Shannon Brady

Sunset Vacations, NC
Database Design Proposal

Table of Contents

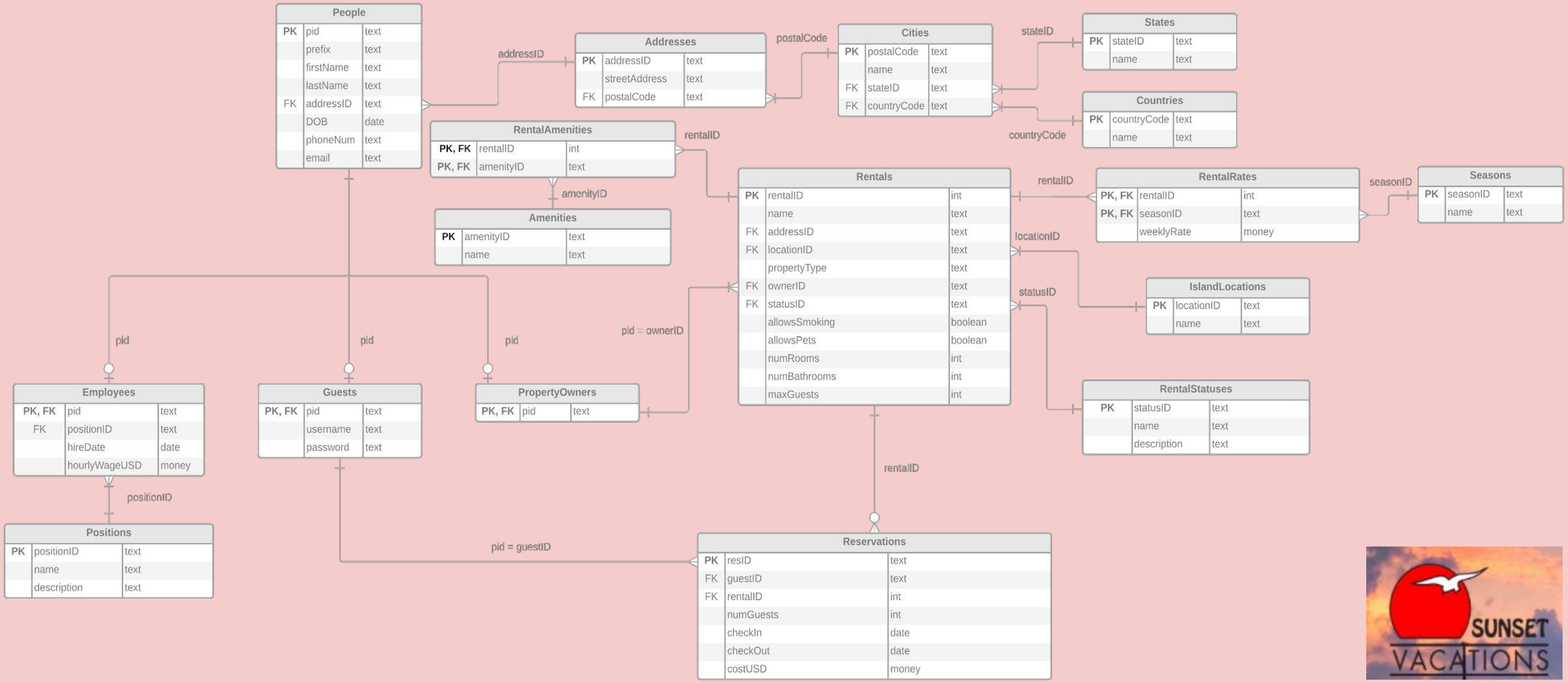
Executive Summary

This database has been created for Sunset Vacations, a family owned vacation rental company located in Sunset Beach, NC. Sunset Vacations has sold real estate and managed vacation rentals since 1984, longer than any other rental management company on the quaint island of Sunset Beach. It maintains the distinct privilege of managing some of the most comfortable and affordable vacation properties, making it an ideal spot for families. This database is designed to manage all aspects necessary to ensure the continued success of this exceptional company.

The following paper has been created to document the extensive uses of this database and provides an intensive review of its implementations. Numerous design components are presented and discussed, including an ER diagram, table create statements, sample data, reports, views, stored procedures, and triggers- all of which have been created and tested for the purposes of improving business functionality.

This paper is intended to consolidate the various aspects of the Sunset Vacations business model and provide a potential database designed with the goal of achieving greater efficiency and efficacy as it relates to company endeavors.

ER Diagram

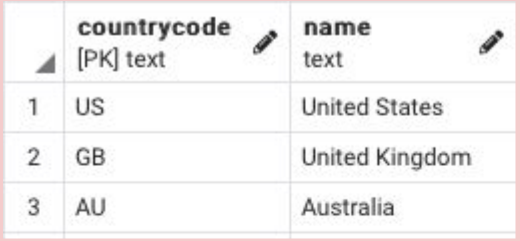


People
PK pid text
prefix text
firstName text
lastName text
FK addressID text
DOB date
phoneNum text
email text
addressID
Addresses
PK addressID text
streetAddress text
FK postalCode text
postalCode
Cities
PK postalCode text
name text
FK stateID text
FK countryCode text
stateID
States
PK stateID text
name text
countryCode
Countries
PK countryCode text
name text
RentalAmenities
PK, FK rentalID int
PK, FK amenityID text
amenityID
Amenities
PK amenityID text
name text
rentalID
Rentals
PK rentalID int
name text
FK addressID text
FK locationID text
propertyType text
FK ownerID text
FK statusID text
allowsSmoking boolean
allowsPets boolean
numRooms int
numBathrooms int
maxGuests int
rentalID
RentalRates
PK, FK rentalID int
PK, FK seasonID text
weeklyRate money
seasonID
Seasons
PK seasonID text
name text
locationID
IslandLocations
PK locationID text
name text
statusID
RentalStatuses
PK statusID text
name text
description text
pid
pid
pid
pid = ownerID
Employees
PK, FK pid text
FK positionID text
hireDate date
hourlyWageUSD money
positionID
Positions
PK positionID text
name text
description text
Guests
PK, FK pid text
username text
password text
PropertyOwners
PK, FK pid text
rentalID
pid = guestID
Reservations
PK resID text
FK guestID text
FK rentalID int
numGuests int
checkIn date
checkOut date
costUSD money

Tables

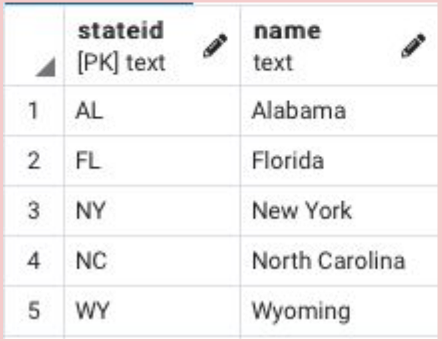
Countries and States

```
CREATE TABLE Countries (
        countryCode     text not null,
        name            text,
primary key(countryCode)
);
```



	countrycode [PK] text	name text
1	US	United States
2	GB	United Kingdom
3	AU	Australia

```
CREATE TABLE States (
        stateID     text not null,
        name        text,
primary key(stateID)
);
```



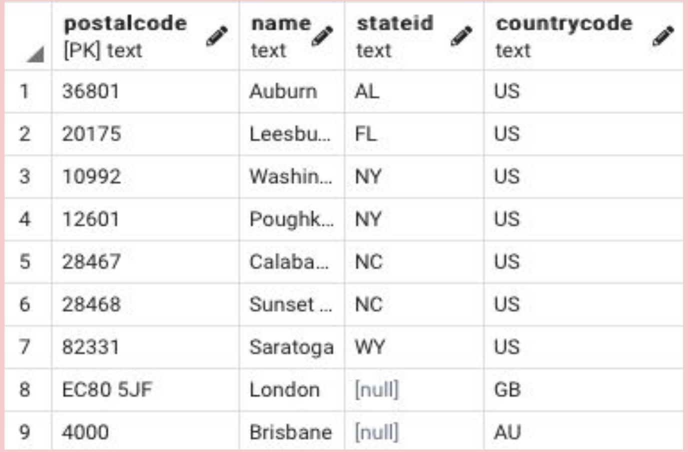
	stateid [PK] text	name text
1	AL	Alabama
2	FL	Florida
3	NY	New York
4	NC	North Carolina
5	WY	Wyoming

These tables contain all countries and states used in the various addresses stored by Sunset Vacations. Both are uniquely identifiable by either a countryCode or a stateID.

Functional Dependencies:
countryCode → name
stateID → name

Cities

```
CREATE TABLE Cities (
        postalCode  text not null,
        name        text,
        stateID             text references States(stateID),
        countryCode         text not null references Countries(countryCode),
primary key(postalCode)
);
```



	postalcode [PK] text	name text	stateid text	countrycode text
1	36801	Auburn	AL	US
2	20175	Leesbu...	FL	US
3	10992	Washin...	NY	US
4	12601	Poughk...	NY	US
5	28467	Calaba...	NC	US
6	28468	Sunset ...	NC	US
7	82331	Saratoga	WY	US
8	EC80 5JF	London	[null]	GB
9	4000	Brisbane	[null]	AU

This table contains all the postal codes relevant for either rental properties or people addresses. For every postal code there is a city name as well as foreign key references to the Countries and States table as necessary.

Functional Dependencies:

postalCode → name, stateID, countryCode

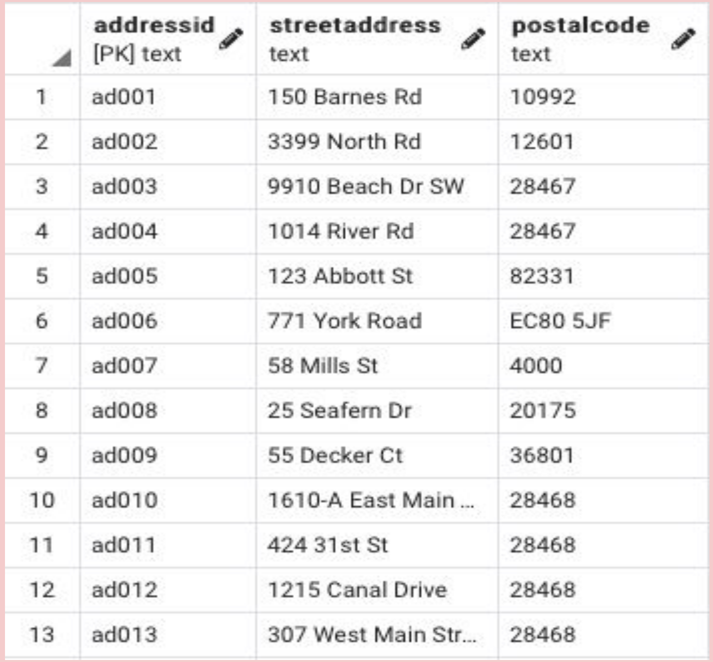
Addresses

```
CREATE TABLE **Addresses** (
        addressID           text not null,
        streetAddress       text,
        postalCode          text not null references Cities(postalCode),
primary key(addressID)
);
```

This table contains all the addresses utilized by Sunset Vacations, whether for the various rental properties or people stored in the database. Every address is uniquely identified by an address id and includes both the street address as well as a postal code that is referenced from the Cities table.

