Hotel Booking System

HD ASSIGNMENT-32606 DATABASE-SPRING 2023

Student number: Man Sing Lee

Student number: 24708772

Hotel Booking System Database

The database models a comprehensive Hospitality Booking System encompassing customer management, reservation tracking, and room assignment.

Entities:

Customers: Individuals booking accommodations.
Reservations: Customer bookings, including details like check-in/out dates, purpose, and payment.
Payment: Records of financial transactions related to reservations.
Hotel Rooms: Information about available rooms, including type, status, and price.
Employees: Staff details involved in room maintenance and cleaning.
Housekeeping (HSK): Assignment of employees to clean specific rooms.
Group Reservations: Special reservations for groups with unique details.

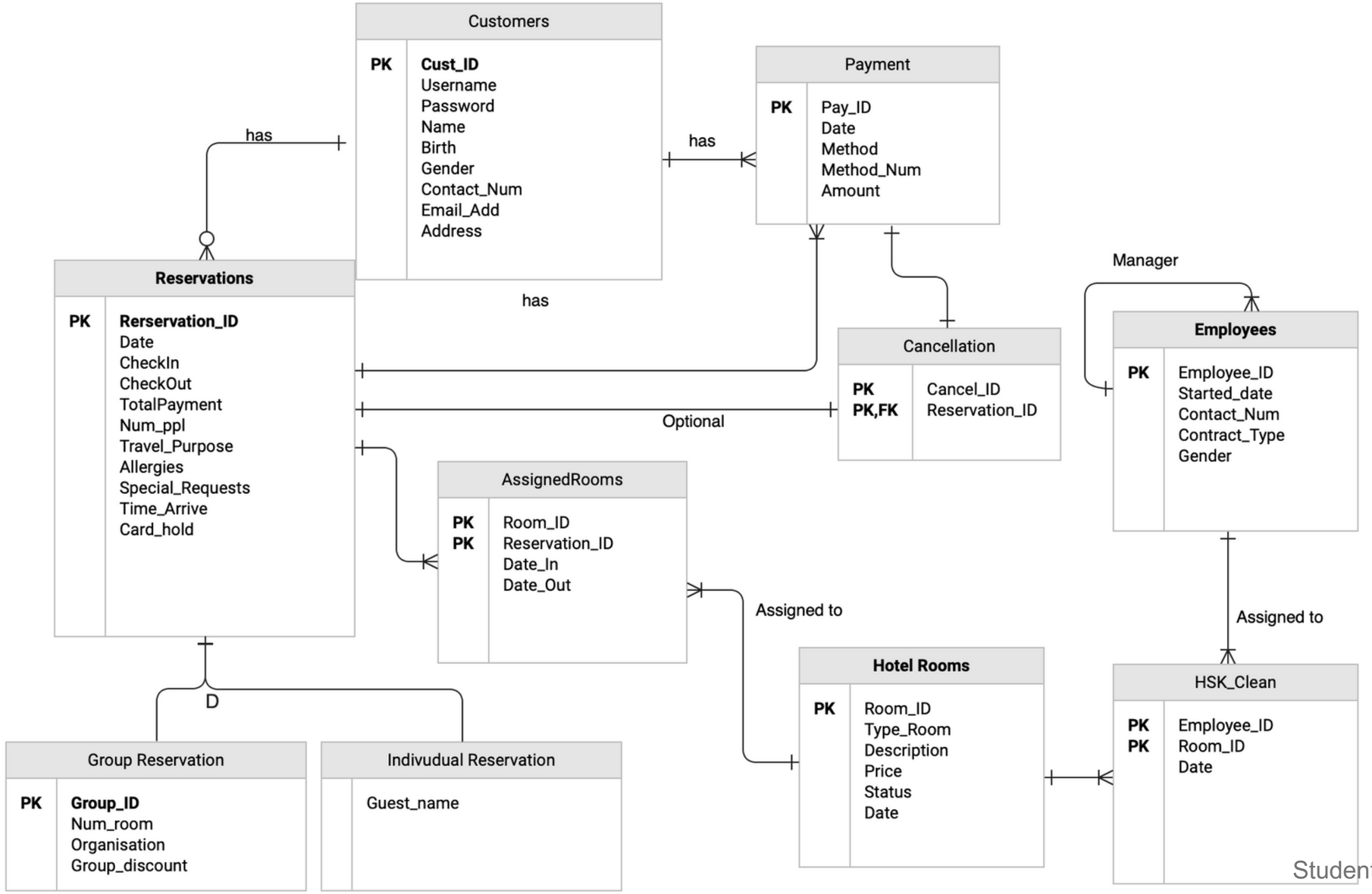
Relationships between entities enable tracking customer reservations, employee assignments, and payment details. The system manages both individual and group reservations seamlessly.

