Dunn Hotel Database Design Proposal

Taylor Dunn

Table of Contents

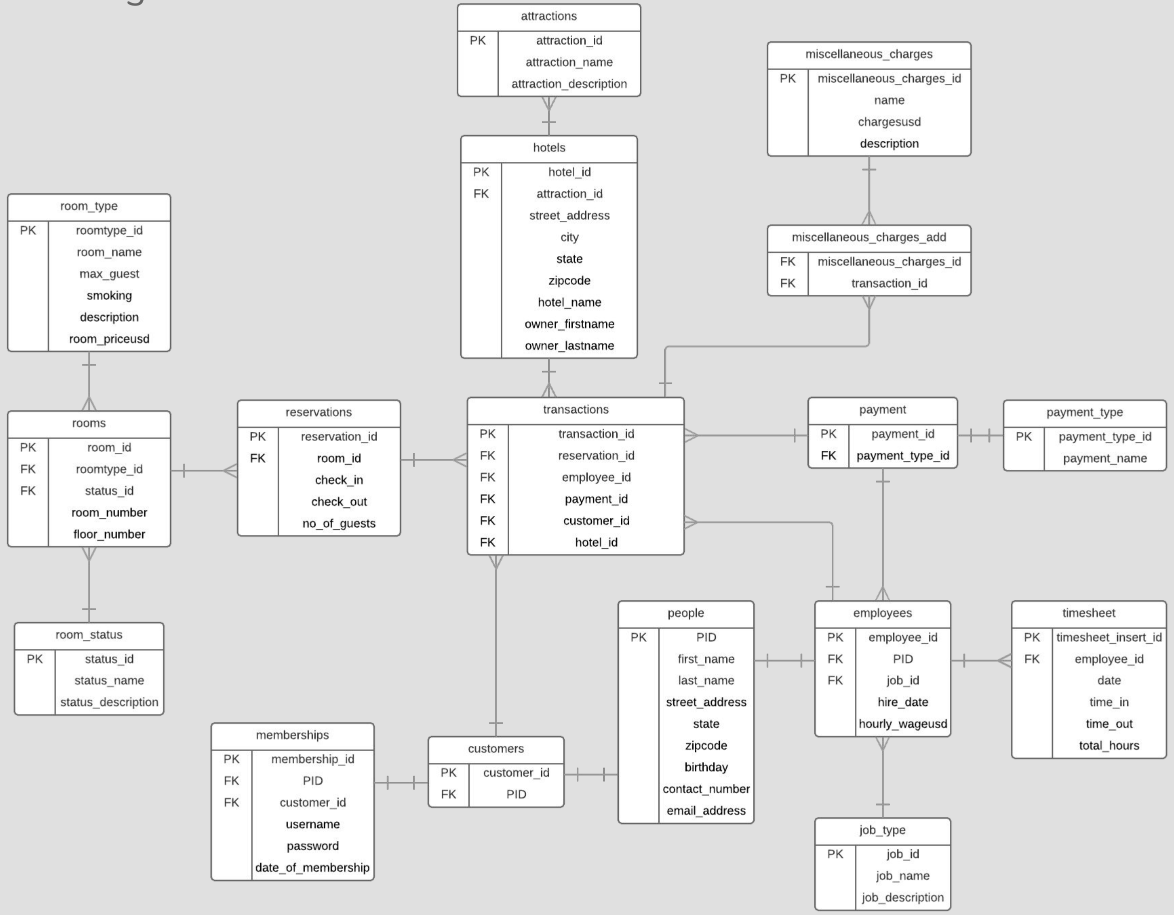
Executive Summary

This database has been created for the Dunn Hotel, a hotel that is run by Taylor Dunn and her minions. It has been created to keep track of all records needed to ensure the success of the hotel.

The information that follows is an intensive review of the database itself, and aspects of it’s uses. There are numerous parts of this review including the ER Diagram, create statements for tables, and the sample data that was inserted into the table. Next are the results of queries, views, stored procedures, reports, and triggers. These were all created and then tested.

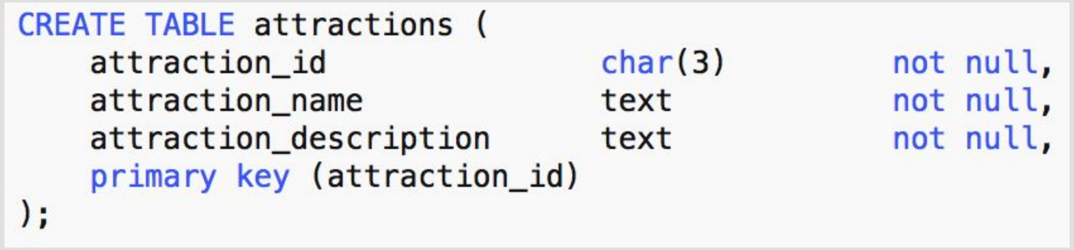
The purpose of this database is to condense all of the information that the hotel needs to function into one central, and organized collection of tables.

ER Diagram:

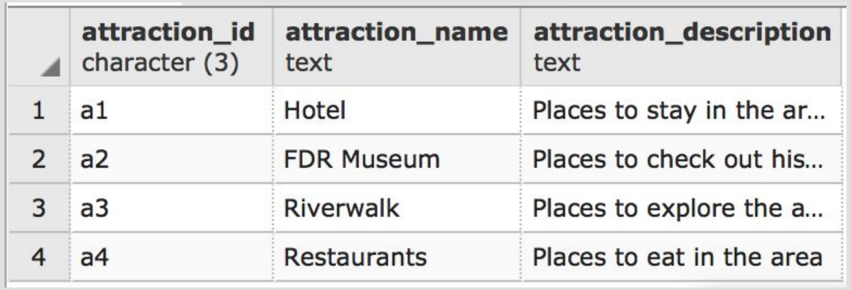

attractions
PK attraction_id
attraction_name
attraction_description
miscellaneous_charges
PK miscellaneous_charges_id
name
chargesusd
description
hotels
PK hotel_id
FK attraction_id
street_address
city
state
zipcode
hotel_name
owner_firstname
owner_lastname
room_type
PK roomtype_id
room_name
max_guest
smoking
description
room_priceusd
miscellaneous_charges_add
FK miscellaneous_charges_id
FK transaction_id
rooms
PK room_id
FK roomtype_id
FK status_id
room_number
floor_number
reservations
PK reservation_id
FK room_id
check_in
check_out
no_of_guests
transactions
PK transaction_id
FK reservation_id
FK employee_id
FK payment_id
FK customer_id
FK hotel_id
payment
PK payment_id
FK payment_type_id
payment_type
PK payment_type_id
payment_name
room_status
PK status_id
status_name
status_description
people
PK PID
first_name
last_name
street_address
state
zipcode
birthday
contact_number
email_address
employees
PK employee_id
FK PID
FK job_id
hire_date
hourly_wageusd
timesheet
PK timesheet_insert_id
FK employee_id
date
time_in
time_out
total_hours
memberships
PK membership_id
FK PID
FK customer_id
username
password
date_of_membership
customers
PK customer_id
FK PID
job_type
PK job_id
job_name
job_description

Tables

Attractions: This table contains different attractions in the area of Liverpool, Texas, including the hotel that this database focuses on.



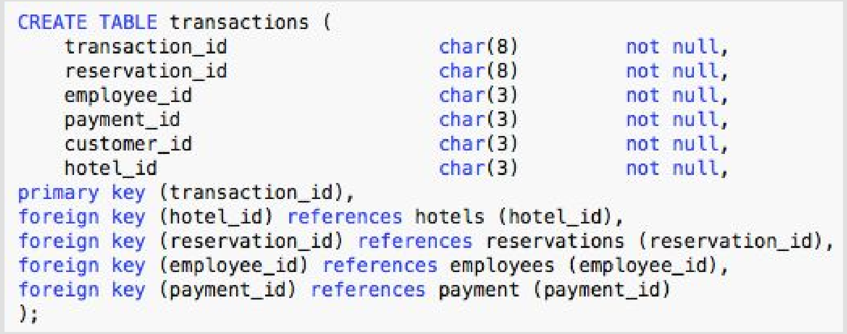
```
CREATE TABLE attractions (
    attraction_id                char(3)         not null,
    attraction_name              text            not null,
    attraction_description       text            not null,
    primary key (attraction_id)
);
```



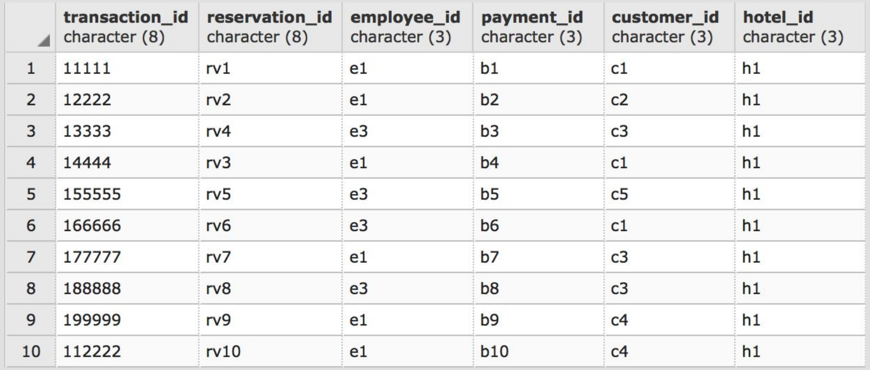
	attraction_id character (3)	attraction_name text	attraction_description text
1	a1	Hotel	Places to stay in the ar...
2	a2	FDR Museum	Places to check out his...
3	a3	Riverwalk	Places to explore the a...
4	a4	Restaurants	Places to eat in the area

Functional Dependencies: attraction_id → attraction_name, attraction_description

Transactions: This table holds all of the information regarding the transactions that go through this hotel day in and day out.



```
CREATE TABLE transactions (
    transaction_id                  char(8)         not null,
    reservation_id                  char(8)         not null,
    employee_id                     char(3)         not null,
    payment_id                      char(3)         not null,
    customer_id                     char(3)         not null,
    hotel_id                        char(3)         not null,
primary key (transaction_id),
foreign key (hotel_id) references hotels (hotel_id),
foreign key (reservation_id) references reservations (reservation_id),
foreign key (employee_id) references employees (employee_id),
foreign key (payment_id) references payment (payment_id)
);
```

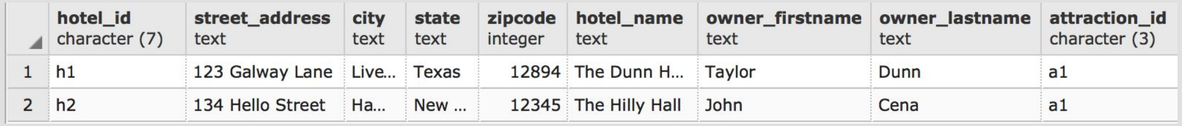


	transaction_id character (8)	reservation_id character (8)	employee_id character (3)	payment_id character (3)	customer_id character (3)	hotel_id character (3)
1	11111	rv1	e1	b1	c1	h1
2	12222	rv2	e1	b2	c2	h1
3	13333	rv4	e3	b3	c3	h1
4	14444	rv3	e1	b4	c1	h1
5	155555	rv5	e3	b5	c5	h1
6	166666	rv6	e3	b6	c1	h1
7	177777	rv7	e1	b7	c3	h1
8	188888	rv8	e3	b8	c3	h1
9	199999	rv9	e1	b9	c4	h1
10	112222	rv10	e1	b10	c4	h1

Functional Dependencies: transaction_id → employee_id, payment_id, reservation_id, customer_id, hotel_id

Hotels: This table contains the specific information about one of the hotel attractions in the area.