<u>Functional Dependencies:</u>

addressID → streetAddress, postalCode



	addressid [PK] text	streetaddress text	postalcode text
1	ad001	150 Barnes Rd	10992
2	ad002	3399 North Rd	12601
3	ad003	9910 Beach Dr SW	28467
4	ad004	1014 River Rd	28467
5	ad005	123 Abbott St	82331
6	ad006	771 York Road	EC80 5JF
7	ad007	58 Mills St	4000
8	ad008	25 Seafern Dr	20175
9	ad009	55 Decker Ct	36801
10	ad010	1610-A East Main ...	28468
11	ad011	424 31st St	28468
12	ad012	1215 Canal Drive	28468
13	ad013	307 West Main Str...	28468

People

Sample data on next slide

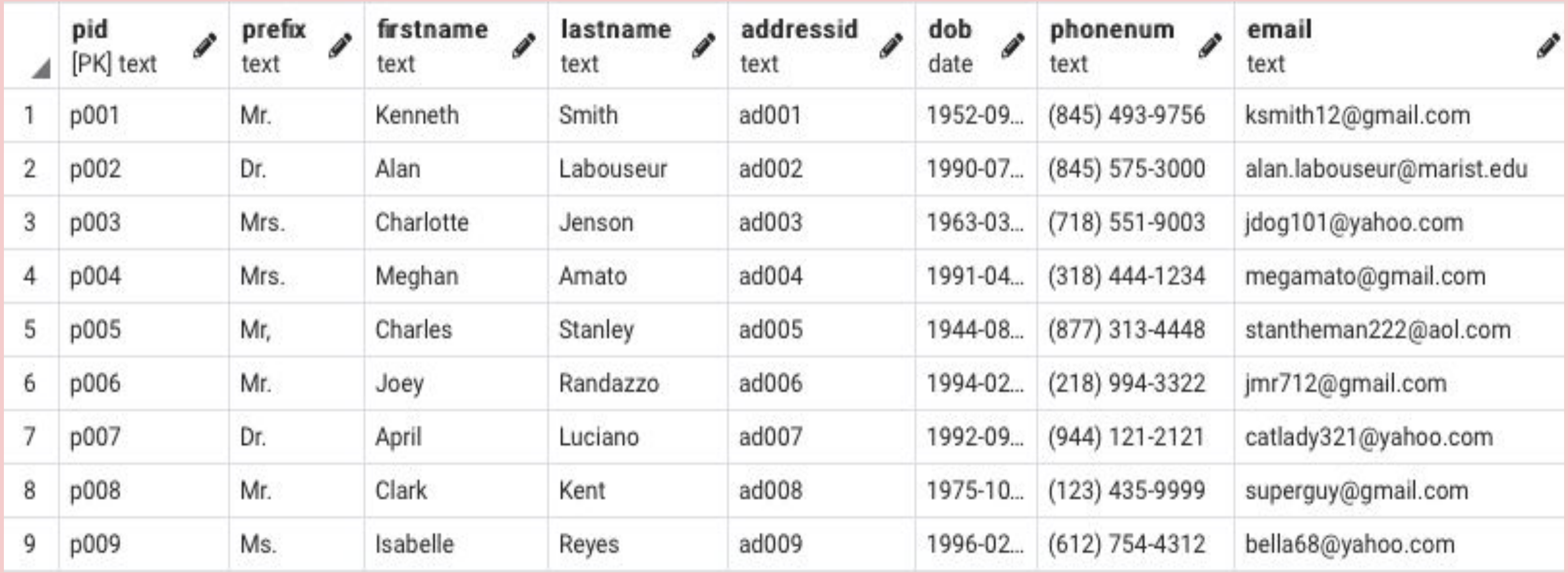
This table identifies all the people and their relevant information stored in the database- whether guests, employees, or property owners. Provides a person’s name, a reference to the Addresses table, date of birth, phone number, and email.

Functional Dependencies:

pid → prefix, firstName, lastName, addressID, DOB, phoneNum, email

```
CREATE TABLE People (
        pid                 text not null,
        prefix              text,
        firstName           text,
        lastName            text,
        addressID           text not null references Addresses(addressID),
        DOB                 date,
        phoneNum            text,
        email               text,
primary key(pid)
);
```

People



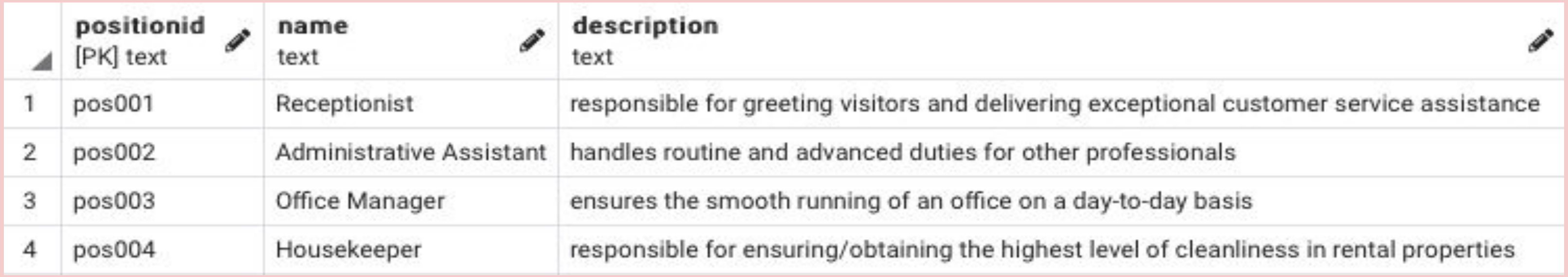
	pid [PK] text	prefix text	firstname text	lastname text	addressid text	dob date	phonenum text	email text
1	p001	Mr.	Kenneth	Smith	ad001	1952-09...	(845) 493-9756	ksmith12@gmail.com
2	p002	Dr.	Alan	Labouseur	ad002	1990-07...	(845) 575-3000	alan.labouseur@marist.edu
3	p003	Mrs.	Charlotte	Jenson	ad003	1963-03...	(718) 551-9003	jdog101@yahoo.com
4	p004	Mrs.	Meghan	Amato	ad004	1991-04...	(318) 444-1234	megamato@gmail.com
5	p005	Mr,	Charles	Stanley	ad005	1944-08...	(877) 313-4448	stantheman222@aol.com
6	p006	Mr.	Joey	Randazzo	ad006	1994-02...	(218) 994-3322	jmr712@gmail.com
7	p007	Dr.	April	Luciano	ad007	1992-09...	(944) 121-2121	catlady321@yahoo.com
8	p008	Mr.	Clark	Kent	ad008	1975-10...	(123) 435-9999	superguy@gmail.com
9	p009	Ms.	Isabelle	Reyes	ad009	1996-02...	(612) 754-4312	bella68@yahoo.com

Positions

```
CREATE TABLE Positions (
        positionID          text not null,
        name                text,
        description         text,
primary key(positionID)
);
```

This table provides all potential employee positions provided by Sunset Vacations. Each position includes a name and relevant description.

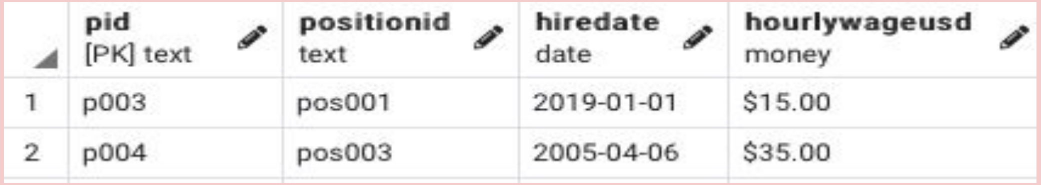
Functional Dependencies:
positionID → name, description



	positionid [PK] text	name text	description text
1	pos001	Receptionist	responsible for greeting visitors and delivering exceptional customer service assistance
2	pos002	Administrative Assistant	handles routine and advanced duties for other professionals
3	pos003	Office Manager	ensures the smooth running of an office on a day-to-day basis
4	pos004	Housekeeper	responsible for ensuring/obtaining the highest level of cleanliness in rental properties

Employees

```
CREATE TABLE Employees (
        pid                    text not null references People(pid),
        positionID             text not null references Positions(positionID),
        hireDate               date,
        hourlyWageUSD          money,
primary key(pid)
);
```



	pid [PK] text	positionid text	hiredate date	hourlywageusd money
1	p003	pos001	2019-01-01	$15.00
2	p004	pos003	2005-04-06	$35.00

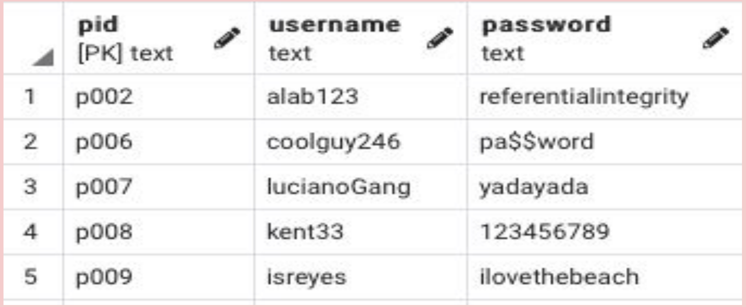
This table contains all Sunset Vacation employees, all of which also exist in the People table. Relevant employee information includes a reference to the Positions table, hire date, and hourly wage.

<u>Functional Dependencies:</u>

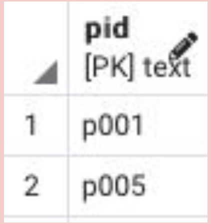
pid → positionID, hireDate, hourlyWageUSD

Guests

```
CREATE TABLE Guests (
        pid             text not null references People(pid),
        username        text not null,
        password        text not null,
primary key(pid)
);

CREATE TABLE PropertyOwners (
        pid             text not null references People(pid),
primary key(pid)
);
```



	pid [PK] text	username text	password text
1	p002	alab123	referentialintegrity
2	p006	coolguy246	pa$$word
3	p007	lucianoGang	yadayada
4	p008	kent33	123456789
5	p009	isreyes	ilovethebeach



	pid [PK] text
1	p001
2	p005

The Guests table identifies all Sunset Beach guests, with reservations either in the past, present, or future. Every guest must have a username and password to make a reservation. The PropertyOwners table identifies all individuals who own a property rented out by Sunset Vacations. All guests and property owners also exist in the People table.

Functional Dependencies:

(Guests) pid → username, password

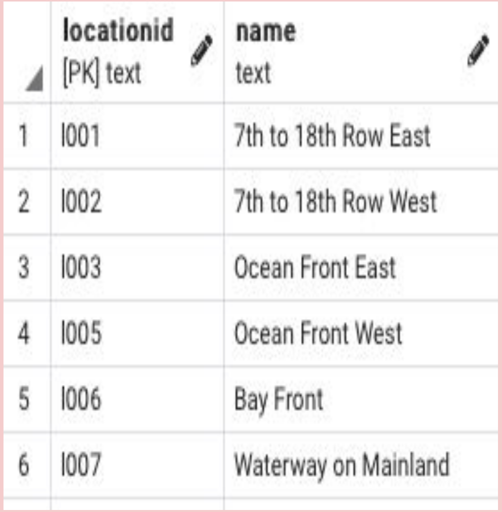
(PropertyOwners) pid →

IslandLocations

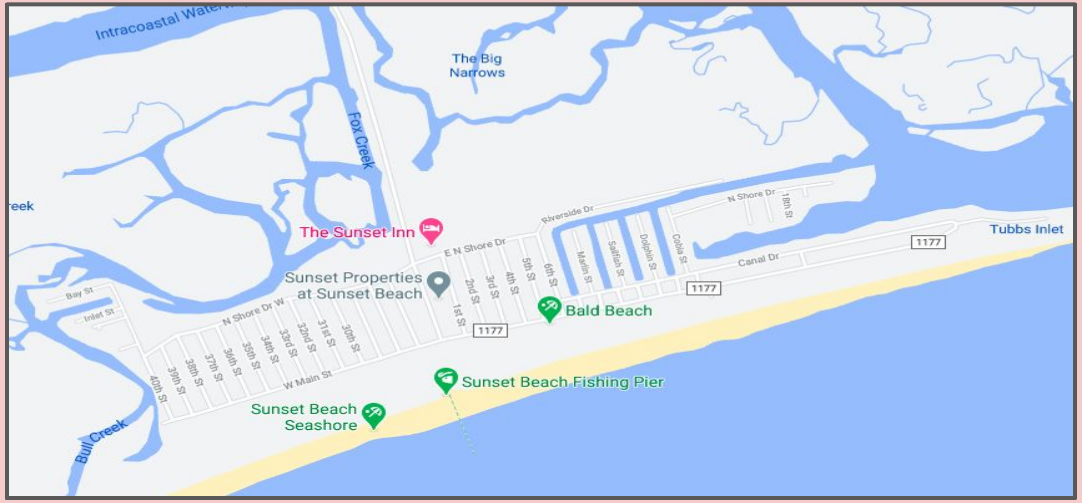
```
CREATE TABLE IslandLocations (
        locationID          text not null,
        name                text,
primary key(locationID)
);
```

This table identifies and names all areas of Sunset Beach where rental properties are located. These locations are intended to give potential guests a greater understanding of island layout when making reservations.