Purpose:

Efficiently manages the end-to-end process of hospitality services. Provides insights into room availability, customer preferences, and financial transactions.

Student number: Man Sing Lee
Student number: 24708772

Hotel Booking System ERD



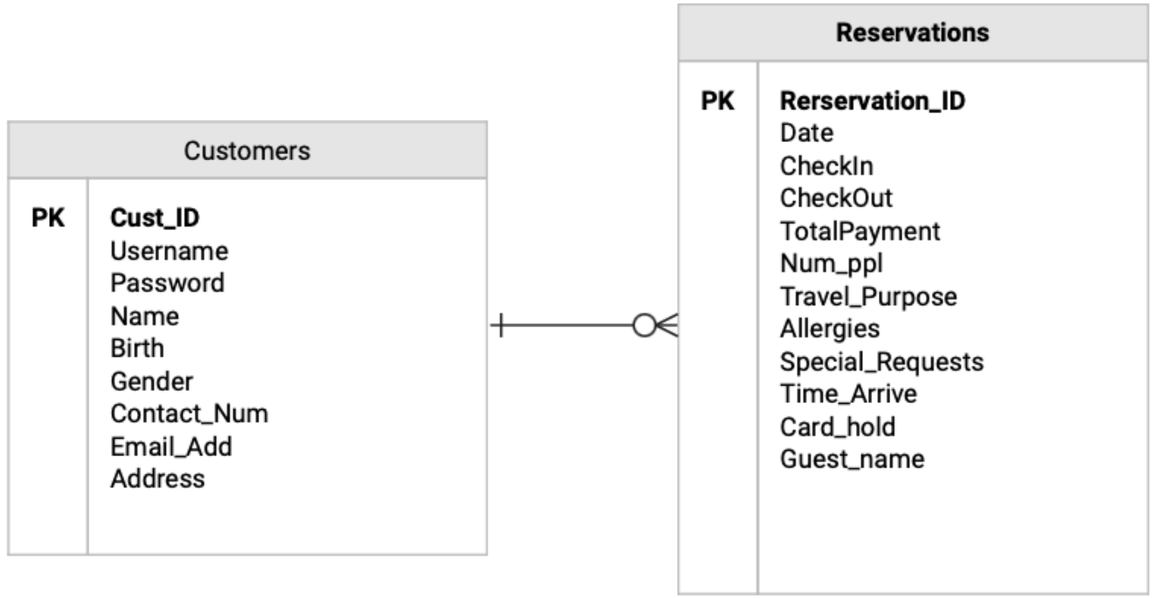
Student number: Man Sing Lee
Student number: 24708772

A Single One-To-Many Relationship:

One Customer can have many Reservations.

```
select customers.cust_id, reservations.reservation_id from customers,
reservations  where customers.cust_id=reservations.cust_id;
```

```
 cust_id | reservation_id
----------+----------------
 H5134789 | R123456
 H5134799 | R789012
 H5134800 | R345678
 H5134890 | R901234
 H5134789 | R567890
```