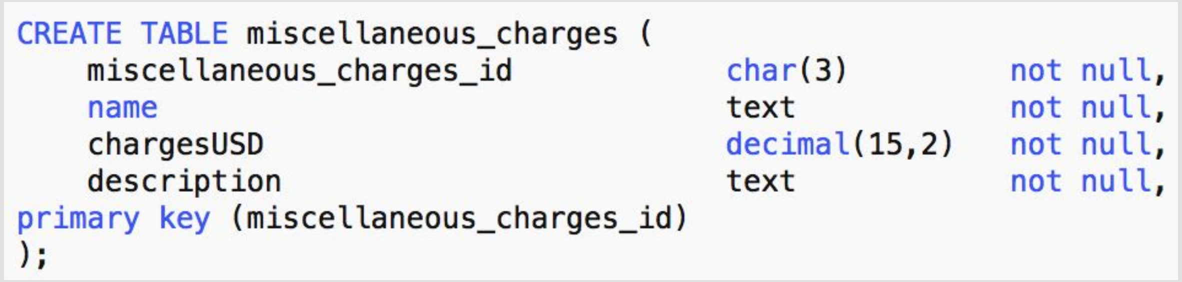
```
CREATE TABLE hotels (
    hotel_id                char(7)         not null,
    street_address          text            not null,
    city                    text            not null,
    state                   text            not null,
    zipcode                 integer         not null,
    hotel_name              text            not null,
    owner_firstname         text            not null,
    owner_lastname          text            not null,
    attraction_id           char(3)         not null,
    primary key (hotel_id),
    foreign key (attraction_id) references attractions (attraction_id)
);
```



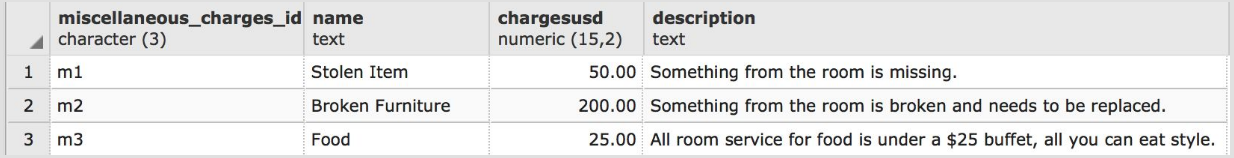
	hotel_id character (7)	street_address text	city text	state text	zipcode integer	hotel_name text	owner_firstname text	owner_lastname text	attraction_id character (3)
1	h1	123 Galway Lane	Live...	Texas	12894	The Dunn H...	Taylor	Dunn	a1
2	h2	134 Hello Street	Ha...	New ...	12345	The Hilly Hall	John	Cena	a1

Functional Dependencies: hotel_id → street_address, city, state, zipcode, hotel_name, owner_firstname, owner_lastname, attraction_id

Miscellaneous_Charges: This table contains the miscellaneous charges options that can be added to a transaction. These charges are broken down into three options, but more could be added as neccessary.



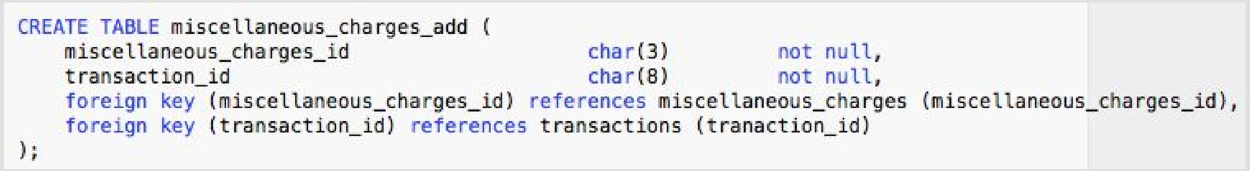
```
CREATE TABLE miscellaneous_charges (
    miscellaneous_charges_id            char(3)         not null,
    name                                text            not null,
    chargesUSD                          decimal(15,2)   not null,
    description                         text            not null,
primary key (miscellaneous_charges_id)
);
```



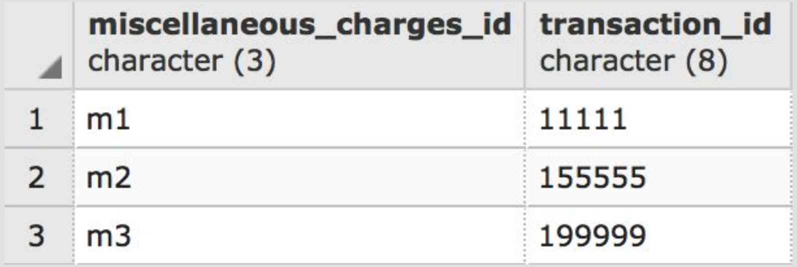
	miscellaneous_charges_id character (3)	name text	chargesusd numeric (15,2)	description text
1	m1	Stolen Item	50.00	Something from the room is missing.
2	m2	Broken Furniture	200.00	Something from the room is broken and needs to be replaced.
3	m3	Food	25.00	All room service for food is under a $25 buffet, all you can eat style.

Functional Dependencies: miscellaneous_charges_id → name, chargesUSD, description

Miscellaneous_Charges_Add: This table displays which transactions have miscellaneous charges in their orders.



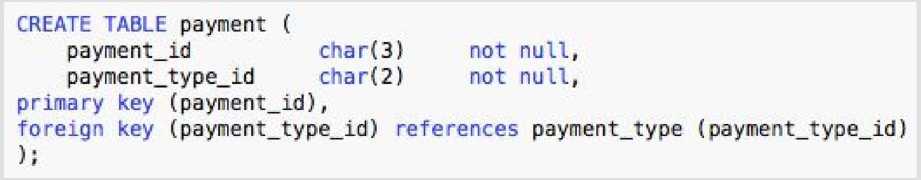
```
CREATE TABLE miscellaneous_charges_add (
   miscellaneous_charges_id                    char(3)        not null,
   transaction_id                              char(8)        not null,
   foreign key (miscellaneous_charges_id) references miscellaneous_charges (miscellaneous_charges_id),
   foreign key (transaction_id) references transactions (tranaction_id)
);
```



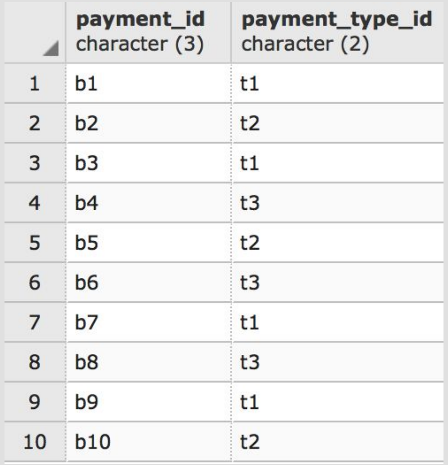
	miscellaneous_charges_id character (3)	transaction_id character (8)
1	m1	11111
2	m2	155555
3	m3	199999

Functional Dependencies : none

Payment: This table shows how a customer paid for their reservation, and is linked to the transaction table through the payment_id column.

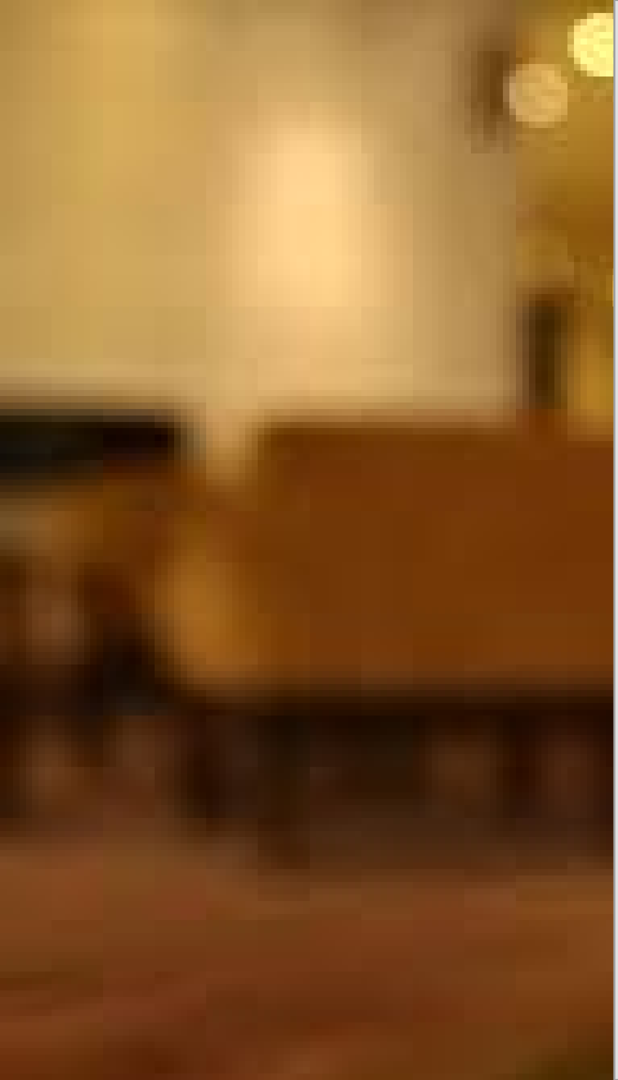


```
CREATE TABLE payment (
    payment_id          char(3)     not null,
    payment_type_id     char(2)     not null,
primary key (payment_id),
foreign key (payment_type_id) references payment_type (payment_type_id)
);
```

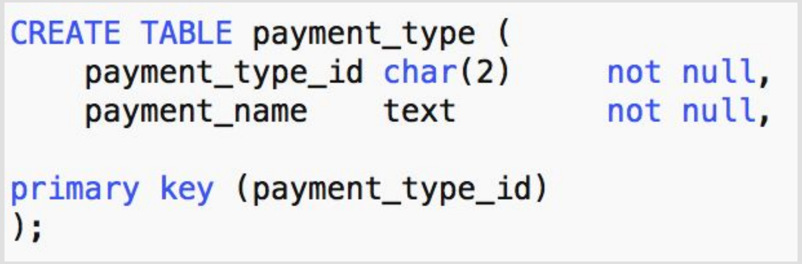


	payment_id character (3)	**payment_type_id** character (2)
1	b1	t1
2	b2	t2
3	b3	t1
4	b4	t3
5	b5	t2
6	b6	t3
7	b7	t1
8	b8	t3
9	b9	t1
10	b10	t2

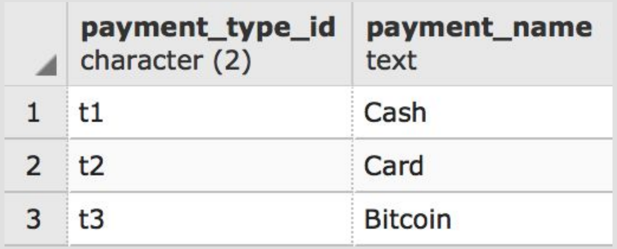
Functional Dependencies: payment_id → payment_type_id



Payment_Type: This table holds the different payment methods that this hotel accepts.



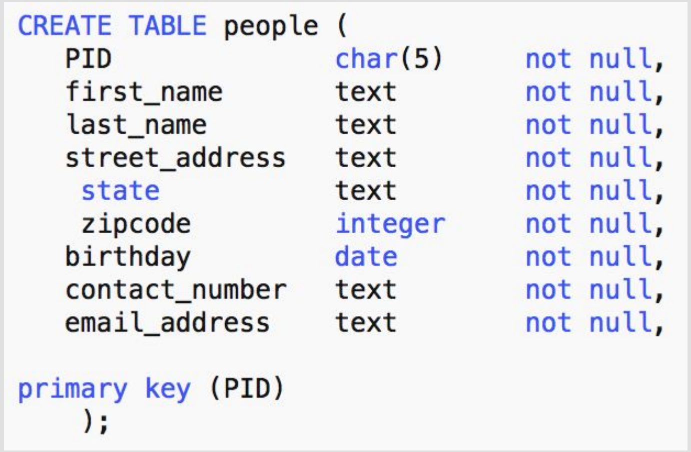
```
CREATE TABLE payment_type (
    payment_type_id char(2)     not null,
    payment_name    text        not null,

primary key (payment_type_id)
);
```



	payment_type_id character (2)	payment_name text
1	t1	Cash
2	t2	Card
3	t3	Bitcoin

Functional Dependencies: payment_type_id → payment_name

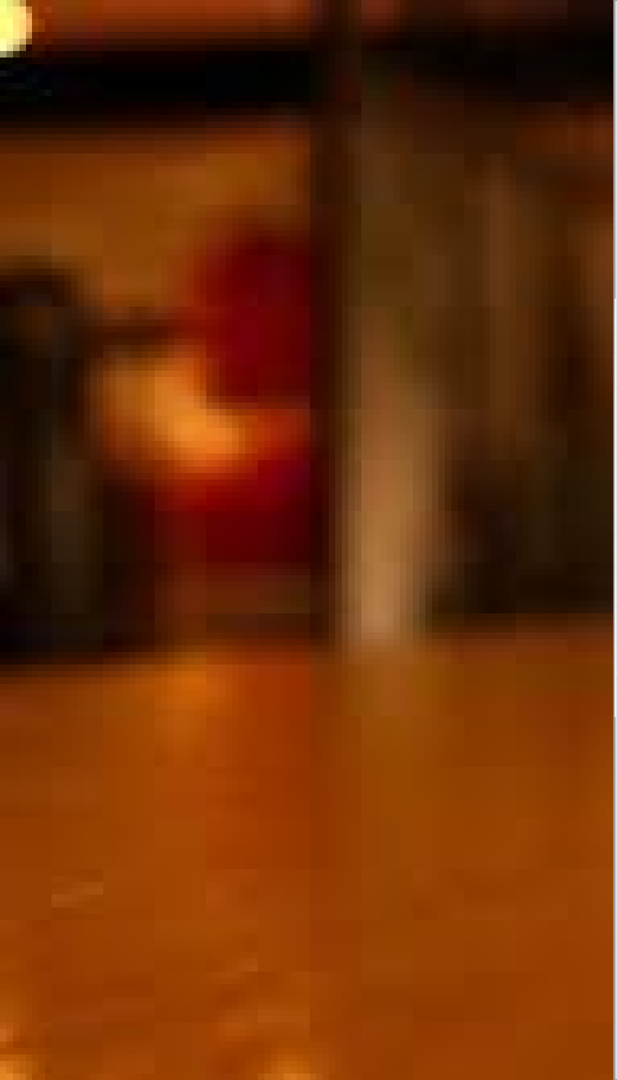
People: This table holds all of the people that interact with the hotel and it’s database.