Functional Dependencies:
locationID → name



	locationid [PK] text	name text
1	l001	7th to 18th Row East
2	l002	7th to 18th Row West
3	l003	Ocean Front East
4	l005	Ocean Front West
5	l006	Bay Front
6	l007	Waterway on Mainland

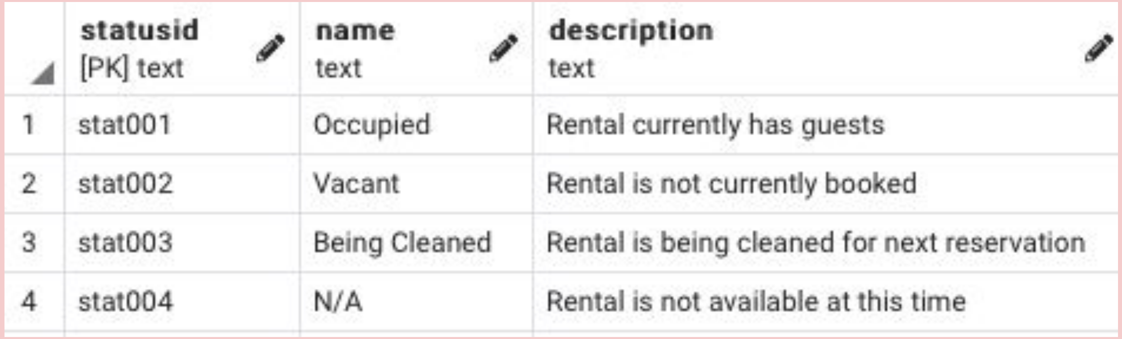


RentalStatuses

```
CREATE TABLE RentalStatuses (
        statusID       text not null,
        name           text,
        description    text,
primary key(statusID)
);
```

This table identifies all potential rental statuses, including whether occupied, vacant, being cleaned, or currently unavailable.

<u>Functional Dependencies:</u>
statusID → name, description



	statusid [PK] text	name text	description text
1	stat001	Occupied	Rental currently has guests
2	stat002	Vacant	Rental is not currently booked
3	stat003	Being Cleaned	Rental is being cleaned for next reservation
4	stat004	N/A	Rental is not available at this time

Rentals

```
CREATE TABLE Rentals (
        rentalID        int not null,
        name            text,
        addressID       text not null references Addresses(addressID),
        locationID      text not null references IslandLocations(locationID),
        propertyType    text,
        ownerID         text not null references PropertyOwners(pid),
        statusID        text not null references RentalStatuses(statusID),
        allowsSmoking   boolean,
        allowsPets      boolean,
        numRooms        int,
        numBathrooms    int,
        maxGuests       int,
primary key(rentalID)
);
```

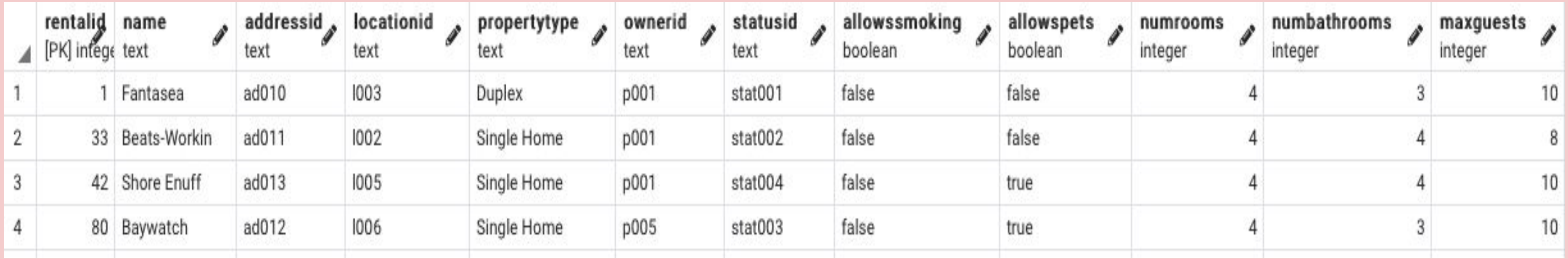
Sample data on next slide

This table contains all the information relating to each rental property managed by Sunset Vacations.

<u>Functional Dependencies:</u>

rentalID → name, addressID, locationID, propertyType, ownerID, statusID, allowsSmoking, allowsPets, numRooms, numBathrooms, maxGuests

Rentals



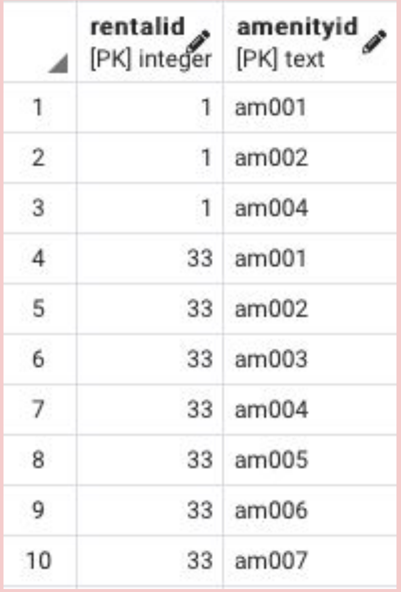
	rentalid [PK] intege	name text	addressid text	locationid text	propertytype text	ownerid text	statusid text	allowssmoking boolean	allowspets boolean	numrooms integer	numbathrooms integer	maxguests integer
1	1	Fantasea	ad010	l003	Duplex	p001	stat001	false	false	4	3	10
2	33	Beats-Workin	ad011	l002	Single Home	p001	stat002	false	false	4	4	8
3	42	Shore Enuff	ad013	l005	Single Home	p001	stat004	false	true	4	4	10
4	80	Baywatch	ad012	l006	Single Home	p005	stat003	false	true	4	3	10

Amenities & RentalAmenities

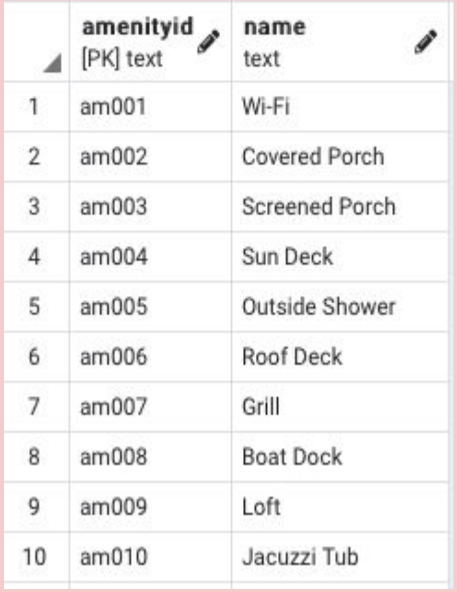
```
CREATE TABLE Amenities (
        amenityID    text not null,
        name         text,
primary key(amenityID)
);

CREATE TABLE RentalAmenities (
        rentalID      int  not null references Rentals(rentalID),
        amenityID    text not null references Amenities(amenityID),
primary key(rentalID, amenityID)
);
```

The Amenities table includes all the amenities that Sunset Vacation offers. RentalAmenities maps the amenities that are available in each rental.

Functional Dependencies:
amenityID → name
rentalID, amenityID →



	rentalid [PK] integer	amenityid [PK] text
1	1	am001
2	1	am002
3	1	am004
4	33	am001
5	33	am002
6	33	am003
7	33	am004
8	33	am005
9	33	am006
10	33	am007

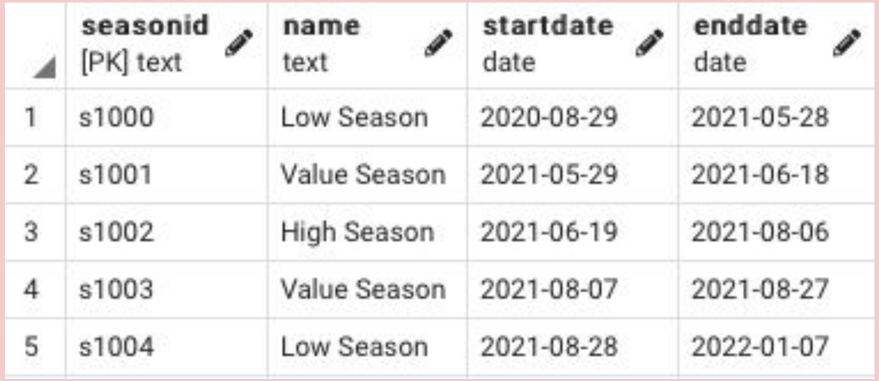


	amenityid [PK] text	name text
1	am001	Wi-Fi
2	am002	Covered Porch
3	am003	Screened Porch
4	am004	Sun Deck
5	am005	Outside Shower
6	am006	Roof Deck
7	am007	Grill
8	am008	Boat Dock
9	am009	Loft
10	am010	Jacuzzi Tub

Portion of sample data

Seasons

```
CREATE TABLE Seasons (
        seasonID    text not null,
        name        text,
        startDate   date,
        endDate     date,
primary key(seasonID)
);
```



	seasonid [PK] text	name text	startdate date	enddate date
1	s1000	Low Season	2020-08-29	2021-05-28
2	s1001	Value Season	2021-05-29	2021-06-18
3	s1002	High Season	2021-06-19	2021-08-06
4	s1003	Value Season	2021-08-07	2021-08-27
5	s1004	Low Season	2021-08-28	2022-01-07

Rental pricing is adjusted according to the time of year it is reserved. This table identifies these different time periods, as well as the start date and end date of each.

Functional Dependencies:

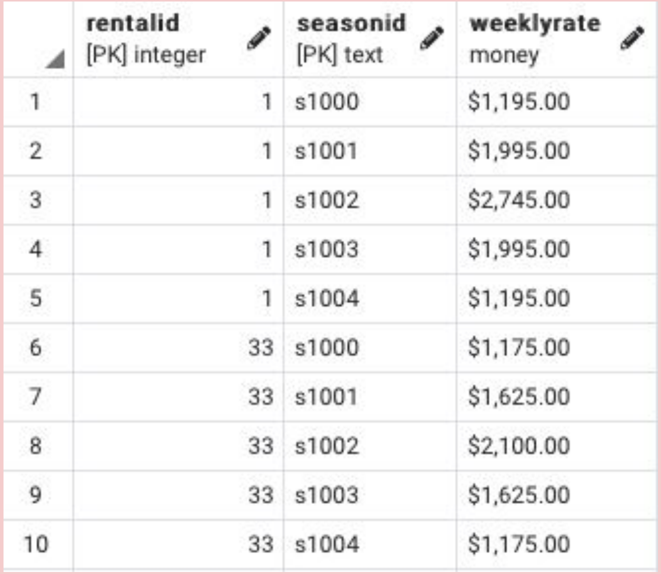
seasonID → name, startDate, endDate

RentalRates

```
CREATE TABLE RentalRates (
        rentalID          int not null references Rentals(rentalID),
        seasonID          text not null references Seasons(seasonID),
        weeklyRate      money,
primary key(rentalID, seasonID)
);
```

This table identifies the weekly rate of any given rental according to each season. RentalID is a reference to the Rentals table and seasonID is a reference to the Seasons table.