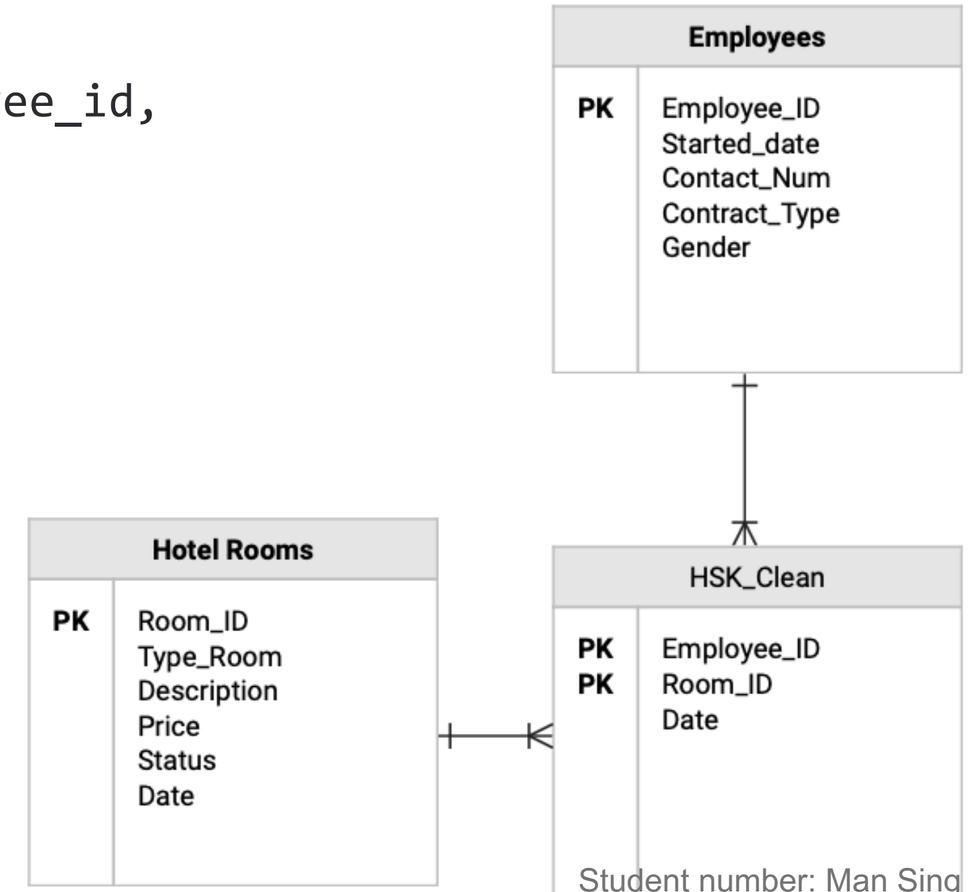
Student number: Man Sing Lee
Student number: 24708772

A Single Many-To-Many Relationship:

Many employees can be assigned to clean many rooms; many employees can clean many rooms.
For example, Room001 was cleaned by employees A and B. Also, employee A was responsible for cleaning Room001, Room002, and Room003.

```
select distinct hotelrooms.room_id, hsk_clean.employee_id,
hsk_clean.date from hotelrooms, hsk_clean where
hotelrooms.room_id=hsk_clean.room_id order by 1;
```

room_id	employee_id	date
R001	EMP001	2023-09-01
R001	EMP005	2023-09-01
R002	EMP001	2023-09-02
R002	EMP002	2023-09-02
R003	EMP002	2023-09-05
R003	EMP003	2023-09-05
R004	EMP003	2023-09-08
R004	EMP004	2023-09-08
R005	EMP004	2023-09-10



Student number: Man Sing Lee
Student number: 24708772

A simple query of a single table.

List all Customer's Names, Customer's ID, Numbers of people, and Special requests whose travel purpose is a family holiday.

```
select guest_name, num_ppl,cust_id, special_request, travel_purpose from
reservations where travel_purpose= 'Family Holiday';
```

```
postgres=# select guest_name, num_ppl,cust_id, special_requests,travel_purpose
 from reservations where travel_purpose= 'Family holiday';
 guest_name | num_ppl | cust_id  |       special_requests       | travel_purpo
se
------------+---------+----------+------------------------------+------------
---
 Bryan Li   |       2 | H5134789 |                              | Family holid
ay
 Lily Kim   |       4 | H5134800 | Adjoining rooms, if possible | Family holid
ay
```

Student number: Man Sing Lee
Student number: 24708772

A query that uses the words "natural join".

List customer with customer ID, customer name, reservation ID, Payment ID and amount for the payment amount more or equal to 400, order by the amount.

```
select cust_id,name, reservation_id, pay_id,amount from customers natural
join payment where amount >=400 order by amount;
```

```
postgres=# select cust_id,name, reservation_id, pay_id,amount from customers n
atural join payment where amount >=400 order by amount;
 cust_id  |   name    | reservation_id | pay_id  | amount
----------+-----------+----------------+---------+--------
 H5134799 | Lyns Chu | R789012        | P789012 |    400
 H5134800 | Lily Kim | R345678        | P345678 |    500
 H5134789 | Bryan Li | R567890        | P567890 |    600
(3 rows)
```

Student number: Man Sing Lee
Student number: 24708772

A query involving a “Group by”, perhaps also with a “HAVING”.