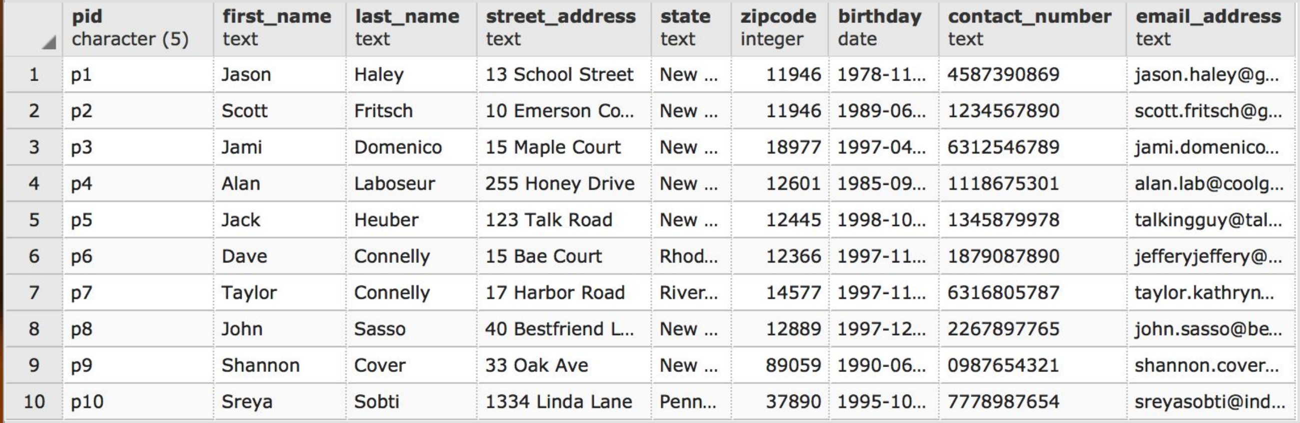
```
CREATE TABLE people (
   PID                char(5)     not null,
   first_name         text        not null,
   last_name          text        not null,
   street_address     text        not null,
    state             text        not null,
    zipcode           integer     not null,
   birthday           date        not null,
   contact_number     text        not null,
   email_address      text        not null,

primary key (PID)
    );
```

Functional Dependencies: pid → first_name, last_name, street_address, state, zipcode, birthday, contact_number, email_address

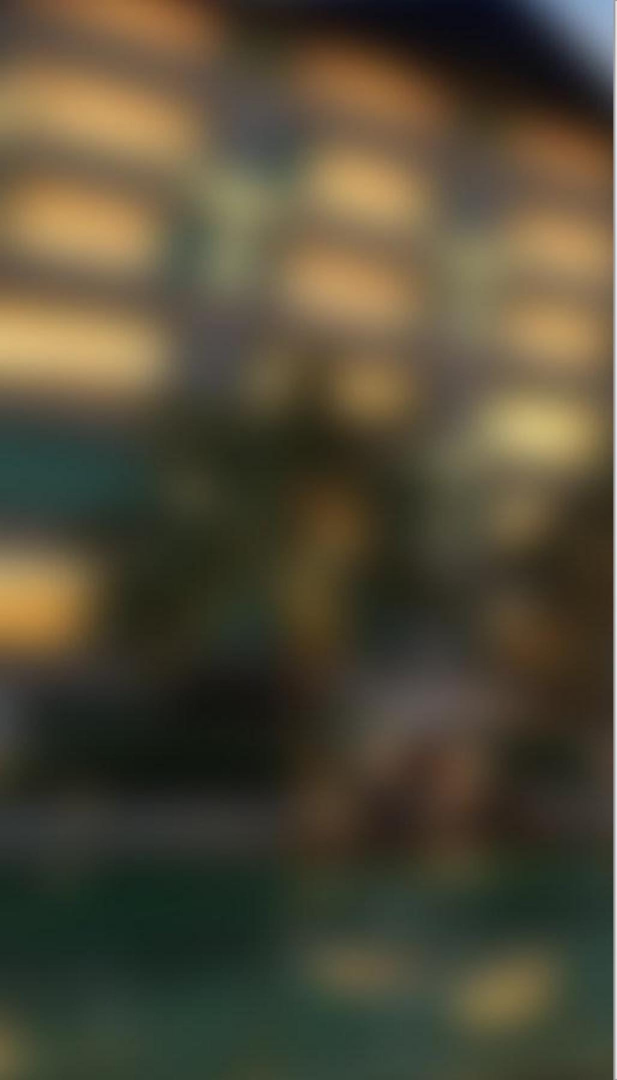
People Sample Data on next slide



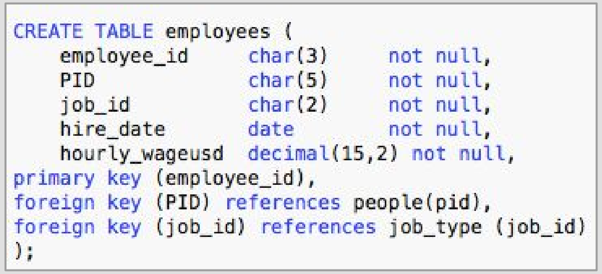
People Sample Data:



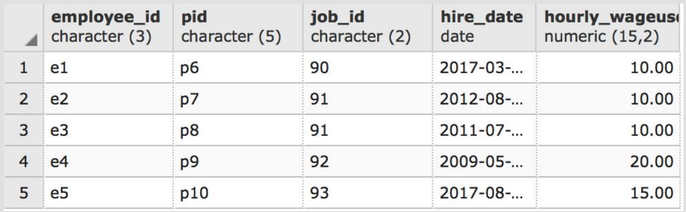
	pid	first_name	last_name	street_address	state	zipcode	birthday	contact_number	email_address
	character (5)	text	text	text	text	integer	date	text	text
1	p1	Jason	Haley	13 School Street	New ...	11946	1978-11...	4587390869	jason.haley@g...
2	p2	Scott	Fritsch	10 Emerson Co...	New ...	11946	1989-06...	1234567890	scott.fritsch@g...
3	p3	Jami	Domenico	15 Maple Court	New ...	18977	1997-04...	6312546789	jami.domenico...
4	p4	Alan	Laboseur	255 Honey Drive	New ...	12601	1985-09...	1118675301	alan.lab@coolg...
5	p5	Jack	Heuber	123 Talk Road	New ...	12445	1998-10...	1345879978	talkingguy@tal...
6	p6	Dave	Connelly	15 Bae Court	Rhod...	12366	1997-11...	1879087890	jefferyjeffery@...
7	p7	Taylor	Connelly	17 Harbor Road	River...	14577	1997-11...	6316805787	taylor.kathryn...
8	p8	John	Sasso	40 Bestfriend L...	New ...	12889	1997-12...	2267897765	john.sasso@be...
9	p9	Shannon	Cover	33 Oak Ave	New ...	89059	1990-06...	0987654321	shannon.cover...
10	p10	Sreya	Sobti	1334 Linda Lane	Penn...	37890	1995-10...	7778987654	sreyasobti@ind...



Employees and Customer: Both people, these tables connect to the people table and include extra information.

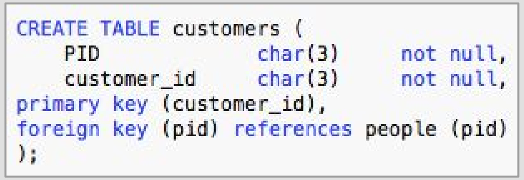


```
CREATE TABLE employees (
    employee_id     char(3)      not null,
    PID             char(5)      not null,
    job_id          char(2)      not null,
    hire_date       date         not null,
    hourly_wageusd  decimal(15,2) not null,
primary key (employee_id),
foreign key (PID) references people(pid),
foreign key (job_id) references job_type (job_id)
);
```

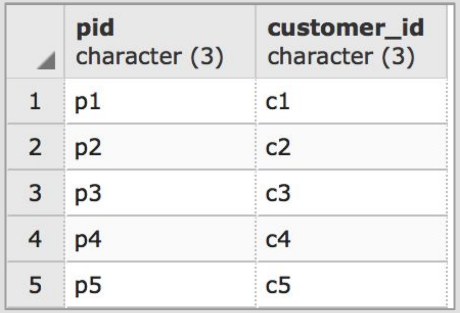


	employee_id character (3)	pid character (5)	job_id character (2)	hire_date date	hourly_wageus numeric (15,2)
1	e1	p6	90	2017-03-...	10.00
2	e2	p7	91	2012-08-...	10.00
3	e3	p8	91	2011-07-...	10.00
4	e4	p9	92	2009-05-...	20.00
5	e5	p10	93	2017-08-...	15.00

Functional Dependencies: employee_id → pid, job_id, hire_date, hourly_wageusd

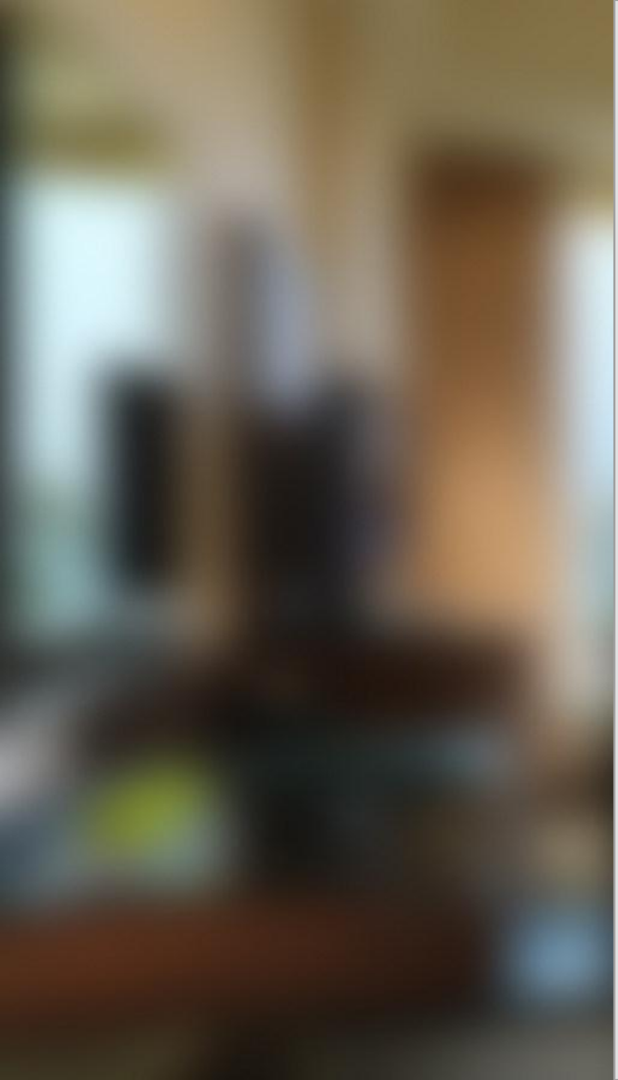


```
CREATE TABLE customers (
    PID             char(3)     not null,
    customer_id     char(3)     not null,
primary key (customer_id),
foreign key (pid) references people (pid)
);
```