<u>Functional Dependencies:</u>
rentalID, seasonID → weeklyRate

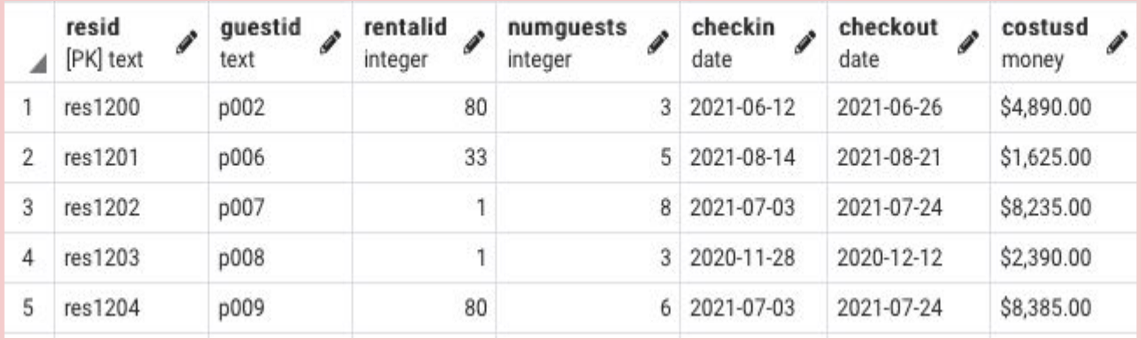


	rentalid [PK] integer	seasonid [PK] text	weeklyrate money
1	1	s1000	$1,195.00
2	1	s1001	$1,995.00
3	1	s1002	$2,745.00
4	1	s1003	$1,995.00
5	1	s1004	$1,195.00
6	33	s1000	$1,175.00
7	33	s1001	$1,625.00
8	33	s1002	$2,100.00
9	33	s1003	$1,625.00
10	33	s1004	$1,175.00

Portion of sample data

Reservations

```
CREATE TABLE Reservations (
        resID           text not null,
        guestID         text not null references Guests(pid),
        rentalID        int  not null references Rentals(rentalID),
        numGuests       int,
        checkIn         date  not null,
        checkOut        date  not null,
        costUSD         money not null,
primary key(resID)
);
```



	resid [PK] text	guestid text	rentalid integer	numguests integer	checkin date	checkout date	costusd money
1	res1200	p002	80	3	2021-06-12	2021-06-26	$4,890.00
2	res1201	p006	33	5	2021-08-14	2021-08-21	$1,625.00
3	res1202	p007	1	8	2021-07-03	2021-07-24	$8,235.00
4	res1203	p008	1	3	2020-11-28	2020-12-12	$2,390.00
5	res1204	p009	80	6	2021-07-03	2021-07-24	$8,385.00

This table identifies all reservations and their relevant information, including guestID, rentalID, number of guests, check-in date, check-out date, and total cost. Cost is calculated purely by the weekly rate identified in the RentalRates table.

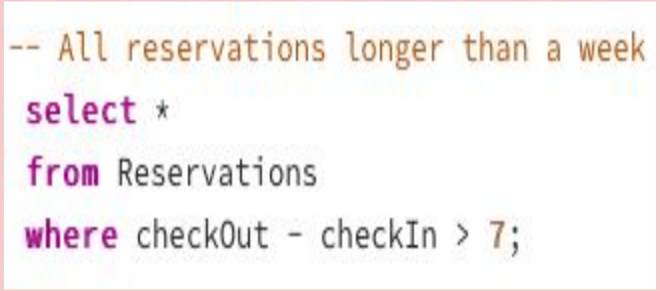
Functional Dependencies:

resID → redID, guestID, rentalID, numGuests, checkIn, checkOut, costUSD

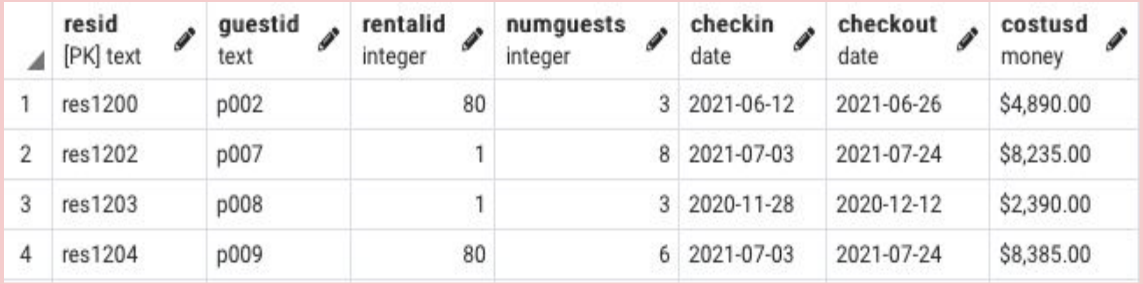
SUNSET VACATIONS

Reports, Views, Triggers, Stored Procedures, and Security

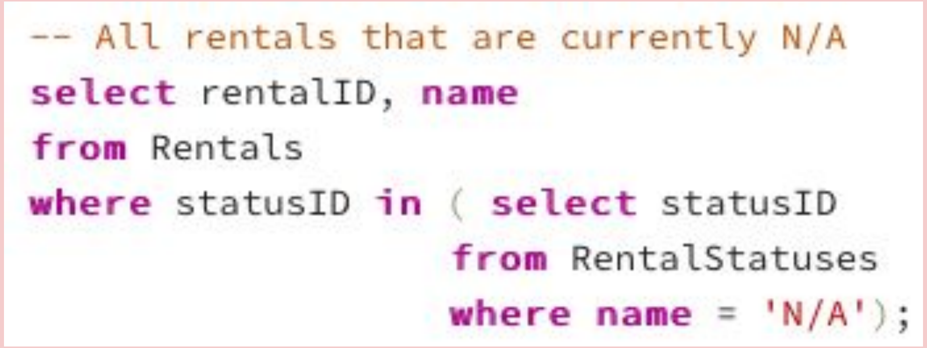
Reports



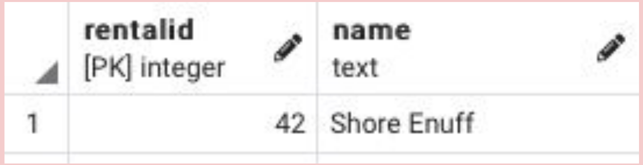
```sql
-- All reservations longer than a week
 select *
 from Reservations
 where checkOut - checkIn > 7;
```



	resid [PK] text	guestid text	rentalid integer	numguests integer	checkin date	checkout date	costusd money
1	res1200	p002	80	3	2021-06-12	2021-06-26	$4,890.00
2	res1202	p007	1	8	2021-07-03	2021-07-24	$8,235.00
3	res1203	p008	1	3	2020-11-28	2020-12-12	$2,390.00
4	res1204	p009	80	6	2021-07-03	2021-07-24	$8,385.00

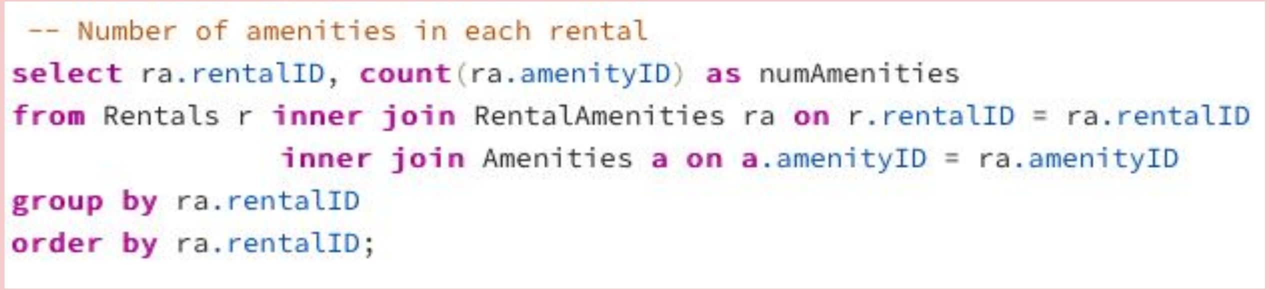


```sql
-- All rentals that are currently N/A
select rentalID, name
from Rentals
where statusID in ( select statusID
                    from RentalStatuses
                    where name = 'N/A');
```

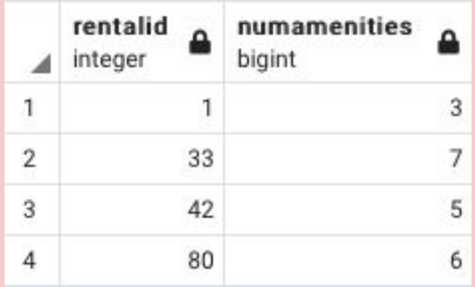


	rentalid [PK] integer	name text
1	42	Shore Enuff

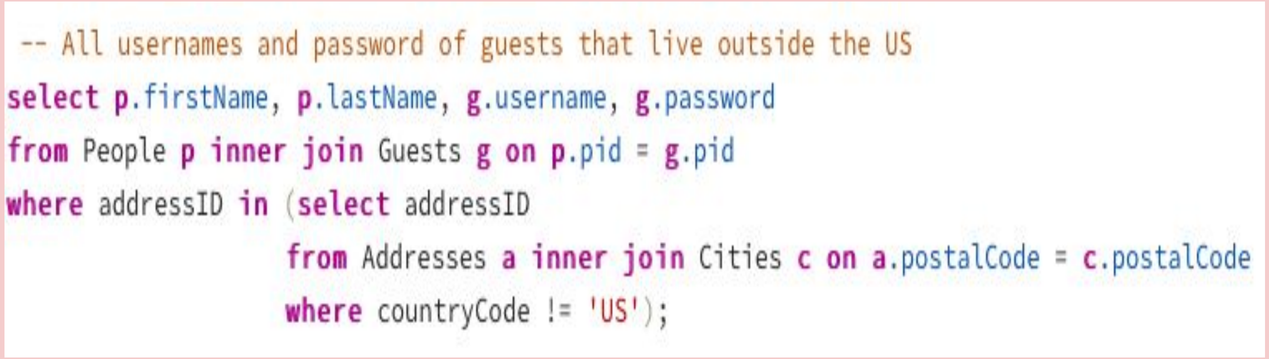
Reports



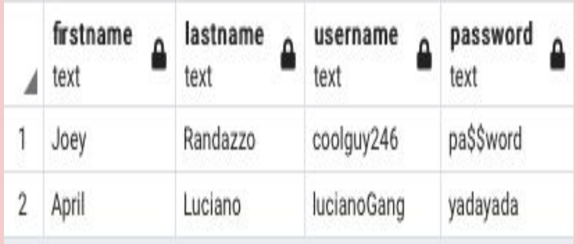
```
 -- Number of amenities in each rental
select ra.rentalID, count(ra.amenityID) as numAmenities
from Rentals r inner join RentalAmenities ra on r.rentalID = ra.rentalID
               inner join Amenities a on a.amenityID = ra.amenityID
group by ra.rentalID
order by ra.rentalID;
```



	rentalid integer	numamenities bigint
1	1	3
2	33	7
3	42	5
4	80	6

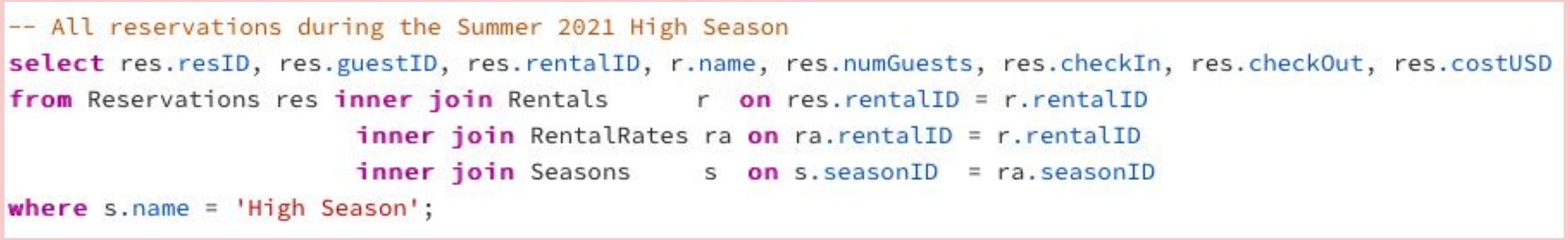


```
 -- All usernames and password of guests that live outside the US
select p.firstName, p.lastName, g.username, g.password
from People p inner join Guests g on p.pid = g.pid
where addressID in (select addressID
                    from Addresses a inner join Cities c on a.postalCode = c.postalCode
                    where countryCode != 'US');
```

