = 9 digit

room\_number: integer = 4 digit

time: HH:MM:SS, sting = 8 chars

date: MM/DD/YYYY, string = 10 chars

employee\_id decimal(9, 0) not null,

room\_number decimal(4, 0) not null,

time\_stamp timestamp not null,

PRIMARY KEY (employee\_id, room\_number, time\_stamp)

);

CREATE TABLE ROOM (

room\_number decimal(4, 0) not null,

bed\_type varchar(20),

room\_type varchar(20),

per\_night\_price double not null,

PRIMARY KEY (room\_number)

);

CREATE TABLE INDIVIDUAL\_CUSTOMER (  
 client\_id decimal(6, 0) not null,

name varchar(40) not null,

sex varchar(10),

date\_of\_birth date,

PRIMARY KEY (client\_id)

);

CREATE TABLE ORGANIZATION (

org\_id decimal(6, 0) not null,

name varchar(60) not null,

PRIMARY KEY (org\_id)

);

CREATE TABLE CHECK\_IN (

check\_in\_id int not null,

employee\_id decimal(9, 0),

client\_id decimal(6, 0) not null,

room\_number decimal(4, 0) not null,

length\_of\_stay int check(length\_of\_stay > 0),

time\_stamp timestamp,

key\_type varchar(10),

lounge\_access varchar(3),

bill\_amount double,

PRIMARY KEY (check\_in\_id)

);

CREATE TABLE CHECK\_OUT (

check\_in\_id int not null,

time\_stamp timestamp not null,

PRIMARY KEY (check\_in\_id)

);

CREATE TABLE PAY\_BILL (

check \_in\_id int not null,

time\_stamp timestamp not null,

amount double,

PRIMARY KEY (check\_in\_id, time\_stamp)

);

CREATE TABLE PHONE (

client\_id decimal(6, 0) not null,

phone char(12) check(phone LIKE ),

PRIMARY KEY (client\_id, phone)

);

CREATE TABLE MEMBERSHIP (

membership\_number char(10) not null,

client\_id decimal(6, 0) not null,

PRIMARY KEY (membership\_number, client\_id)

);

CREATE TABLE BILL\_ACOUNT (

org\_id decimal(6, 0) not null,

bank varchar(20),

account\_number varchar(20) not null,

PRIMARY KEY (org\_id)

);

CREATE TABLE EVENT (  
 event\_id decimal(4, 0) not null,

name varchar(40) not null,

time\_stamp timestamp,

duration int,

employee\_id decimal(9, 0),

PRIMARY KEY (event\_id)

);

CREATE TABLE EVENT\_BILL (

bill\_id decimal(6, 0) not null,

event\_id decimal(4, 0),

date date,

amount double,

PRIMARY KEY (bill\_id)

);

CREATE TABLE PREPARE\_BILL (  
 employee\_id decimal(9, 0) not null,

bill\_id decimal(6, 0) not null,

PRIMARY KEY (emplotee\_id, bill\_id)

);

CREATE TABLE MANAGE\_EVENT (

employee\_id decimal(9, 0) not null,

event\_id decimal(4, 0) not null,

PRIMARY KEY (employee\_id, event\_id)

);

CREATE TABLE PAY\_EVENT\_BILL (

bill\_id decimal(6, 0) not null,

org\_id decimal(6, 0) not null,

time\_stamp timestamp not null,

amount double

type varchar(10),

PRIMARY KEY (bill\_id, org\_id, time\_stamp)

);

CREATE TABLE ORGANIZATION\_HOLD\_EVENT (  
 event\_id decimal(4, 0) not null,

org\_id decimal(6, 0) not null,

PRIMARY KEY (event\_id, org\_id)

);

CREATE TABLE SERVE\_EVENT (  
 employee\_id decimal(9, 0) not null,

event\_id decimal(4, 0) not null,

PRIMARY KEY (employee\_id, event\_id)

);

**2 Creation of Views**

1. Available rooms: show the available rooms in the hotel.

CREATE VIEW Available\_room AS

(

SELECT ci.room\_number

FROM

(

SELECT room\_number, max(time\_stamp) AS latest\_time

FROM CHECK\_IN

GROUP BY room\_number

) ci

INNER JOIN CHECK\_OUT co

ON ci.check\_in\_id = co.check\_in\_id

)

UNION

(

(

SELECT room\_number

FROM ROOM

)

EXCEPT

(

SELECT room\_number

FROM CHECK\_IN

)

)

2. Popular event manager: show the popular event managers who have helped organize more than 10 events in **this** month.

CREATE VIEW Popular\_event\_manager AS

SELECT DISTINCT employee\_id, count(\*) AS num\_of\_event

FROM MANAGE\_EVENT me, EVENT e

WHERE me.event\_id = e.event\_id AND month(e.time\_stamp) = month(now())

GROUP BY employee\_id

HAVING num\_of\_event > 10

The event managers who have helped organize more than 10 events in **one** month.