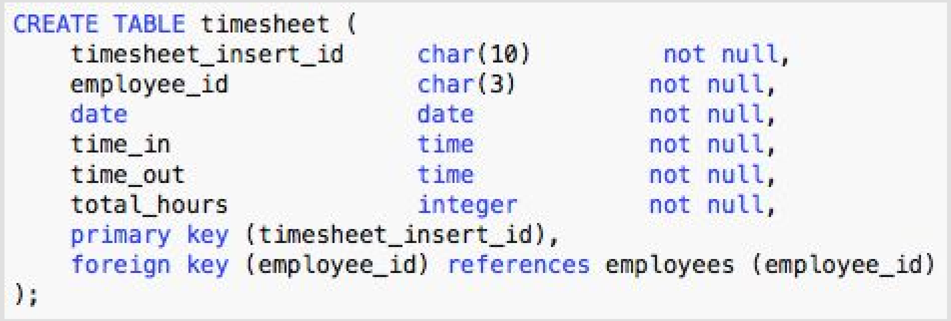


	pid character (3)	customer_id character (3)
1	p1	c1
2	p2	c2
3	p3	c3
4	p4	c4
5	p5	c5

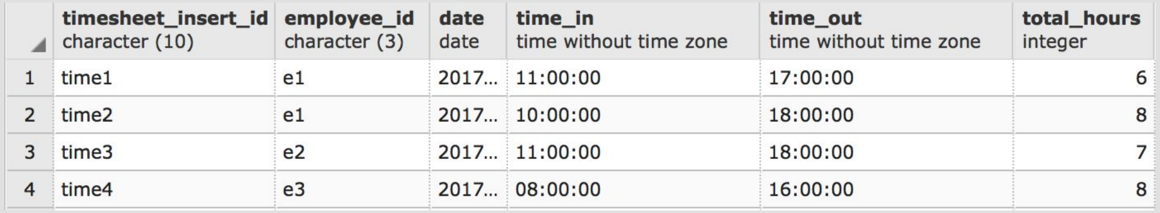
pid → customer_id



Timesheet: This table includes all of the employees and their hours.



```
CREATE TABLE timesheet (
    timesheet_insert_id     char(10)         not null,
    employee_id             char(3)         not null,
    date                    date            not null,
    time_in                 time            not null,
    time_out                time            not null,
    total_hours             integer         not null,
    primary key (timesheet_insert_id),
    foreign key (employee_id) references employees (employee_id)
);
```

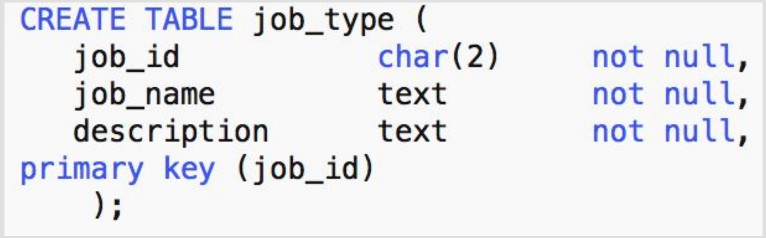


	timesheet_insert_id character (10)	employee_id character (3)	date date	time_in time without time zone	time_out time without time zone	total_hours integer
1	time1	e1	2017...	11:00:00	17:00:00	6
2	time2	e1	2017...	10:00:00	18:00:00	8
3	time3	e2	2017...	11:00:00	18:00:00	7
4	time4	e3	2017...	08:00:00	16:00:00	8

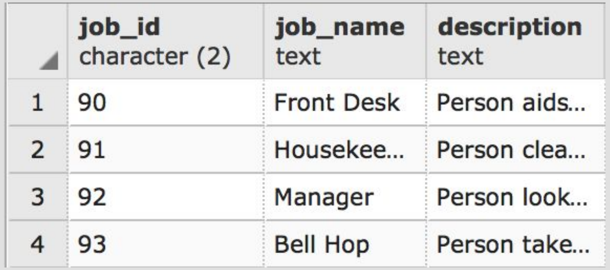
Functional Dependencies : employee_id → date, time_in, time_out

total _hours → time_in, time_out

Job_Type: This table holds information about different jobs that the employees hold.



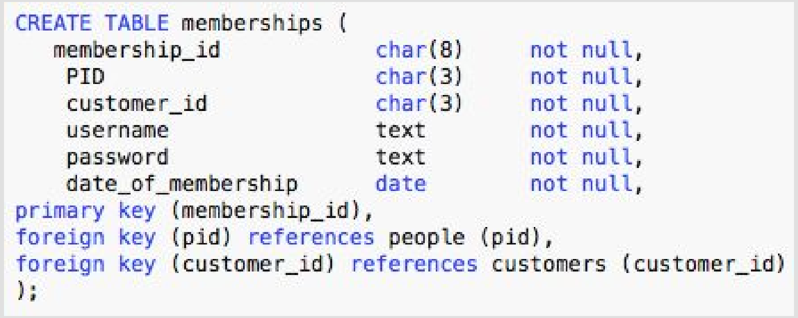
```
CREATE TABLE job_type (
   job_id          char(2)     not null,
   job_name        text        not null,
   description     text        not null,
primary key (job_id)
    );
```



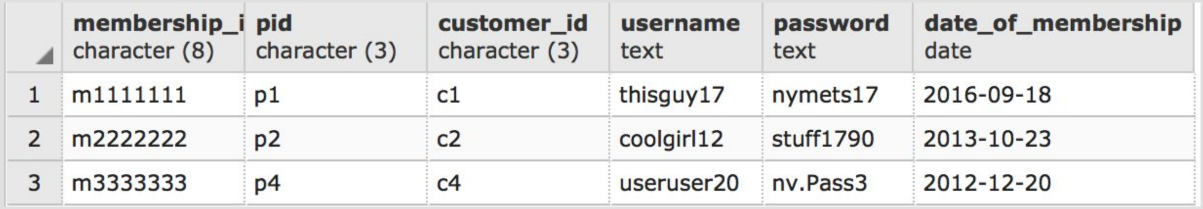
	job_id character (2)	job_name text	description text
1	90	Front Desk	Person aids...
2	91	Housekee...	Person clea...
3	92	Manager	Person look...
4	93	Bell Hop	Person take...

Functional Dependencies: job_id → job_name, job_description

Memberships: This table contains membership information for customers who are considered members.



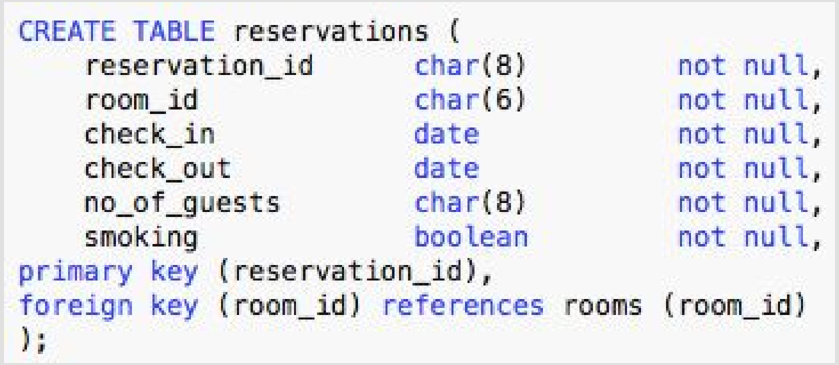
```
CREATE TABLE memberships (
   membership_id            char(8)     not null,
    PID                     char(3)     not null,
    customer_id             char(3)     not null,
    username                text        not null,
    password                text        not null,
    date_of_membership      date        not null,
primary key (membership_id),
foreign key (pid) references people (pid),
foreign key (customer_id) references customers (customer_id)
);
```



	membership_i character (8)	pid character (3)	customer_id character (3)	username text	password text	date_of_membership date
1	m1111111	p1	c1	thisguy17	nymets17	2016-09-18
2	m2222222	p2	c2	coolgirl12	stuff1790	2013-10-23
3	m3333333	p4	c4	useruser20	nv.Pass3	2012-12-20

Functional Dependencies: membership_id → pid, customer_id, username, password, date_of_membership

Reservations: This table contains all information about the reservations a customer submits or a front desk worker processes.

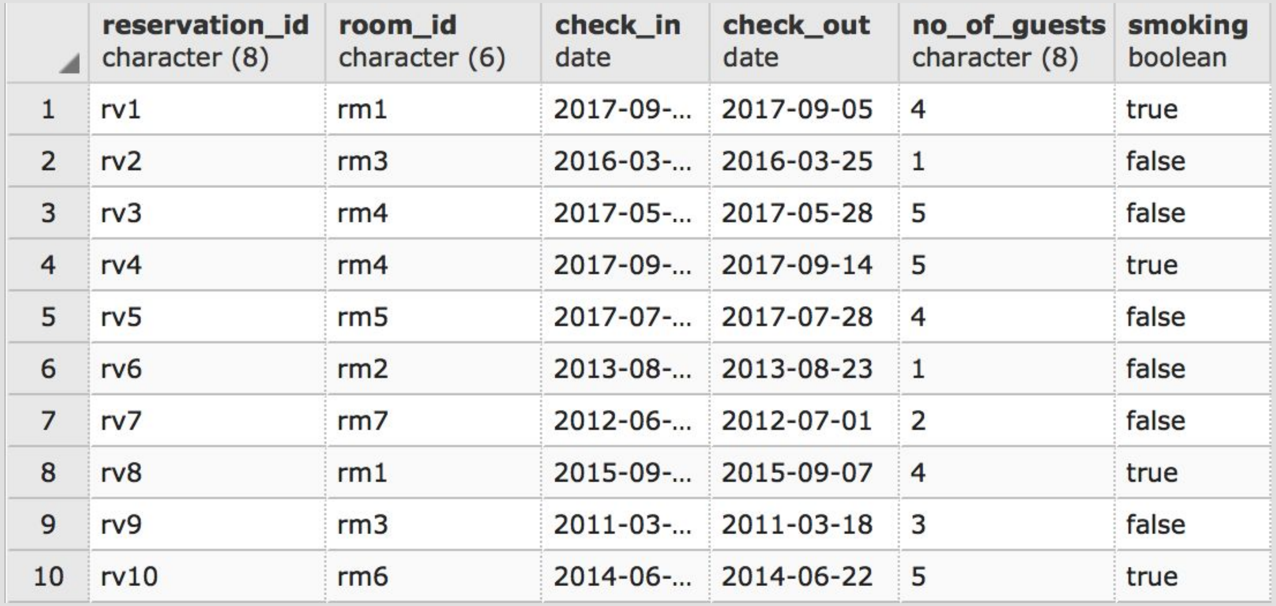


```
CREATE TABLE reservations (
    reservation_id      char(8)         not null,
    room_id             char(6)         not null,
    check_in            date            not null,
    check_out           date            not null,
    no_of_guests        char(8)         not null,
    smoking             boolean         not null,
primary key (reservation_id),
foreign key (room_id) references rooms (room_id)
);
```

Functional Dependencies: reservation_id → check_in, check_out, no_of_guests, room_id, customer_id, transaction_id

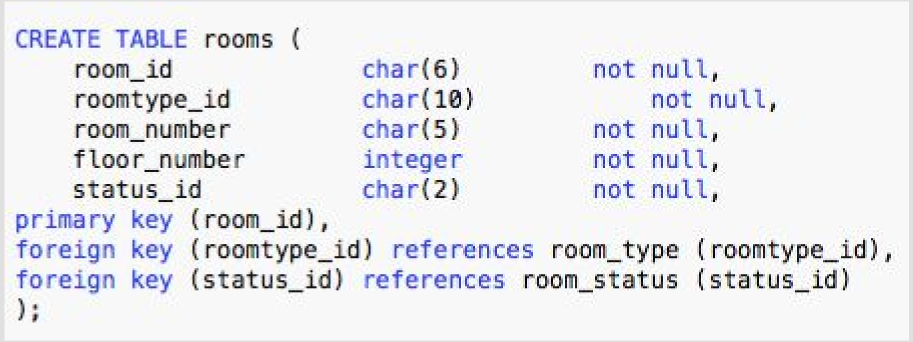
Reservation sample data on next slide

Reservations Sample Data:

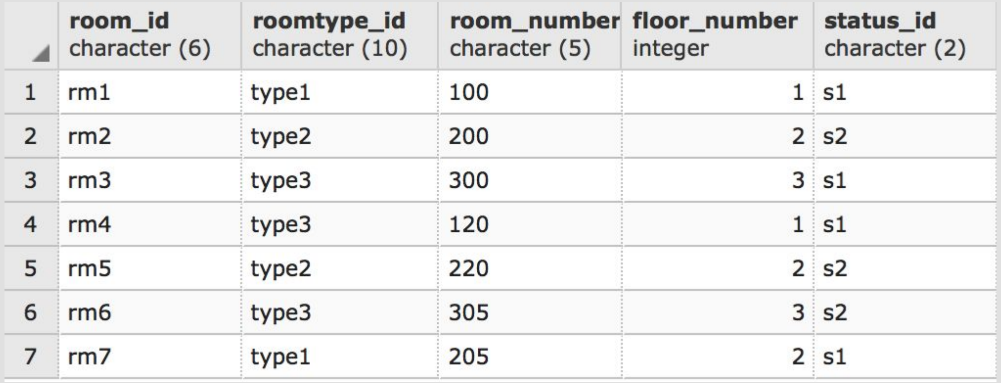


	reservation_id character (8)	room_id character (6)	check_in date	check_out date	no_of_guests character (8)	smoking boolean
1	rv1	rm1	2017-09-...	2017-09-05	4	true
2	rv2	rm3	2016-03-...	2016-03-25	1	false
3	rv3	rm4	2017-05-...	2017-05-28	5	false
4	rv4	rm4	2017-09-...	2017-09-14	5	true
5	rv5	rm5	2017-07-...	2017-07-28	4	false
6	rv6	rm2	2013-08-...	2013-08-23	1	false
7	rv7	rm7	2012-06-...	2012-07-01	2	false
8	rv8	rm1	2015-09-...	2015-09-07	4	true
9	rv9	rm3	2011-03-...	2011-03-18	3	false
10	rv10	rm6	2014-06-...	2014-06-22	5	true

Rooms: This table holds all the information about different rooms in the hotel.