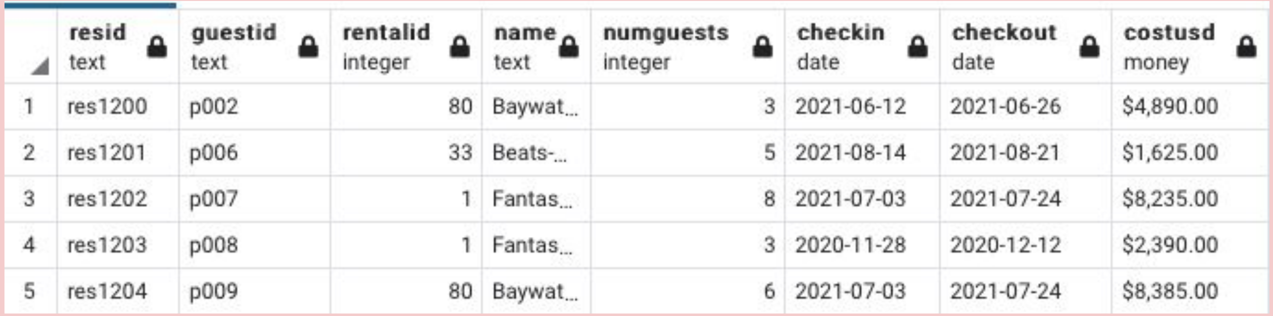


	firstname text	lastname text	username text	password text
1	Joey	Randazzo	coolguy246	pa$$word
2	April	Luciano	lucianoGang	yadayada

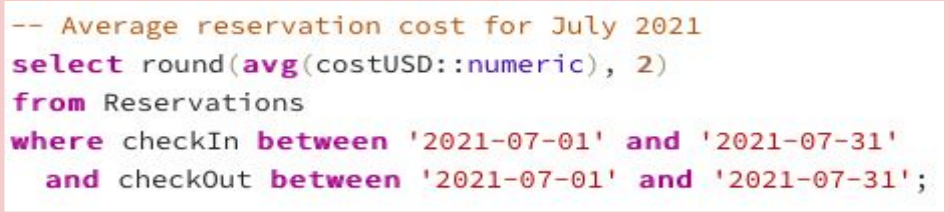
Reports



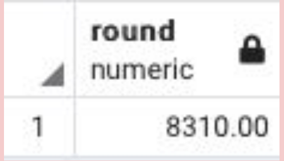
```sql
-- All reservations during the Summer 2021 High Season
select res.resID, res.guestID, res.rentalID, r.name, res.numGuests, res.checkIn, res.checkOut, res.costUSD
from Reservations res inner join Rentals     r  on res.rentalID = r.rentalID
                      inner join RentalRates ra on ra.rentalID = r.rentalID
                      inner join Seasons     s  on s.seasonID  = ra.seasonID
where s.name = 'High Season';
```



	resid text	guestid text	rentalid integer	name text	numguests integer	checkin date	checkout date	costusd money
1	res1200	p002	80	Baywat...	3	2021-06-12	2021-06-26	$4,890.00
2	res1201	p006	33	Beats-...	5	2021-08-14	2021-08-21	$1,625.00
3	res1202	p007	1	Fantas...	8	2021-07-03	2021-07-24	$8,235.00
4	res1203	p008	1	Fantas...	3	2020-11-28	2020-12-12	$2,390.00
5	res1204	p009	80	Baywat...	6	2021-07-03	2021-07-24	$8,385.00

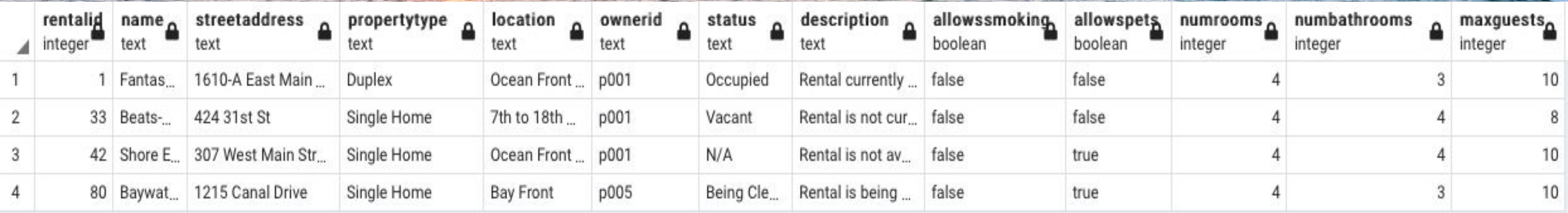


```sql
-- Average reservation cost for July 2021
select round(avg(costUSD::numeric), 2)
from Reservations
where checkIn between '2021-07-01' and '2021-07-31'
  and checkOut between '2021-07-01' and '2021-07-31';
```

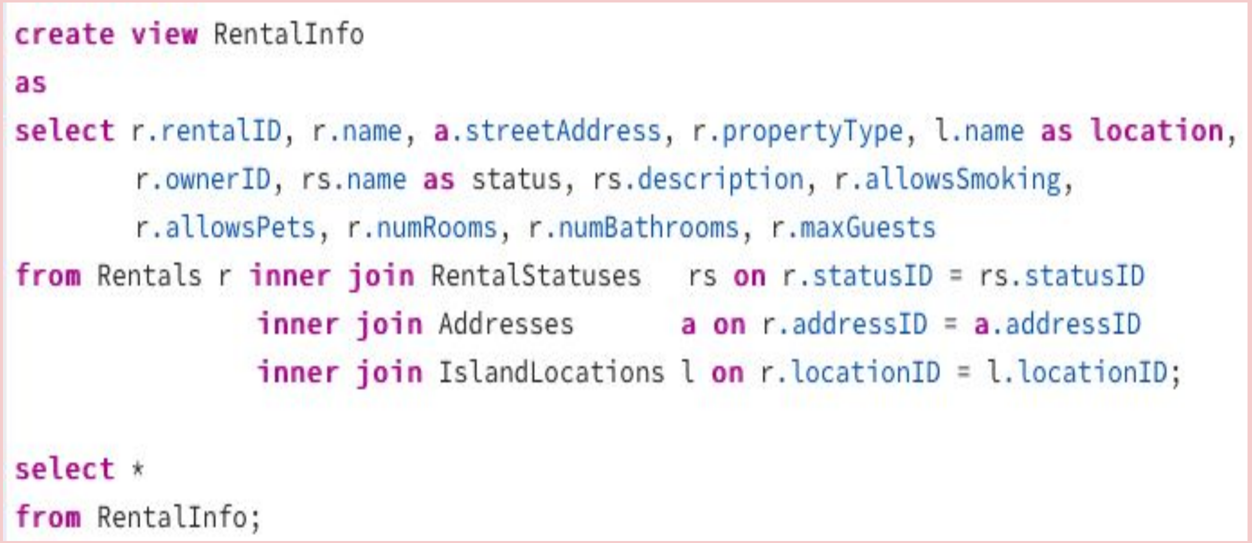


	round numeric
1	8310.00

Views: RentalInfo



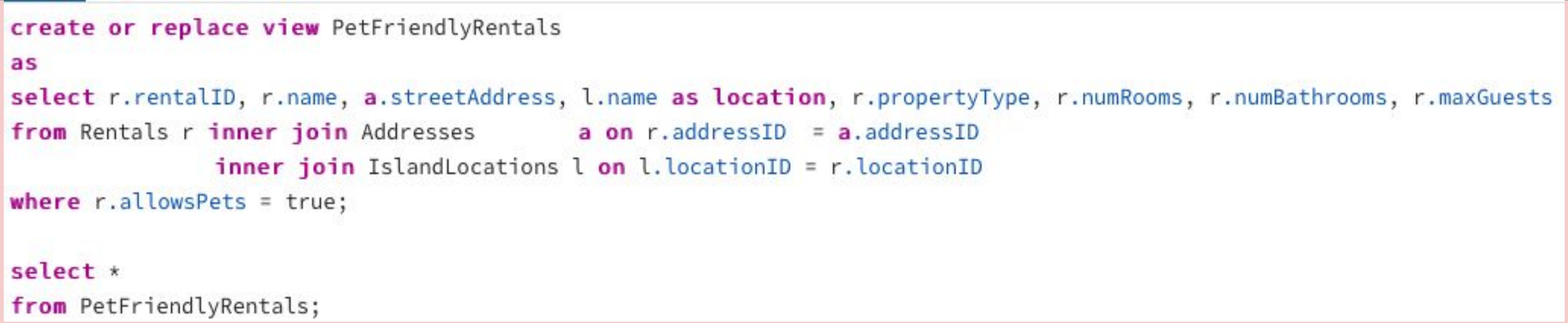
	rentalid integer	name text	streetaddress text	propertytype text	location text	ownerid text	status text	description text	allowssmoking boolean	allowspets boolean	numrooms integer	numbathrooms integer	maxguests integer
1	1	Fantas...	1610-A East Main ...	Duplex	Ocean Front ...	p001	Occupied	Rental currently ...	false	false	4	3	10
2	33	Beats-...	424 31st St	Single Home	7th to 18th ...	p001	Vacant	Rental is not cur...	false	false	4	4	8
3	42	Shore E...	307 West Main Str...	Single Home	Ocean Front ...	p001	N/A	Rental is not av...	false	true	4	4	10
4	80	Baywat...	1215 Canal Drive	Single Home	Bay Front	p005	Being Cle...	Rental is being ...	false	true	4	3	10



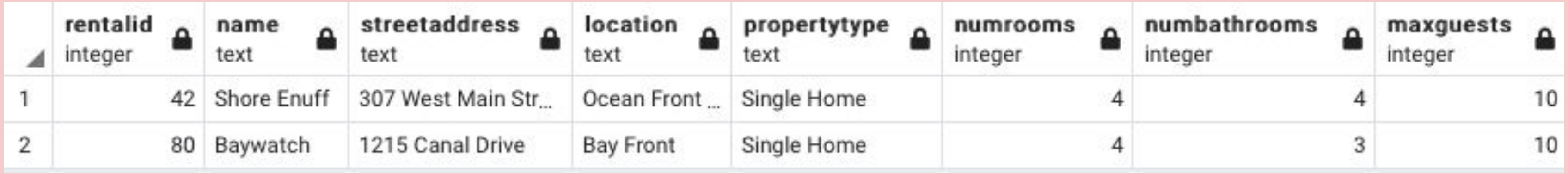
```
create view RentalInfo
as
select r.rentalID, r.name, a.streetAddress, r.propertyType, l.name as location,
        r.ownerID, rs.name as status, rs.description, r.allowsSmoking,
        r.allowsPets, r.numRooms, r.numBathrooms, r.maxGuests
from Rentals r inner join RentalStatuses   rs on r.statusID = rs.statusID
                inner join Addresses        a on r.addressID = a.addressID
                inner join IslandLocations l on r.locationID = l.locationID;

select *
from RentalInfo;
```

Identifies important rental information stored in various other tables.