List Reservation ID and Numbers of people which have the maximum number of days staying in the hotel, order by number of people.

```
SELECT r.reservation_id, r.num_ppl, MAX(ar.dateout - ar.datein) AS
days from reservations r,assignedrooms ar where r.reservation_id =
ar.reservation_id GROUP BY r.reservation_id,r.num_ppl HAVING MAX(ar.dateout
- ar.datein) > 5 order by num_ppl;
```

```
postgres=# SELECT r.reservation_id, r.num_ppl, MAX(ar.dateout - ar.datein) AS
days from reservations r,assignedrooms ar where r.reservation_id = ar.reservat
ion_id GROUP BY r.reservation_id,r.num_ppl HAVING MAX(ar.dateout - ar.datein)
> 5 order by num_ppl;
 reservation_id | num_ppl | days
----------------+---------+------
 R123456        |       2 |    9
 R345678        |       4 |    9
(2 rows)
```

Student number: Man Sing Lee
Student number: 24708772

A cross product which cannot be implemented using the words “natural join” (e.g. self join)

List the manager of each employees.

```
select e1. employee_id as employees, e1.name as name, e2.employee_id as manager,
e2.name as managername from employees e1, employees e2 where e1.manager_id=
e2.employee_id;
```

```
postgres=# select e1. employee_id as employees, e1.name as name, e2.employee_i
d as manager, e2.name as managername from employees e1, employees e2 where e1.
manager_id= e2.employee_id;
 employees |      name       | manager | managername
-----------+-----------------+---------+-------------
 EMP002    | Jane Smith      | EMP001  | John Doe
 EMP003    | Bob Johnson     | EMP001  | John Doe
 EMP004    | Alice Brown     | EMP002  | Jane Smith
 EMP005    | Charlie Wilson  | EMP002  | Jane Smith
(4 rows)
```

Student number: Man Sing Lee
Student number: 24708772

Check Statements:

```
-- Create "customers" table
CREATE TABLE customers (
    cust_ID      varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    username     varchar(20),
    password     varchar(255),
    name         varchar(50) NOT NULL,
    birth        DATE,
    gender       char(1) CHECK (gender IN ('M', 'F')),
    contact_num  varchar(15),
    email_add    varchar(50) NOT NULL,
    home_address TEXT
);
```

```
-- Create "reservations" table
CREATE TABLE reservations (
    reservation_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    checkin    date CHECK (checkin >= date),
    checkout    date CHECK (checkout >= checkin),
    total_payment   integer,
    num_ppl integer,
    travel_purpose      TEXT,
    special_requests    TEXT,
    time_arrive  integer,
    card_hold_type  char(1) CHECK (card_hold_type IN ('M','V')),
    cust_id varchar(20) REFERENCES customers(cust_id),
    guest_name       varchar(50) -- Guest name for individual reservations
);
```

```
-- Create "payment" table
CREATE TABLE payment (
    pay_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    method    char(2) CHECK (method IN ('CC','DB','GC','PP')),
    method_num    integer,
    amount   integer,
    reservation_id varchar(20) REFERENCES reservations(reservation_id)
    cust_id varchar(20) REFERENCES customers(cust_ID)
);
```

```
-- Create "assignedrooms" table
CREATE TABLE assignedrooms (
    room_ID  varchar(10) REFERENCES hotelrooms(room_id),
    reservation_ID varchar(20) REFERENCES reservations(reservation_id),
    datein date,
    dateout date CHECK (dateout > datein),
    PRIMARY KEY (room_ID, reservation_ID)
);
```

Student number: Man Sing Lee
Student number: 24708772

Check Statements:

```
-- Create "customers" table
CREATE TABLE customers (
    cust_ID      varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    username     varchar(20),
    password     varchar(255),
    name         varchar(50) NOT NULL,
    birth        DATE,
    gender       char(1) CHECK (gender IN ('M', 'F')),
    contact_num  varchar(15),
    email_add    varchar(50) NOT NULL,
    home_address TEXT
);

-- Create "reservations" table
CREATE TABLE reservations (
    reservation_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    checkin    date CHECK (checkin >= date),
    checkout    date CHECK (checkout >= checkin),
    total_payment   integer,
    num_ppl integer,
    travel_purpose      TEXT,
    special_requests    TEXT,
    time_arrive  integer,
    card_hold_type  char(1) CHECK (card_hold_type IN ('M','V')),
    cust_id varchar(20) REFERENCES customers(cust_id),
    guest_name       varchar(50) -- Guest name for individual reservations
);
```