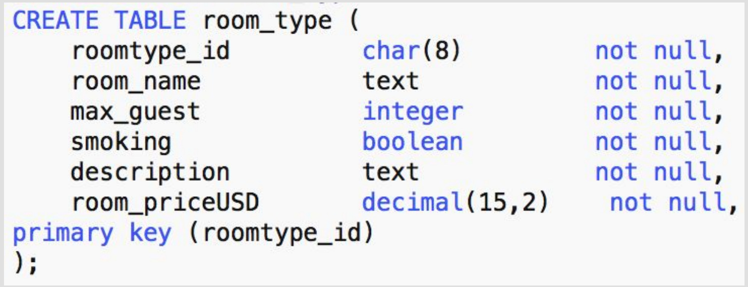
```
CREATE TABLE rooms (
    room_id                char(6)          not null,
    roomtype_id            char(10)             not null,
    room_number            char(5)          not null,
    floor_number           integer          not null,
    status_id              char(2)          not null,
primary key (room_id),
foreign key (roomtype_id) references room_type (roomtype_id),
foreign key (status_id) references room_status (status_id)
);
```



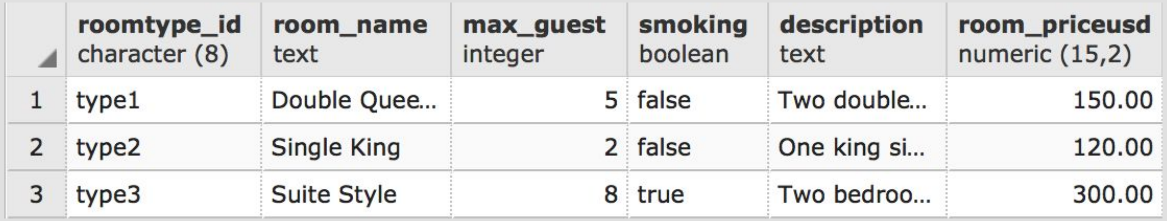
	room_id character (6)	roomtype_id character (10)	room_number character (5)	floor_number integer	status_id character (2)
1	rm1	type1	100	1	s1
2	rm2	type2	200	2	s2
3	rm3	type3	300	3	s1
4	rm4	type3	120	1	s1
5	rm5	type2	220	2	s2
6	rm6	type3	305	3	s2
7	rm7	type1	205	2	s1

Functional Dependencies: room_id → roomtype_id, room_number, floor_num, status_id

Room_Type: This table holds all of the room types and their other attributes.



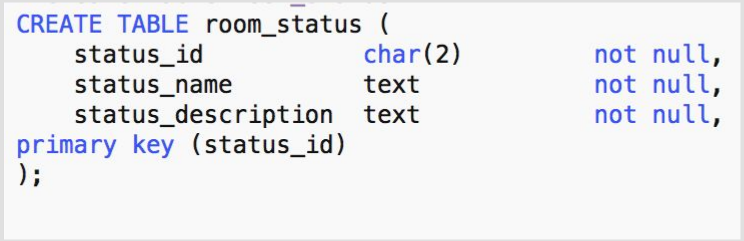
```
CREATE TABLE room_type (
    roomtype_id         char(8)          not null,
    room_name           text             not null,
    max_guest           integer          not null,
    smoking             boolean          not null,
    description         text             not null,
    room_priceUSD       decimal(15,2)     not null,
primary key (roomtype_id)
);
```



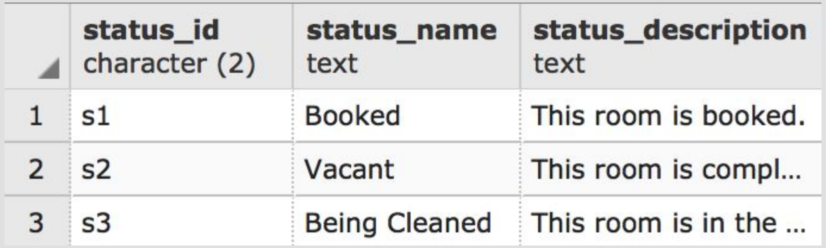
	roomtype_id character (8)	room_name text	max_guest integer	smoking boolean	description text	room_priceusd numeric (15,2)
1	type1	Double Quee...	5	false	Two double...	150.00
2	type2	Single King	2	false	One king si...	120.00
3	type3	Suite Style	8	true	Two bedroo...	300.00

Functional Dependencies: roomtype_id → room_name, max_guest, smoking, description, room_price_usd

Room_Status: This table displays whether the room is booked, vacant or being cleaned.



```
CREATE TABLE room_status (
    status_id           char(2)         not null,
    status_name         text            not null,
    status_description  text            not null,
primary key (status_id)
);
```



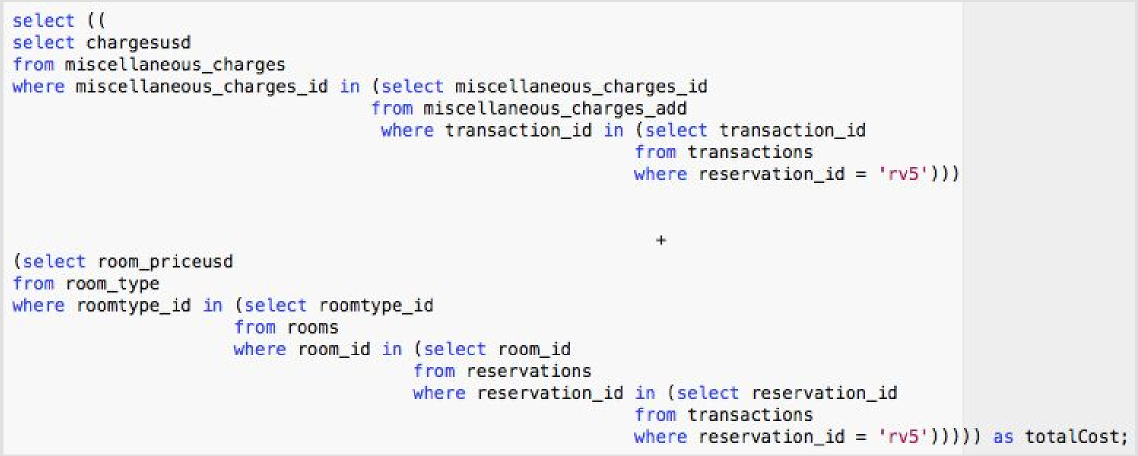
	status_id character (2)	status_name text	status_description text
1	s1	Booked	This room is booked.
2	s2	Vacant	This room is compl...
3	s3	Being Cleaned	This room is in the ...

Functional Dependencies: status_id → status_name, status_description

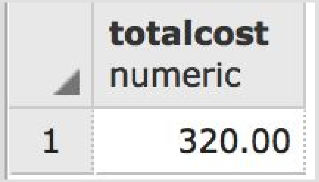
Views, Triggers, Stored Procedures, Reports

Views: Total Cost

This query will find the total price a customer must pay for their visit, based upon reservation id. This is a quick and easy way for the total cost of a customer's visit to be calculated and eventually processed through payment methods.



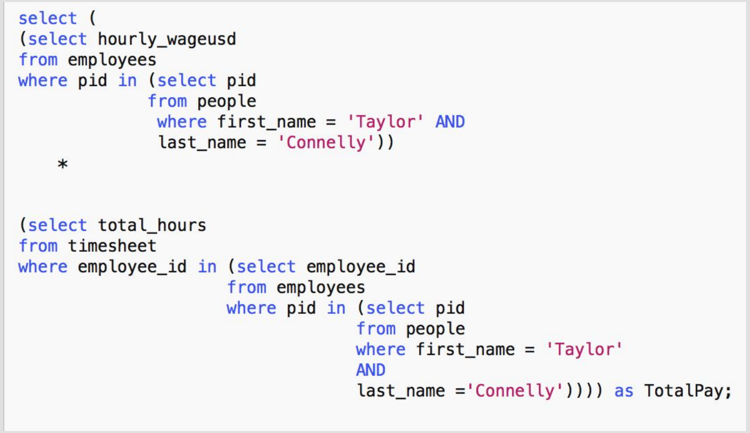
```
select ((
select chargesusd
from miscellaneous_charges
where miscellaneous_charges_id in (select miscellaneous_charges_id
                                  from miscellaneous_charges_add
                                   where transaction_id in (select transaction_id
                                                            from transactions
                                                            where reservation_id = 'rv5')))


                                                               +
(select room_priceusd
from room_type
where roomtype_id in (select roomtype_id
                     from rooms
                     where room_id in (select room_id
                                      from reservations
                                      where reservation_id in (select reservation_id
                                                               from transactions
                                                               where reservation_id = 'rv5'))))) as totalCost;
```



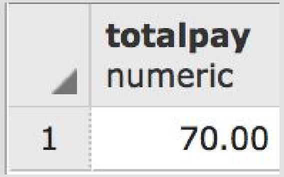
	totalcost numeric
1	320.00

Views: Total Pay

This view will show how much an employee will make for working a certain number of hours. This is helpful for the manager or owner to calculate how much money they will have to pay their employees for working their hours that week.



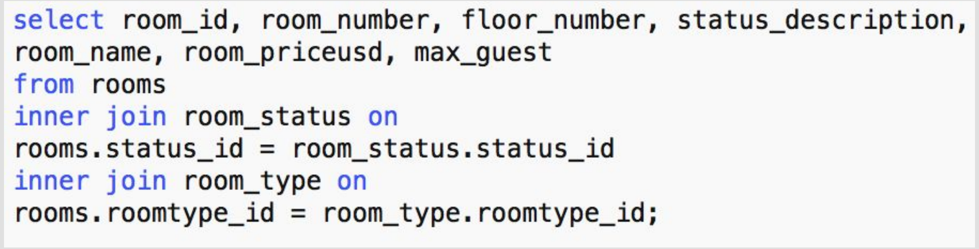
```
select (
(select hourly_wageusd
from employees
where pid in (select pid
              from people
               where first_name = 'Taylor' AND
               last_name = 'Connelly'))
    *


(select total_hours
from timesheet
where employee_id in (select employee_id
                      from employees
                      where pid in (select pid
                                    from people
                                    where first_name = 'Taylor'
                                    AND
                                    last_name ='Connelly')))) as TotalPay;
```



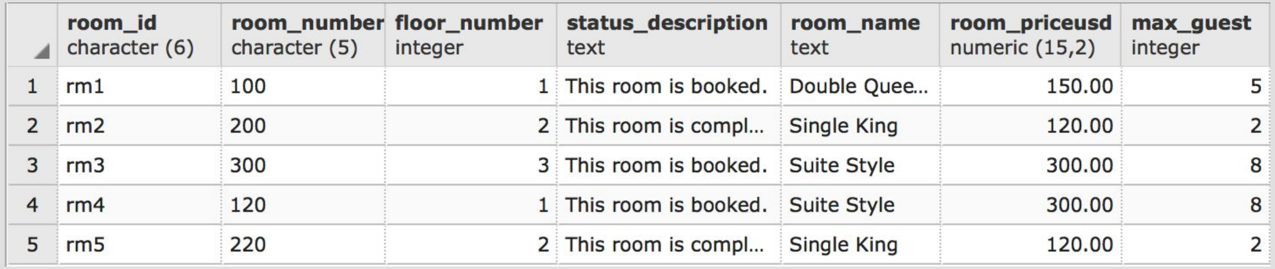
	totalpay numeric
1	70.00

Views: Quick View Of Room Information

This view will give the employee working at the front desk a quick dashboard of the important information they need to know if a customer wants to book a room.