Views: PetFriendlyRentals



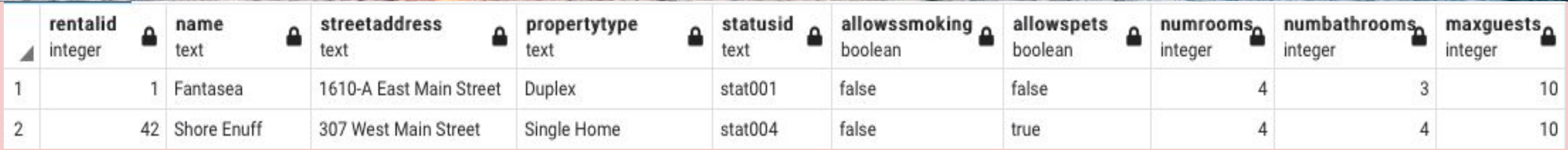
```
create or replace view PetFriendlyRentals
as
select r.rentalID, r.name, a.streetAddress, l.name as location, r.propertyType, r.numRooms, r.numBathrooms, r.maxGuests
from Rentals r inner join Addresses        a on r.addressID  = a.addressID
               inner join IslandLocations l on l.locationID = r.locationID
where r.allowsPets = true;

select *
from PetFriendlyRentals;
```



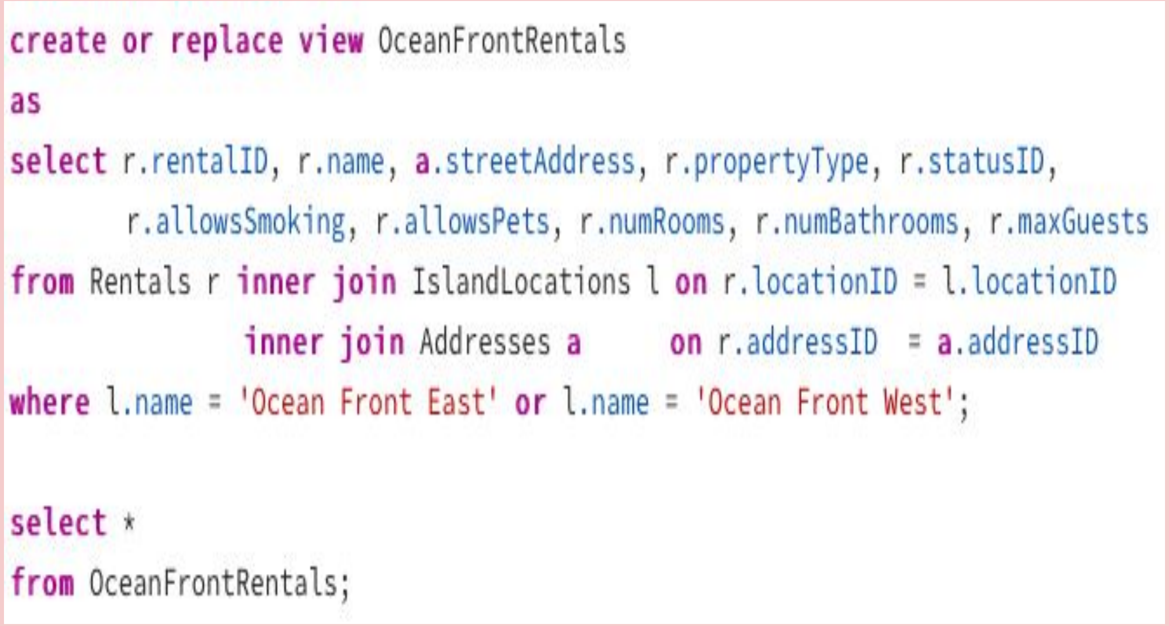
	rentalid integer	name text	streetaddress text	location text	propertytype text	numrooms integer	numbathrooms integer	maxguests integer
1	42	Shore Enuff	307 West Main Str...	Ocean Front ...	Single Home	4	4	10
2	80	Baywatch	1215 Canal Drive	Bay Front	Single Home	4	3	10

Identifies all rentals that allow pets, as well as other relevant rental information.

Views: OceanFrontRentals



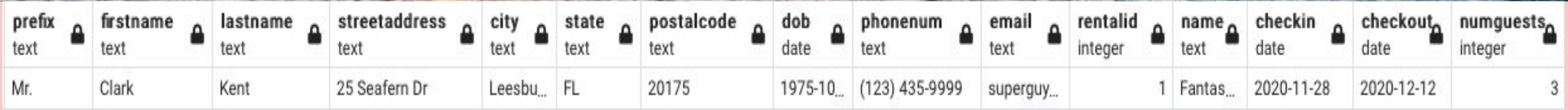
	rentalid integer	name text	streetaddress text	propertytype text	statusid text	allowssmoking boolean	allowspets boolean	numrooms integer	numbathrooms integer	maxguests integer
1	1	Fantasea	1610-A East Main Street	Duplex	stat001	false	false	4	3	10
2	42	Shore Enuff	307 West Main Street	Single Home	stat004	false	true	4	4	10



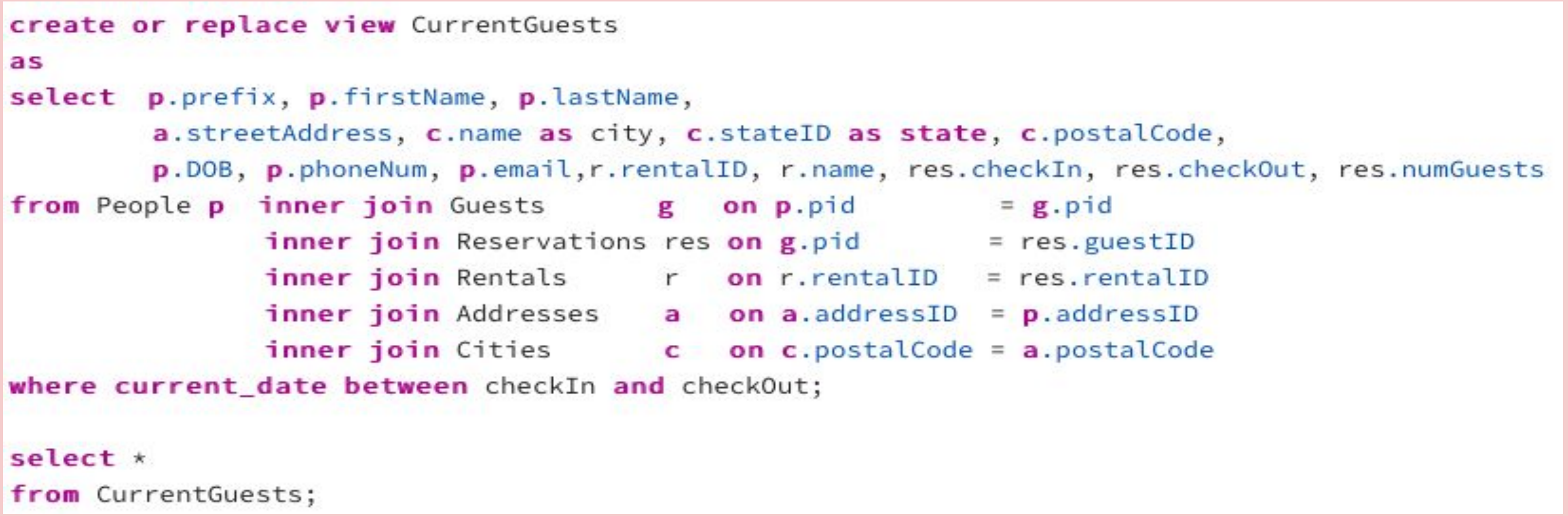
```
create or replace view OceanFrontRentals
as
select r.rentalID, r.name, a.streetAddress, r.propertyType, r.statusID,
       r.allowsSmoking, r.allowsPets, r.numRooms, r.numBathrooms, r.maxGuests
from Rentals r inner join IslandLocations l on r.locationID = l.locationID
               inner join Addresses a      on r.addressID  = a.addressID
where l.name = 'Ocean Front East' or l.name = 'Ocean Front West';

select *
from OceanFrontRentals;
```

Given the fact that oceanfront rentals are usually very popular, this view identifies all rentals that fit such criteria, in addition to other relevant rental information.

Views: Current Guests



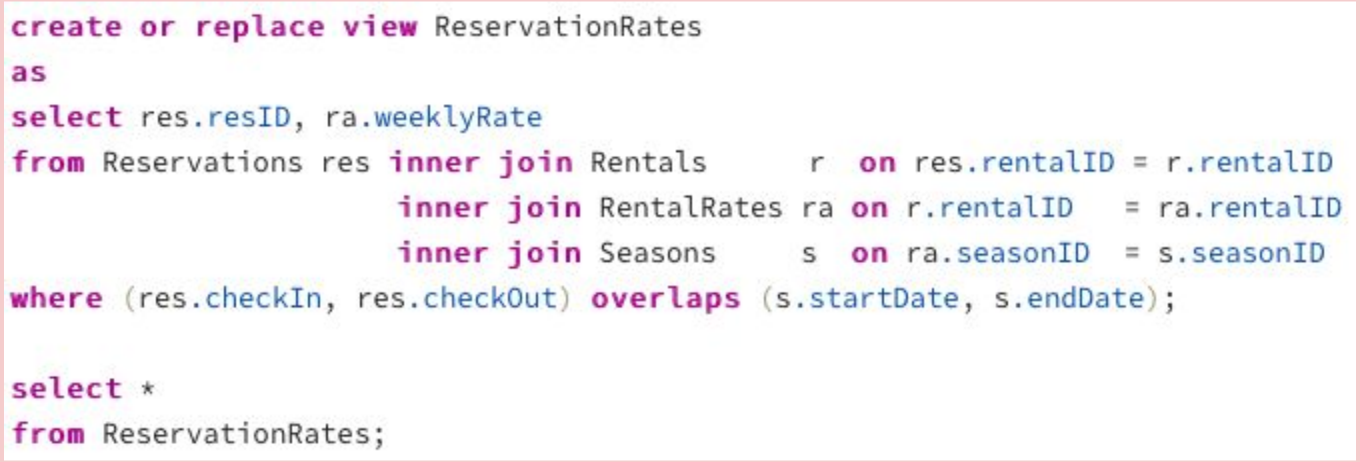
prefix text	firstname text	lastname text	streetaddress text	city text	state text	postalcode text	dob date	phonenum text	email text	rentalid integer	name text	checkin date	checkout date	numguests integer
Mr.	Clark	Kent	25 Seafern Dr	Leesbu...	FL	20175	1975-10...	(123) 435-9999	superguy...	1	Fantas...	2020-11-28	2020-12-12	3



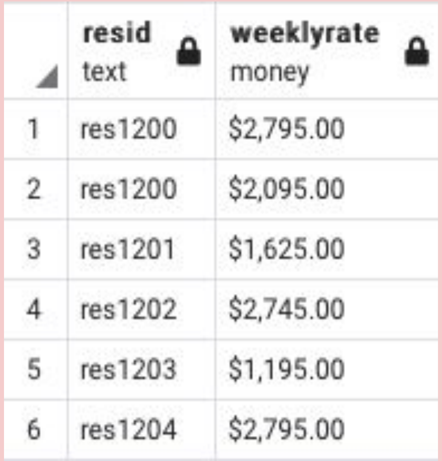
```
create or replace view CurrentGuests
as
select  p.prefix, p.firstName, p.lastName,
        a.streetAddress, c.name as city, c.stateID as state, c.postalCode,
        p.DOB, p.phoneNum, p.email,r.rentalID, r.name, res.checkIn, res.checkOut, res.numGuests
from People p  inner join Guests       g   on p.pid         = g.pid
               inner join Reservations res on g.pid         = res.guestID
               inner join Rentals      r   on r.rentalID    = res.rentalID
               inner join Addresses    a   on a.addressID   = p.addressID
               inner join Cities       c   on c.postalCode = a.postalCode
where current_date between checkIn and checkOut;

select *
from CurrentGuests;
```

This view identifies all guests that are currently staying in a Sunset Vacations rental, as well as their personal information, rental identification, and reservation information.