Student number: Man Sing Lee
Student number: 24708772

Student number: Man Sing Lee
Student number: 24708772

Check Statements:

```
-- Create "payment" table
CREATE TABLE payment (
    pay_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    method    char(2) CHECK (method IN ('CC','DB','GC','PP')),
    method_num    integer,
    amount   integer,
    reservation_id varchar(20) REFERENCES reservations(reservation_id),
    cust_id varchar(20) REFERENCES customers(cust_ID)
);


-- Create "assignedrooms" table
CREATE TABLE assignedrooms (
    room_ID  varchar(10) REFERENCES hotelrooms(room_id),
    reservation_ID varchar(20) REFERENCES reservations(reservation_id),
    datein date,
    dateout date CHECK (dateout > datein),
    PRIMARY KEY (room_ID, reservation_ID)
);
```

Student number: Man Sing Lee
Student number: 24708772

Check Statements:

```
-- Create "payment" table
CREATE TABLE payment (
    pay_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    method    char(2) CHECK (method IN ('CC','DB','GC','PP')),
    method_num    integer,
    amount   integer,
    reservation_id varchar(20) REFERENCES reservations(reservation_id),
    cust_id varchar(20) REFERENCES customers(cust_ID)
);


-- Create "assignedrooms" table
CREATE TABLE assignedrooms (
    room_ID  varchar(10) REFERENCES hotelrooms(room_id),
    reservation_ID varchar(20) REFERENCES reservations(reservation_id),
    datein date,
    dateout date CHECK (dateout > datein),
    PRIMARY KEY (room_ID, reservation_ID)
);
```

Student number: Man Sing Lee
Student number: 24708772

ON DELETE RESTRICT:

```
CREATE TABLE reservations (
    reservation_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    checkin    date CHECK (checkin >= date),
    checkout    date CHECK (checkout >= checkin),
    total_payment   integer,
    num_ppl integer,
    travel_purpose      TEXT,
    special_requests    TEXT,
    time_arrive  integer,
    card_hold_type  char(1) CHECK (card_hold_type IN ('M','V')),
    cust_id varchar(20) REFERENCES customers(cust_id) ON DELETE RESTRICT,
    guest_name       varchar(50) -- Guest name for individual reservations
);
```

```
CREATE TABLE payment (
    pay_ID  varchar(20) NOT NULL UNIQUE PRIMARY KEY,
    date     date,
    method    char(2) CHECK (method IN ('CC','DB','GC','PP')),
    method_num    integer,
    amount   integer,
    reservation_id varchar(20) REFERENCES reservations(reservation_id),
    cust_id varchar(20) REFERENCES customers(cust_ID) ON DELETE RESTRICT
);
```

Student number: Man Sing Lee
Student number: 24708772

ON DELETE CASCADE:

```
-- Create "assignedrooms" table
CREATE TABLE assignedrooms (
    room_ID  varchar(10) REFERENCES hotelrooms(room_id) ON DELETE CASCADE,
    reservation_ID varchar(20) REFERENCES reservations(reservation_id) ON DELETE CASCADE,
    datein date,
    dateout date CHECK (dateout > datein),
    PRIMARY KEY (room_ID, reservation_ID)
);
```

Student number: Man Sing Lee
Student number: 24708772

CREATE VIEW :

1. **reservation_info: View of the reservation information which including reservation ID, check in and check out date, customer ID, customer name and amount paid.**

```
postgres=# CREATE VIEW reservation_info AS
SELECT r.reservation_ID,r.date,r.checkin,r.checkout,c.cust_ID, c.name AS custo
mer_name,p.amount from reservations r, payment p, customers c where r.cust_id
= c.cust_ID and  r.reservation_ID = p.reservation_id order by r.reservation_id
;
```

2. housekeeping_assignments: View of the housekeeping room cleaning assignments, including employee ID, room type and room status.

```
postgres=# CREATE VIEW housekeeping_assignments AS SELECT hsk.employee_ID,hsk.
room_ID,hsk.date,e.name AS employee_name,hr.type_room,hr.status AS room_status
 from HSK_Clean hsk, employees e, hotelrooms hr where hsk.employee_ID = e.empl
oyee_ID and hsk.room_ID = hr.room_ID;
```

Student number: Man Sing Lee
Student number: 24708772

Thank You!