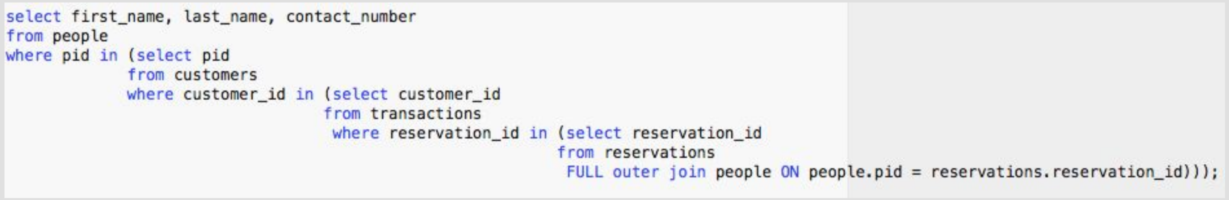


```
select room_id, room_number, floor_number, status_description,
room_name, room_priceusd, max_guest
from rooms
inner join room_status on
rooms.status_id = room_status.status_id
inner join room_type on
rooms.roomtype_id = room_type.roomtype_id;
```

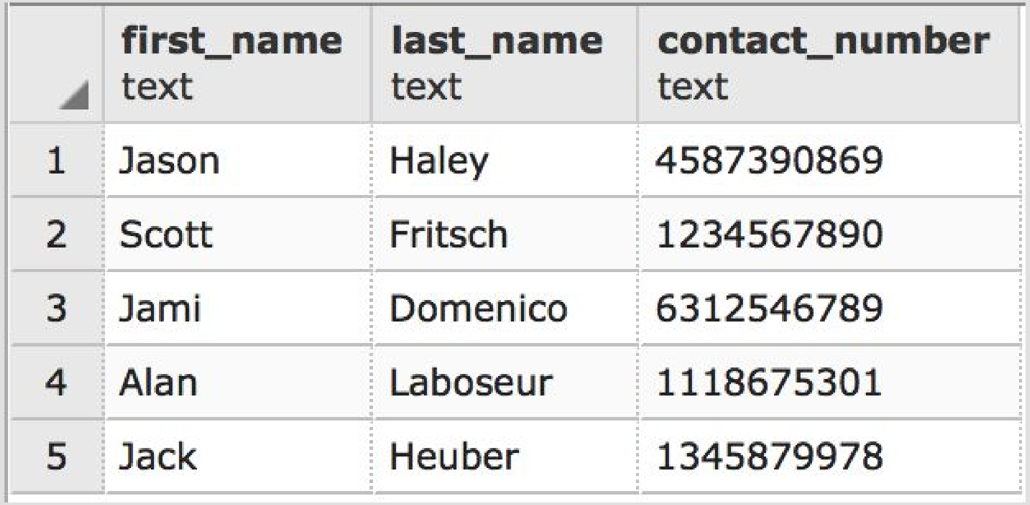


	room_id character (6)	room_number character (5)	floor_number integer	status_description text	room_name text	room_priceusd numeric (15,2)	max_guest integer
1	rm1	100	1	This room is booked.	Double Quee...	150.00	5
2	rm2	200	2	This room is compl...	Single King	120.00	2
3	rm3	300	3	This room is booked.	Suite Style	300.00	8
4	rm4	120	1	This room is booked.	Suite Style	300.00	8
5	rm5	220	2	This room is compl...	Single King	120.00	2

Views: Customer Information
This view shows customers that have made reservations, and their important information.



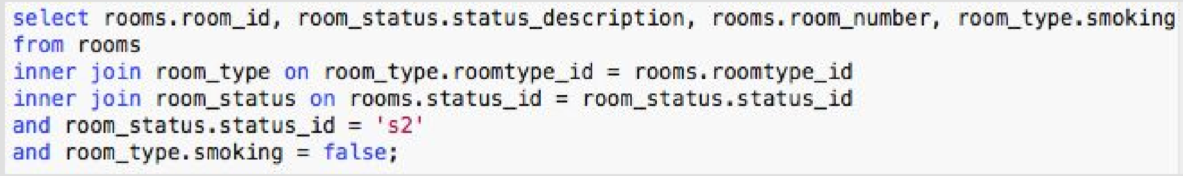
```
select first_name, last_name, contact_number
from people
where pid in (select pid
             from customers
             where customer_id in (select customer_id
                                   from transactions
                                    where reservation_id in (select reservation_id
                                                             from reservations
                                                              FULL outer join people ON people.pid = reservations.reservation_id)));
```



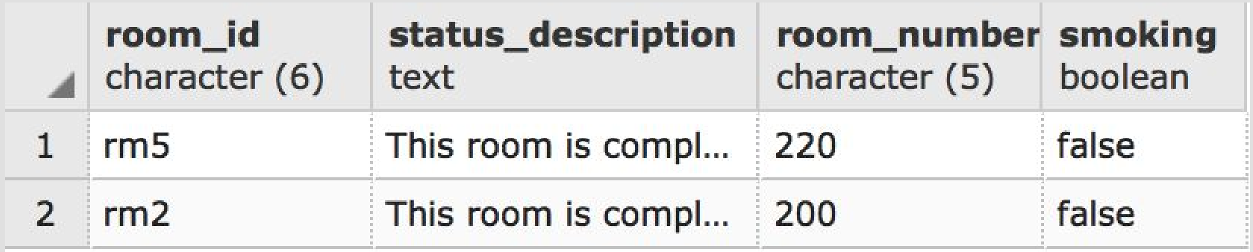
	first_name text	last_name text	contact_number text
1	Jason	Haley	4587390869
2	Scott	Fritsch	1234567890
3	Jami	Domenico	6312546789
4	Alan	Laboseur	1118675301
5	Jack	Heuber	1345879978

Views: Non-Smoking Rooms

This view simply shows the rooms that are non-smoking, and also available to be booked at the time of the query.



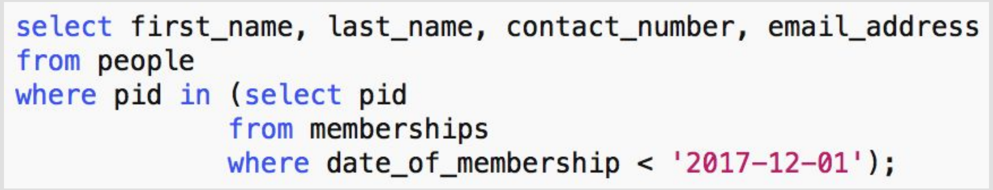
```
select rooms.room_id, room_status.status_description, rooms.room_number, room_type.smoking
from rooms
inner join room_type on room_type.roomtype_id = rooms.roomtype_id
inner join room_status on rooms.status_id = room_status.status_id
and room_status.status_id = 's2'
and room_type.smoking = false;
```



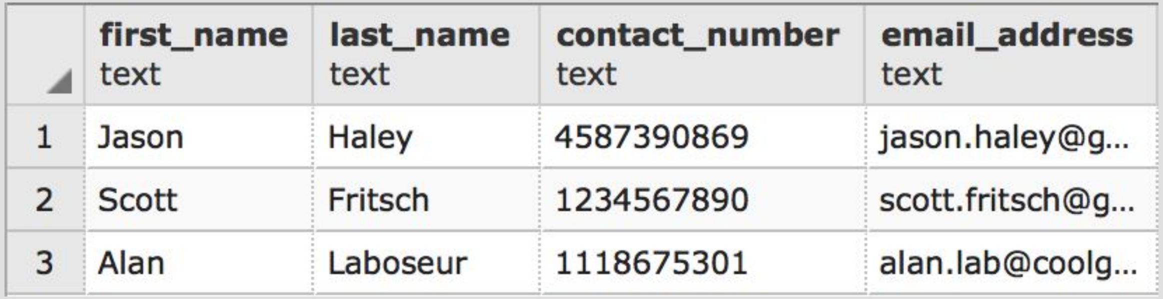
	room_id character (6)	status_description text	room_number character (5)	smoking boolean
1	rm5	This room is compl...	220	false
2	rm2	This room is compl...	200	false

Views: Gold Members

This view simply shows members that have been with the hotel for over a year. This accomplishment warrants special treatment from the hotel, whether that be some sort of discount or promo.



```
select first_name, last_name, contact_number, email_address
from people
where pid in (select pid
              from memberships
              where date_of_membership < '2017-12-01');
```

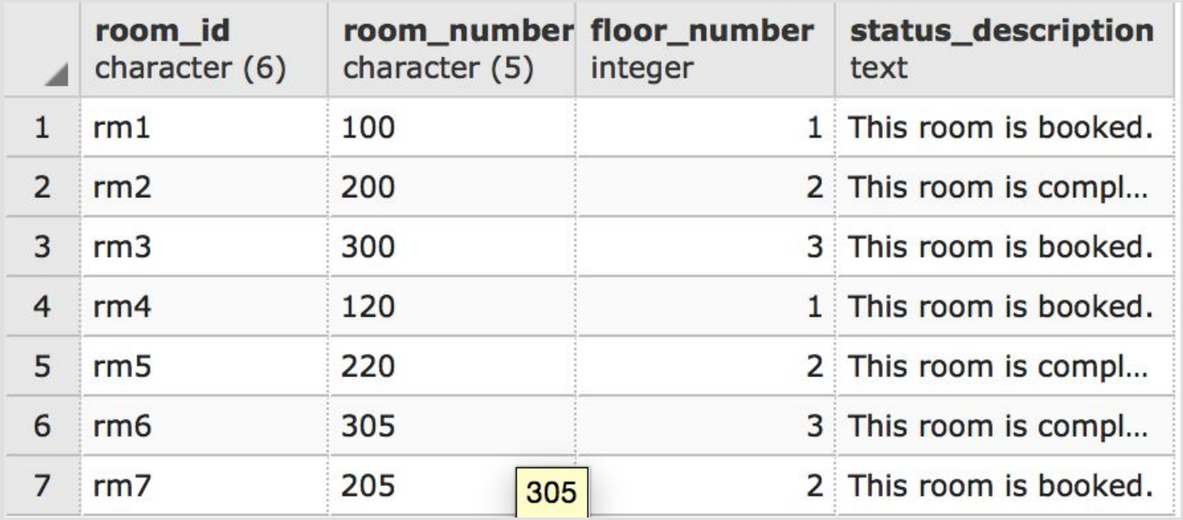


	first_name text	last_name text	contact_number text	email_address text
1	Jason	Haley	4587390869	jason.haley@g...
2	Scott	Fritsch	1234567890	scott.fritsch@g...
3	Alan	Laboseur	1118675301	alan.lab@coolg...