Views: ReservationRates



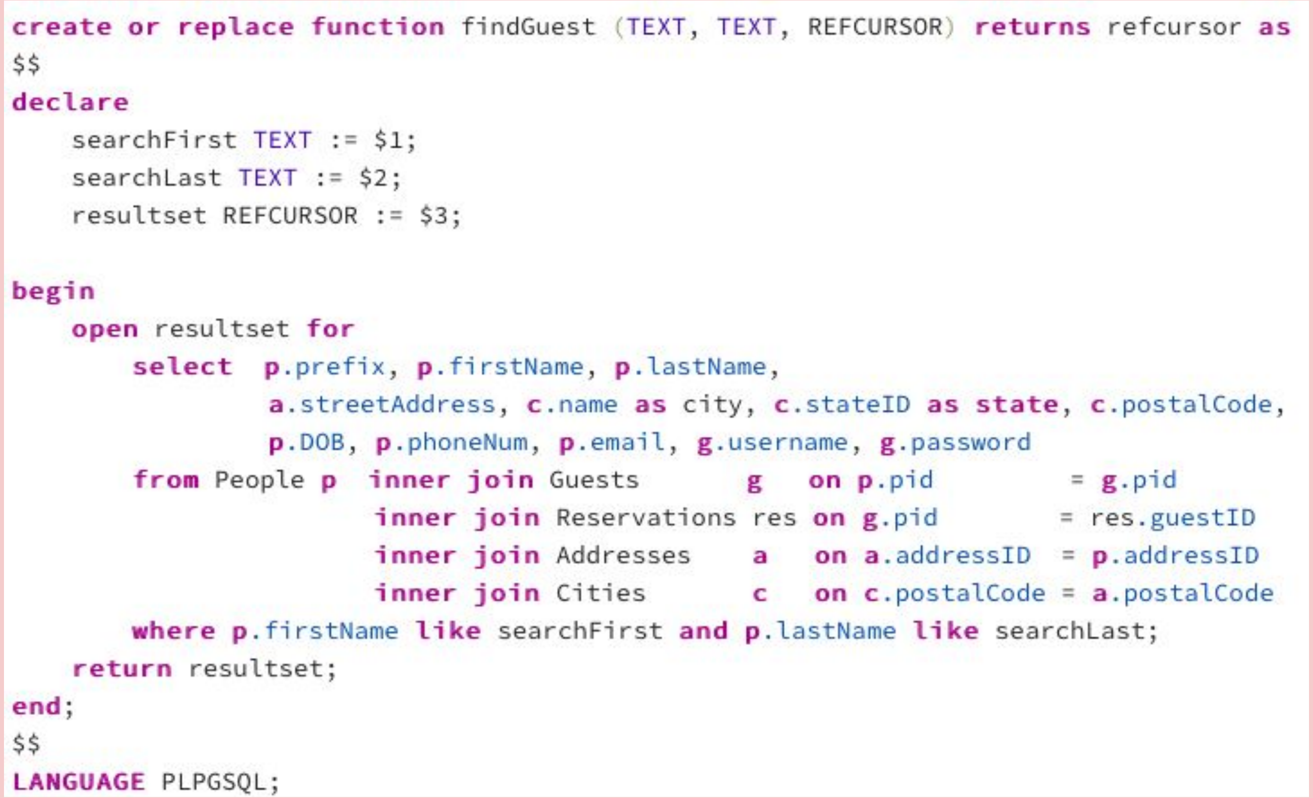
```
create or replace view ReservationRates
as
select res.resID, ra.weeklyRate
from Reservations res inner join Rentals      r  on res.rentalID = r.rentalID
                      inner join RentalRates ra on r.rentalID   = ra.rentalID
                      inner join Seasons     s  on ra.seasonID  = s.seasonID
where (res.checkIn, res.checkOut) overlaps (s.startDate, s.endDate);

select *
from ReservationRates;
```



	resid text	weeklyrate money
1	res1200	$2,795.00
2	res1200	$2,095.00
3	res1201	$1,625.00
4	res1202	$2,745.00
5	res1203	$1,195.00
6	res1204	$2,795.00

This view identifies each reservation and the rate for each/all week(s) between check-in and check-out.

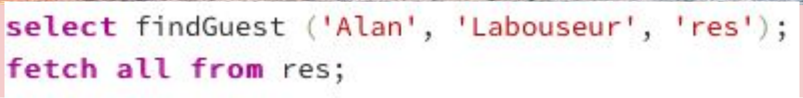
Stored Procedures: findGuest



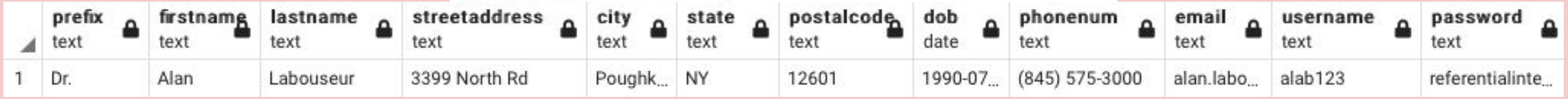
```
create or replace function findGuest (TEXT, TEXT, REFCURSOR) returns refcursor as
$$
declare
    searchFirst TEXT := $1;
    searchLast TEXT := $2;
    resultset REFCURSOR := $3;

begin
    open resultset for
        select  p.prefix, p.firstName, p.lastName,
                a.streetAddress, c.name as city, c.stateID as state, c.postalCode,
                p.DOB, p.phoneNum, p.email, g.username, g.password
        from People p  inner join Guests       g   on p.pid         = g.pid
                       inner join Reservations res on g.pid         = res.guestID
                       inner join Addresses    a   on a.addressID   = p.addressID
                       inner join Cities       c   on c.postalCode  = a.postalCode
        where p.firstName like searchFirst and p.lastName like searchLast;
    return resultset;
end;
$$
LANGUAGE PLPGSQL;
```

This function allows employees to search for guest information based on an element of the guest’s first name, last name, or both.

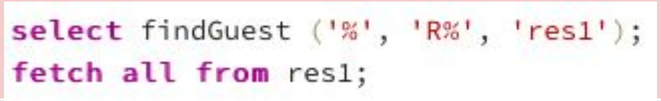
Testing: findGuest



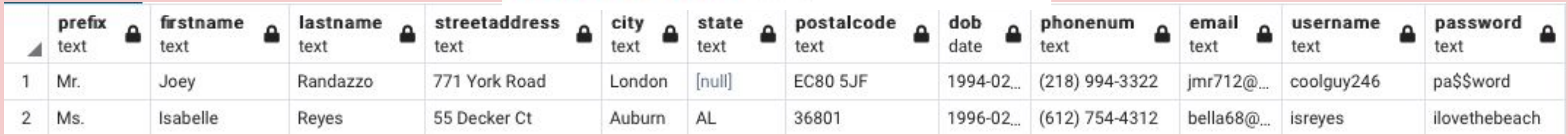
```sql
select findGuest ('Alan', 'Labouseur', 'res');
fetch all from res;
```



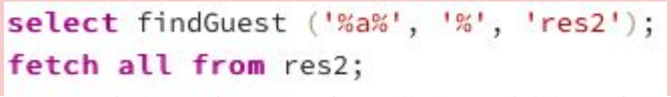
	prefix text	firstname text	lastname text	streetaddress text	city text	state text	postalcode text	dob date	phonenum text	email text	username text	password text
1	Dr.	Alan	Labouseur	3399 North Rd	Poughk...	NY	12601	1990-07...	(845) 575-3000	alan.labo...	alab123	referentialinte...



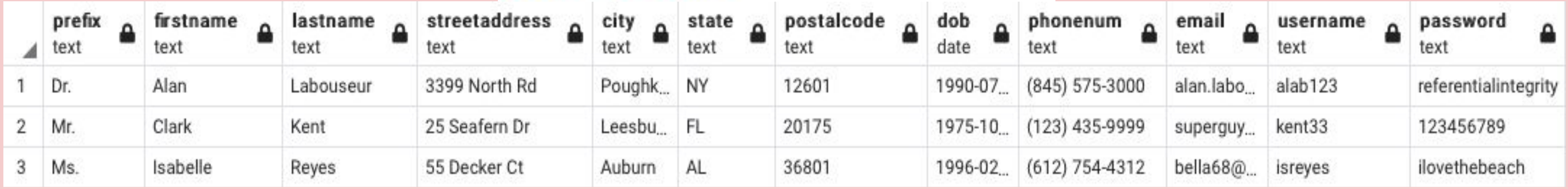
```sql
select findGuest ('%', 'R%', 'res1');
fetch all from res1;
```



	prefix text	firstname text	lastname text	streetaddress text	city text	state text	postalcode text	dob date	phonenum text	email text	username text	password text
1	Mr.	Joey	Randazzo	771 York Road	London	[null]	EC80 5JF	1994-02...	(218) 994-3322	jmr712@...	coolguy246	pa$$word
2	Ms.	Isabelle	Reyes	55 Decker Ct	Auburn	AL	36801	1996-02...	(612) 754-4312	bella68@...	isreyes	ilovethebeach

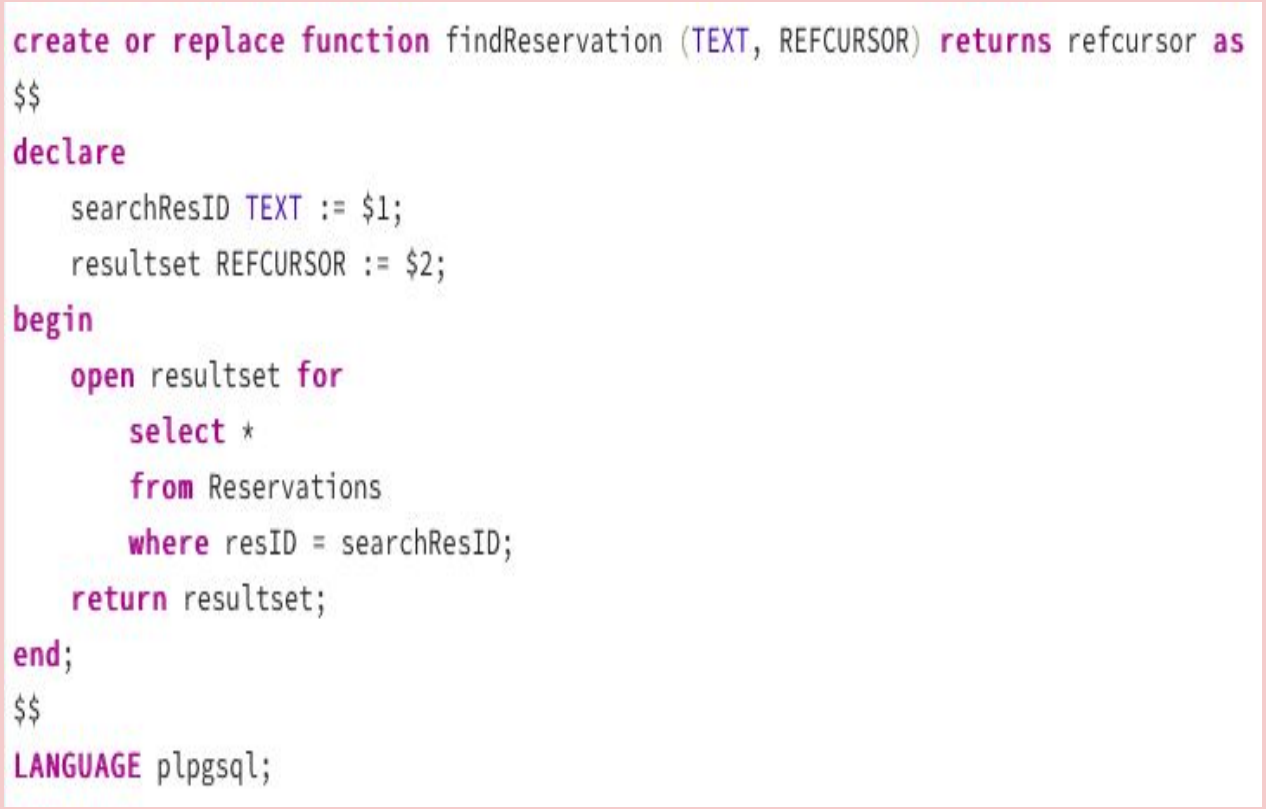


```sql
select findGuest ('%a%', '%', 'res2');
fetch all from res2;
```



