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)
```