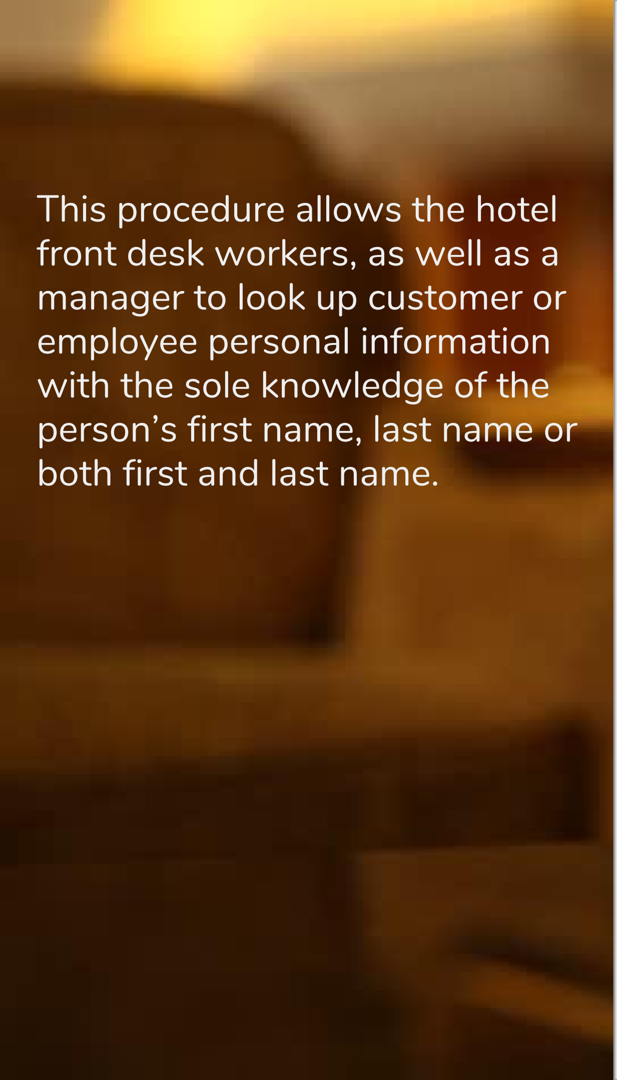
Views: Room Status

This view tells you the status of the rooms in hotel. This is helpful for those employees who are booking the reservations.

```
select room_id, room_number, floor_number, status_description
from rooms
inner join room_status on rooms.status_id = room_status.status_id;
```

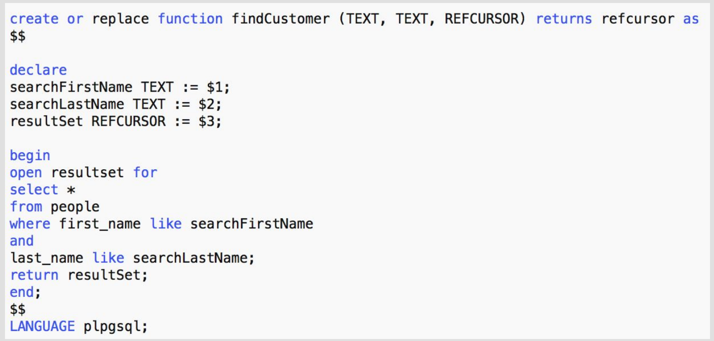


	room_id character (6)	room_number character (5)	floor_number integer	status_description text
1	rm1	100	1	This room is booked.
2	rm2	200	2	This room is compl...
3	rm3	300	3	This room is booked.
4	rm4	120	1	This room is booked.
5	rm5	220	2	This room is compl...
6	rm6	305	3	This room is compl...
7	rm7	205	2	This room is booked.

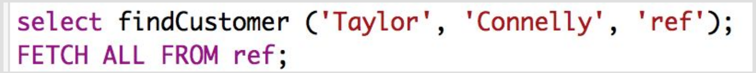


This procedure allows the hotel front desk workers, as well as a manager to look up customer or employee personal information with the sole knowledge of the person’s first name, last name or both first and last name.

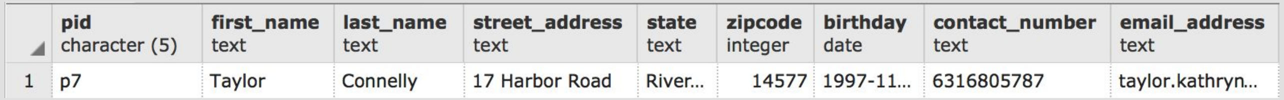
Stored Procedure: findCustomer



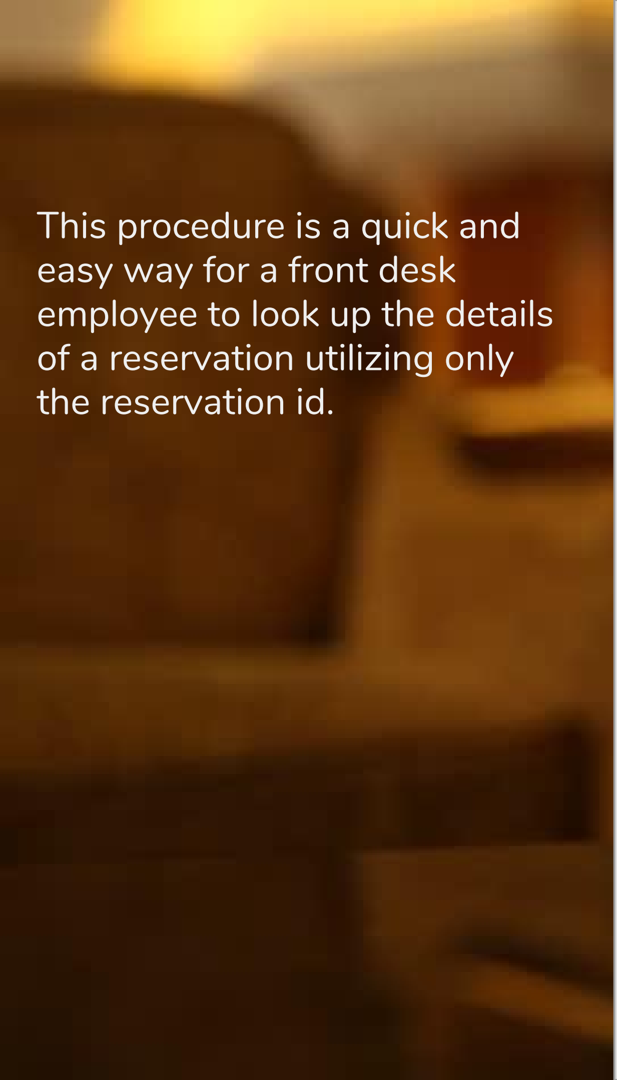
```
create or replace function findCustomer (TEXT, TEXT, REFCURSOR) returns refcursor as
$$

declare
searchFirstName TEXT := $1;
searchLastName TEXT := $2;
resultSet REFCURSOR := $3;

begin
open resultset for
select *
from people
where first_name like searchFirstName
and
last_name like searchLastName;
return resultSet;
end;
$$
LANGUAGE plpgsql;
```



```
select findCustomer ('Taylor', 'Connelly', 'ref');
FETCH ALL FROM ref;
```

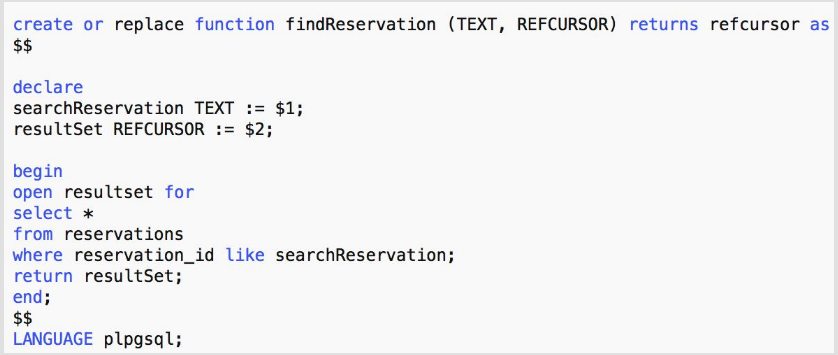


	pid character (5)	first_name text	last_name text	street_address text	state text	zipcode integer	birthday date	contact_number text	email_address text
1	p7	Taylor	Connelly	17 Harbor Road	River...	14577	1997-11...	6316805787	taylor.kathryn...

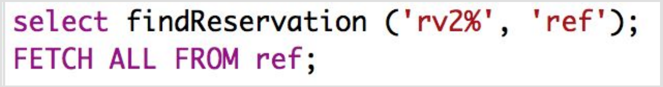


This procedure is a quick and easy way for a front desk employee to look up the details of a reservation utilizing only the reservation id.

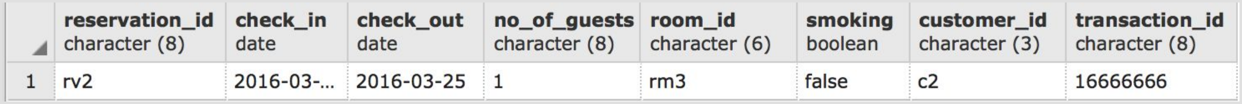
Stored Procedure: findReservation



```
create or replace function findReservation (TEXT, REFCURSOR) returns refcursor as
$$

declare
searchReservation TEXT := $1;
resultSet REFCURSOR := $2;

begin
open resultset for
select *
from reservations
where reservation_id like searchReservation;
return resultSet;
end;
$$
LANGUAGE plpgsql;
```



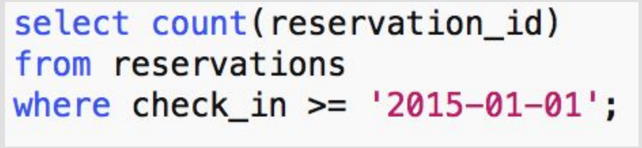
```
select findReservation ('rv2%', 'ref');
FETCH ALL FROM ref;
```



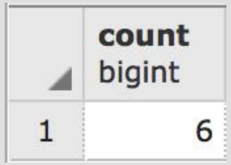
	reservation_id character (8)	check_in date	check_out date	no_of_guests character (8)	room_id character (6)	smoking boolean	customer_id character (3)	transaction_id character (8)
1	rv2	2016-03-...	2016-03-25	1	rm3	false	c2	16666666

Reports:

Total number of reservations after 2015 (look for trends, see what to do to improve the number of reservations overtime):

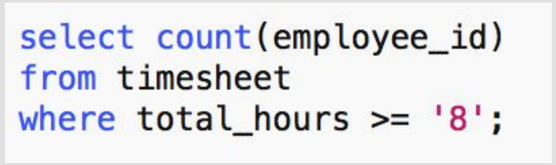


```
select count(reservation_id)
from reservations
where check_in >= '2015-01-01';
```

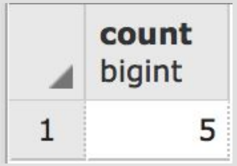


	count bigint
1	6

Total number of employees that have worked over 8 hours (could be adapted to show overtime pay information):



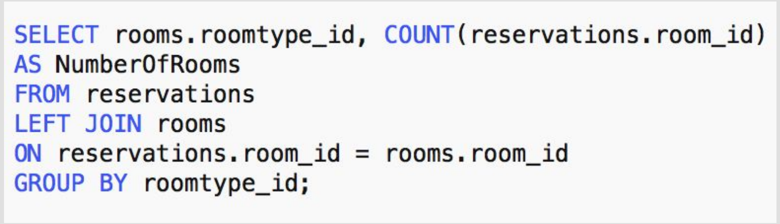
```
select count(employee_id)
from timesheet
where total_hours >= '8';
```



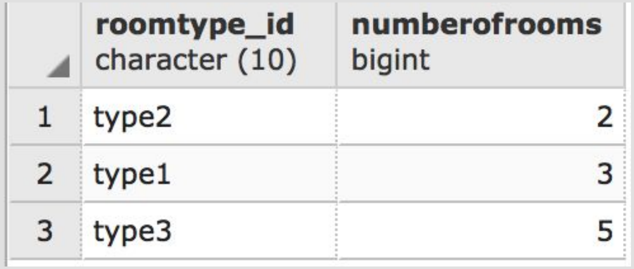
	count bigint
1	5

Reports:

This report groups together how many reservations are being booked in each room type. This could show the owners of the hotel which rooms are in the highest demand, and could lead to changes within the hotels infrastructure, such as adding more of a certain room type to the hotel itself.



```
SELECT rooms.roomtype_id, COUNT(reservations.room_id)
AS NumberOfRooms
FROM reservations
LEFT JOIN rooms
ON reservations.room_id = rooms.room_id
GROUP BY roomtype_id;
```

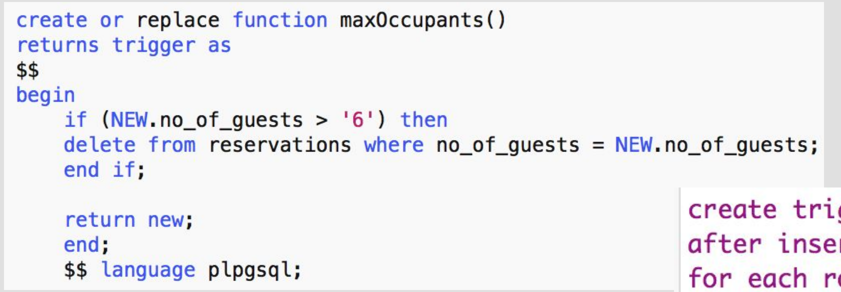


	roomtype_id character (10)	numberofrooms bigint
1	type2	2
2	type1	3
3	type3	5

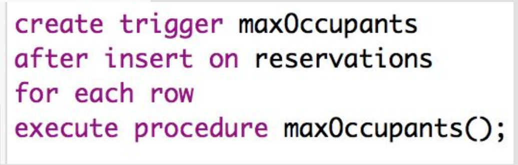
Trigger: maxOccupants

The hotel does not allow more than 6 occupants to a room in one reservation. Any time that this is entered into the database it is deleted immediately.