CREATE VIEW Popular\_event\_manager\_general AS

SELECT DISTINCT employee\_id

FROM MANAGE\_EVENT me, EVENT e

WHERE me.event\_id = e.event\_id

GROUP BY employee\_id, year(time\_stamp), month(time\_stamp)

HAVING count(\*) > 10

3. Frequent customers: show the individual customers who checked in at least 10 times this year.

CREATE VIEW Frequent\_customers AS  
SELECT client\_id, count(\*) AS num\_of\_check\_in

FROM CHECK\_IN

WHERE year(time\_stamp) = year(now())

GROUP BY client\_id

HAVING num\_of\_check\_in >= 10

4. Popular rooms: show the rooms that were checked in at least 30 times this year.

CREATE VIEW Popular\_rooms AS

SELECT room\_number, count(\*) AS num\_of\_check\_in

FROM CHECK\_IN

WHERE year(time\_stamp) = year(now())

GROUP BY room\_number

HAVING num\_of\_check\_in >= 30

**3 Creation of Queries**

1. Retrieve the number of employees who work at the lounge/bar.

SELECT employee\_id

FROM DINNING\_STAFF

WHERE dining\_type = 'lounge/bar'

2. Retrieve the average salary of the receptionists.

SELECT avg(salary\_rate)

FROM EMPOYEE e, RECEPTIONIST r

WHERE e.employee\_id = r.employee\_id

3. Retrieve the information of individual customers who have been billed more than $1,000 in total this year.

SELECT c.client\_id, c.name, c.sex, c.date\_of\_birth

FROM INDIVIDUAL\_CUSTOMER c, CHECK\_IN ci

WHERE c.client\_id = ci.client\_id AND year(co.time\_stamp) = year(now())

GROUP BY c.client\_id

HAVING sum(ci.bill\_amount) > 1000

4. For each individual, retrieve his/her bill amount in ascending order of each check-in date.

SELECT c.client, ci.bill\_amount, ci.time\_stamp

FROM INDIVIDUAL\_CUSTOMER c, CHECK\_IN ci

WHERE c.client\_id = ci.client\_id

ORDER BY c.client\_id, ci.time\_stamp ASC

5. Retrieve the information of the frequent customers who have stayed for at least 15 nights this year.

SELECT c.client\_id, c.name, c.sex, c.date\_of\_birth

FROM Frequent\_customers c, CHECK\_IN ci

WHERE c.client\_id = ci.client\_id AND year(ci.time\_stamp) = year(now())

GROUP BY c.client\_id, c.name, c.sex, c.date\_of\_birth

HAVING sum(length\_of\_stay) >= 15

6. Retrieve the average age of individual customers who were helped by a receptionist who only speaks Spanish.

SELECT avg(e.age)

FROM EMPLOYEE e, RECEPTIONIST r, LANGUAGE l

WHERE e.employee\_id = r.employee\_id AND r.employee\_id = l.employee\_id AND r.language = 'Spanish'

7. Retrieve the information of the organization that organized at least two events and got bills of over $2000 in total.

SELECT o.org\_id, o.name

FROM ORGANIZATION o, ORGANIZATION\_HOLD\_EVENT e, EVENT\_BILL b

WHERE o.org\_id = e.org\_id AND e.event\_id = b.event\_id

GROUP BY o.org\_id, o.name

HAVING count(\*) >= 2 AND sum(b.amount) > 2000

8. Retrieve the highest amount of bill of the events helped by the most popular event manager.

SELECT max(b.amount)

FROM

(SELECT employee\_id FROM Popular\_event\_manager WHERE num\_of\_event = max(num\_of\_event)) e, EVENT\_BILL b, MANAGE\_EVENT me

WHERE e.employee\_id = me.employee\_id AND me.event\_id = b.event\_id

9. Retrieve information of the event that each of its organizers pays the highest amount for the event (suppose organizers of the same event pay the bill evenly).

SELECT e.event\_id, e.name, e.time\_stamp, e.duration

FROM EVENT e,

(

SELECT b.event\_id, b.amount / count(\*) AS avg\_bill\_of\_org

FROM ORGANIZATION\_HOLD\_EVENT he, EVENT\_BILL b

WHERE he.event\_id = b.event\_id

GROUP BY b.event\_id

) a

WHERE e.event\_id = a.event\_id AND a.avg\_bill\_of\_org = max(a.avg\_bill\_of\_org)

10. Retrieve the date and time the most popular room was last checked in.

SELECT max(ci.time\_stamp)

FROM CHECK\_IN ci, Popular\_rooms r

WHERE ci.room\_number = r.room\_number AND r.num\_of\_check\_in = max(num\_of\_check\_in)