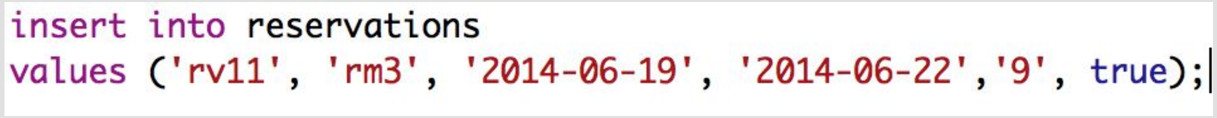
The following reservation was attempted to be added. The result is the dataset without rv11.



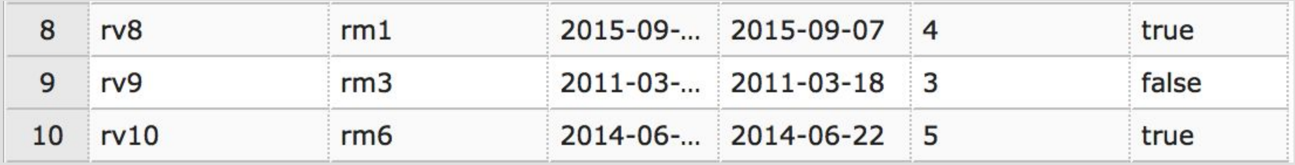
```
create or replace function maxOccupants()
returns trigger as
$$
begin
    if (NEW.no_of_guests > '6') then
    delete from reservations where no_of_guests = NEW.no_of_guests;
    end if;

    return new;
    end;
    $$ language plpgsql;
```



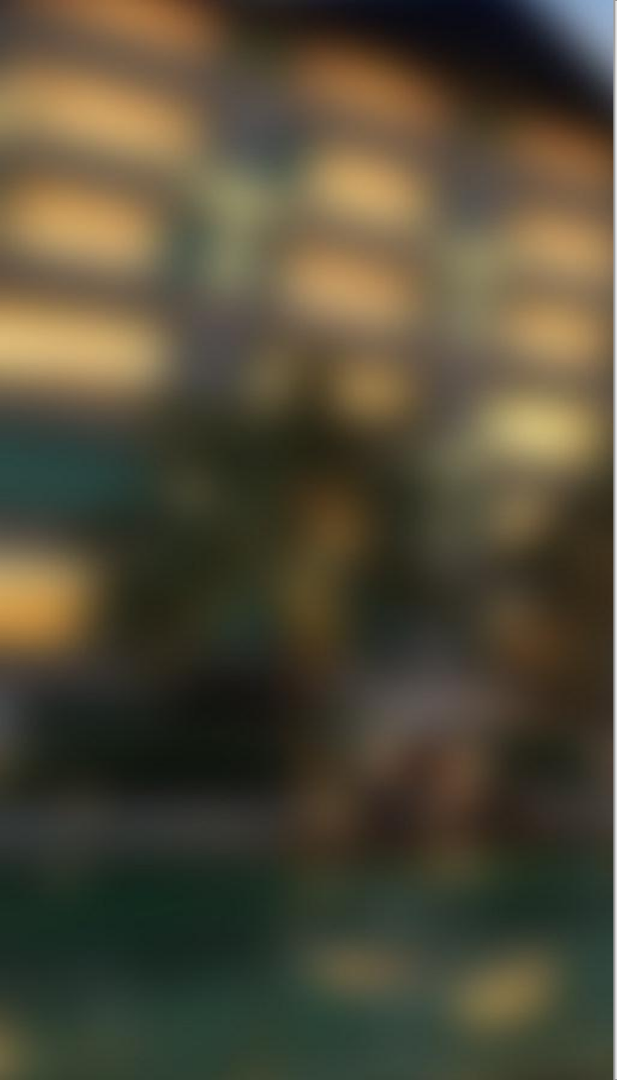
```
create trigger maxOccupants
after insert on reservations
for each row
execute procedure maxOccupants();
```



```
insert into reservations
values ('rv11', 'rm3', '2014-06-19', '2014-06-22','9', true);
```

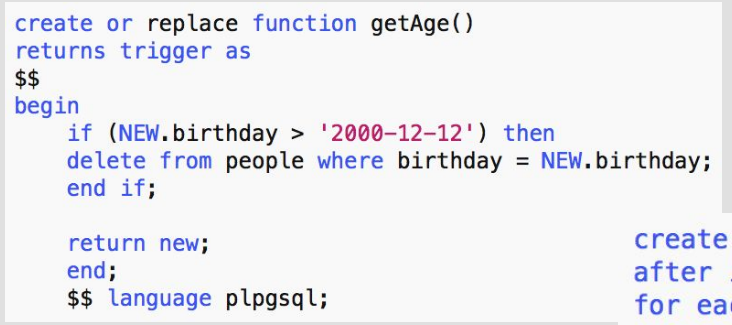


8	rv8	rm1	2015-09-...	2015-09-07	4	true
9	rv9	rm3	2011-03-...	2011-03-18	3	false
10	rv10	rm6	2014-06-...	2014-06-22	5	true

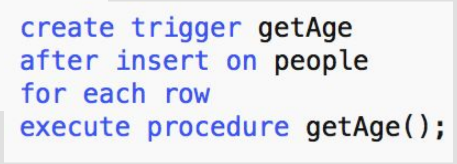


Trigger: getAge

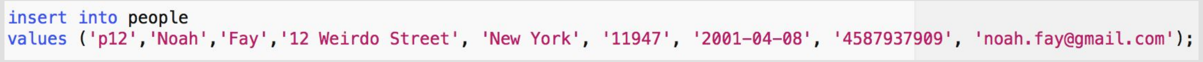
The hotel does not want any employees or customers working or booking reservations under the age of 18 for liability reasons. Customers and employees are deleted from the database if this is the case.



```
create or replace function getAge()
returns trigger as
$$
begin
    if (NEW.birthday > '2000-12-12') then
    delete from people where birthday = NEW.birthday;
    end if;

    return new;
    end;
    $$ language plpgsql;
```

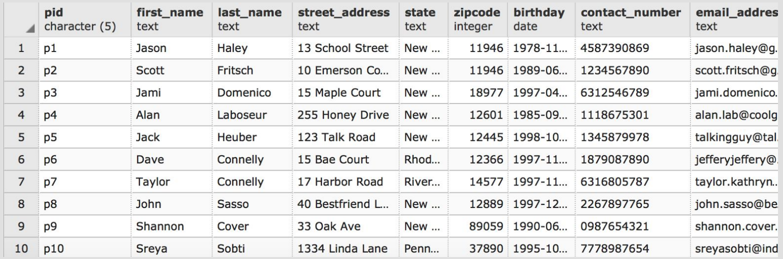


```
create trigger getAge
after insert on people
for each row
execute procedure getAge();
```

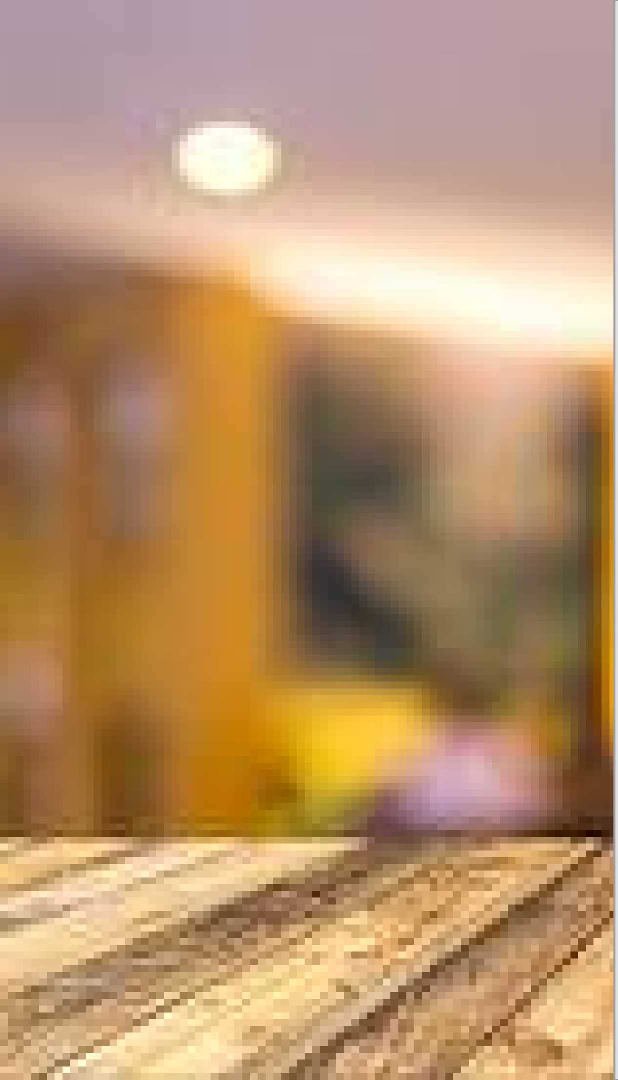


```
insert into people
values ('p12','Noah','Fay','12 Weirdo Street', 'New York', '11947', '2001-04-08', '4587937909', 'noah.fay@gmail.com');
```

P12 Noah Fay not added



	pid character (5)	first_name text	last_name text	street_address text	state text	zipcode integer	birthday date	contact_number text	email_addres text
1	p1	Jason	Haley	13 School Street	New ...	11946	1978-11...	4587390869	jason.haley@g.
2	p2	Scott	Fritsch	10 Emerson Co...	New ...	11946	1989-06...	1234567890	scott.fritsch@g.
3	p3	Jami	Domenico	15 Maple Court	New ...	18977	1997-04...	6312546789	jami.domenico.
4	p4	Alan	Laboseur	255 Honey Drive	New ...	12601	1985-09...	1118675301	alan.lab@coolg
5	p5	Jack	Heuber	123 Talk Road	New ...	12445	1998-10...	1345879978	talkingguy@tal.
6	p6	Dave	Connelly	15 Bae Court	Rhod...	12366	1997-11...	1879087890	jefferyjeffery@.
7	p7	Taylor	Connelly	17 Harbor Road	River...	14577	1997-11...	6316805787	taylor.kathryn..
8	p8	John	Sasso	40 Bestfriend L...	New ...	12889	1997-12...	2267897765	john.sasso@be
9	p9	Shannon	Cover	33 Oak Ave	New ...	89059	1990-06...	0987654321	shannon.cover.
10	p10	Sreya	Sobti	1334 Linda Lane	Penn...	37890	1995-10...	7778987654	sreyasobti@ind



Security:

```
create role admin;
grant all on
all tables
in schema public
to admin;
```

Admin: This is either the owner of the business, or a person who would need access to everything within the database.

```
CREATE ROLE hotel_manager;
GRANT SELECT, INSERT, UPDATE
ON ALL TABLES IN SCHEMA PUBLIC
TO hotel_manager;
```

Hotel Manager: The Hotel Manager has much access to the database, as they need to be able to add all types of data into the database.

```
CREATE ROLE front_desk;
GRANT SELECT, INSERT, UPDATE
ON reservations, customer
TO front_desk;
```

Front Desk: The Front Desk needs to be able to access the reservations and customer database, and book the reservations.

```
CREATE  ROLE housekeepers;
GRANT   SELECT  ON  room_status, rooms
TO  housekeepers;
```

Housekeepers: These employees have the least amount of access to the database. They just need to know which rooms need to be cleaned.

Known Problems/Future Enhancements:

- ❏ The sample data for the purposes of this project are limited. Much more data is needed in each of the tables for a thorough understanding of the scope of this database. Since I used a lot of tables, there was a plethora of information that needed to be added to make the database sufficient.
- ❏ I redid my entire ER diagram after I realized that I had repeats of different keys within tables that did not even connect.
- ❏ I had a lot of trouble joining tables because many tables have to dig deeper to get certain information (for example, the first and last name) since only one table holds that information.
- ❏ I had some issues with the foreign keys and primary keys with some of my tables. The tables have to be inserted in the order I submitted in my .sql code.
- ❏ The miscellaneous_charges_add table does not really have a primary key, but I did not know how else to work this.
- ❏ While naming the different IDs, I realized that I was running out of ideas for different number patterns for IDs. If I were to redo this, I would make sure that none of the IDs were without a letter in front. To redo that now would be extremely time consuming and I have internetworking. RIP.