	prefix text	firstname text	lastname text	streetaddress text	city text	state text	postalcode text	dob date	phonenum text	email text	username text	password text
1	Dr.	Alan	Labouseur	3399 North Rd	Poughk...	NY	12601	1990-07...	(845) 575-3000	alan.labo...	alab123	referentialintegrity
2	Mr.	Clark	Kent	25 Seafern Dr	Leesbu...	FL	20175	1975-10...	(123) 435-9999	superguy...	kent33	123456789
3	Ms.	Isabelle	Reyes	55 Decker Ct	Auburn	AL	36801	1996-02...	(612) 754-4312	bella68@...	isreyes	ilovethebeach

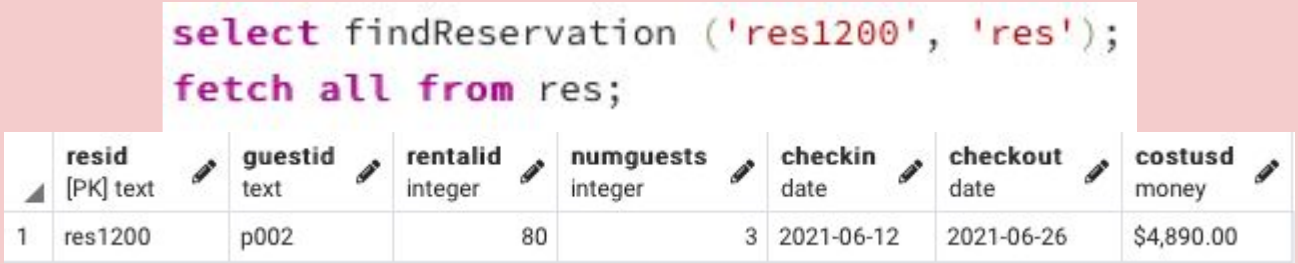
Stored Procedures: findReservation



```
create or replace function findReservation (TEXT, REFCURSOR) returns refcursor as
$$
declare
    searchResID TEXT := $1;
    resultset REFCURSOR := $2;
begin
    open resultset for
        select *
        from Reservations
        where resID = searchResID;
    return resultset;
end;
$$
LANGUAGE plpgsql;
```

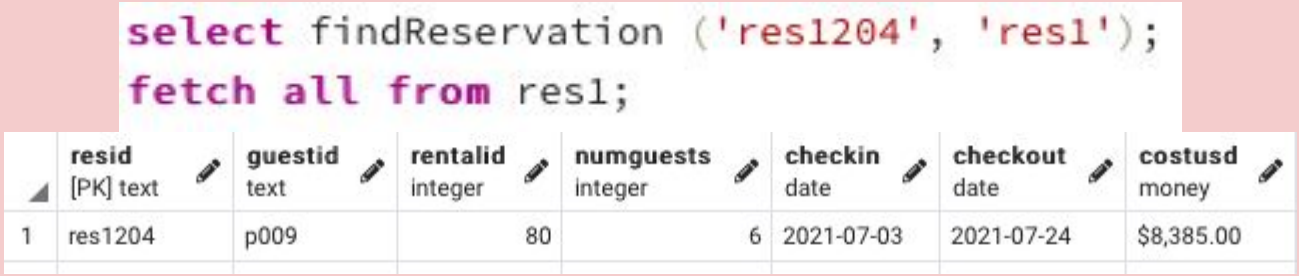
This function allows employees to find any reservation by simply inputting the reservation ID.

Testing: findReservation



```
select findReservation ('res1200', 'res');
fetch all from res;
```

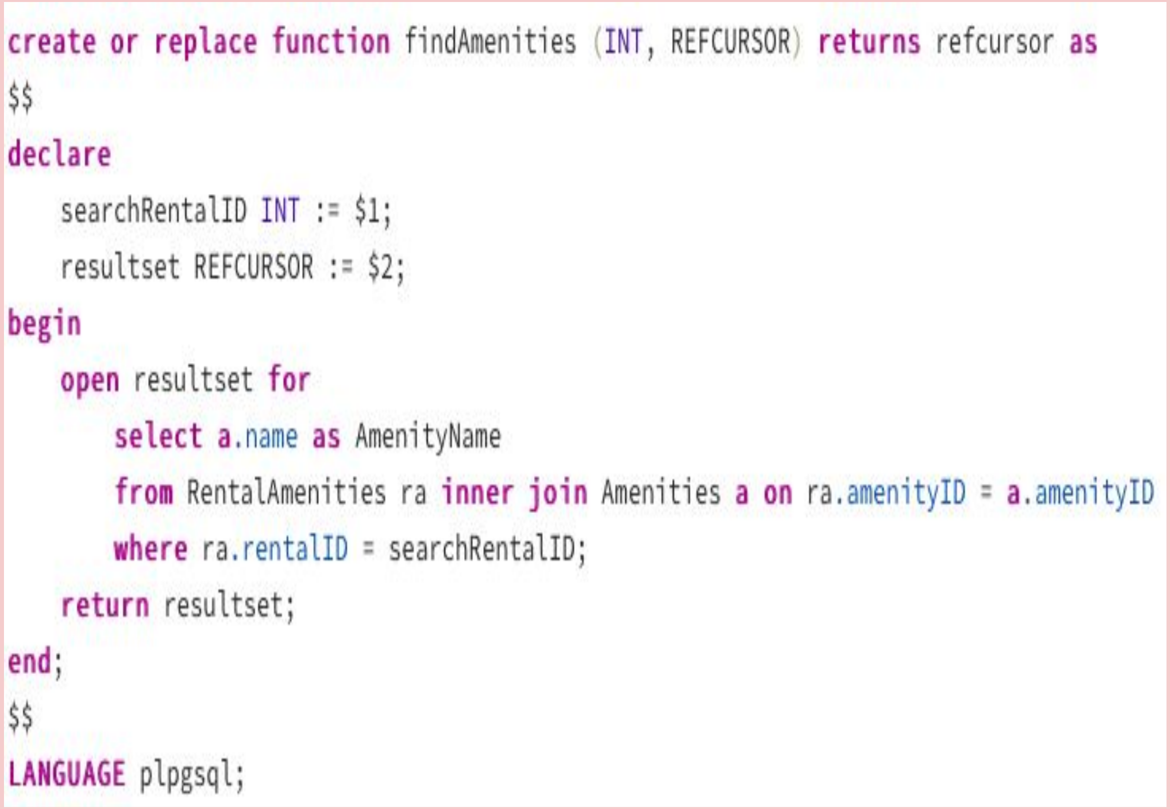
	resid [PK] text	guestid text	rentalid integer	numguests integer	checkin date	checkout date	costusd money
1	res1200	p002	80	3	2021-06-12	2021-06-26	$4,890.00



```
select findReservation ('res1204', 'res1');
fetch all from res1;
```

	resid [PK] text	guestid text	rentalid integer	numguests integer	checkin date	checkout date	costusd money
1	res1204	p009	80	6	2021-07-03	2021-07-24	$8,385.00

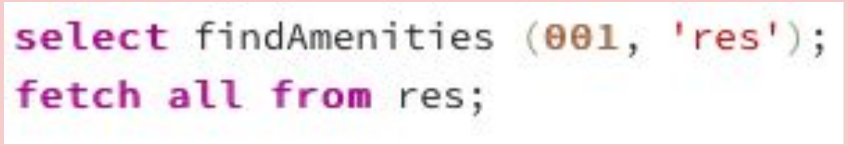
Stored Procedures: findAmenities



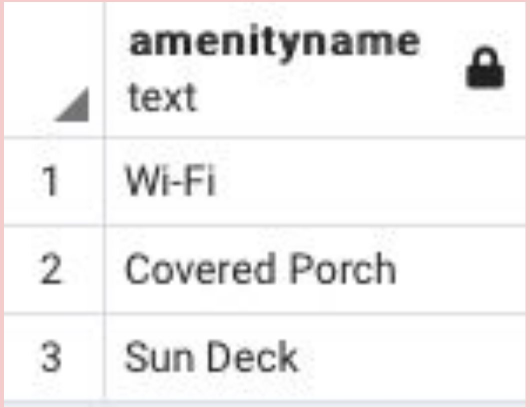
```
create or replace function findAmenities (INT, REFCURSOR) returns refcursor as
$$
declare
    searchRentalID INT := $1;
    resultset REFCURSOR := $2;
begin
    open resultset for
        select a.name as AmenityName
        from RentalAmenities ra inner join Amenities a on ra.amenityID = a.amenityID
        where ra.rentalID = searchRentalID;
    return resultset;
end;
$$
LANGUAGE plpgsql;
```

This function allows employees to quickly find all the amenities available in each rental by inputting the rental ID.

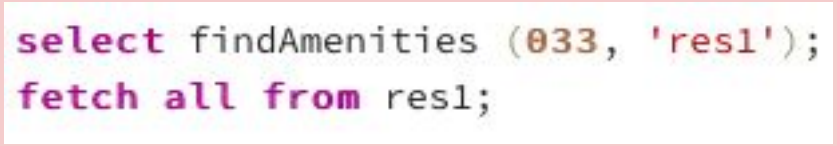
Testing: findAmenities



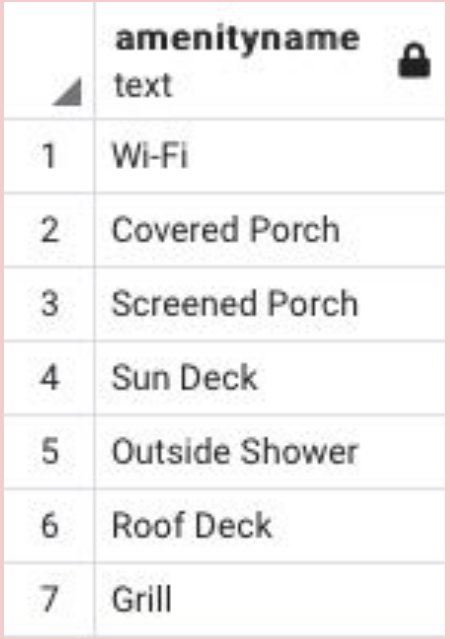
```
select findAmenities (001, 'res');
fetch all from res;
```



	amenityname text
1	Wi-Fi
2	Covered Porch
3	Sun Deck

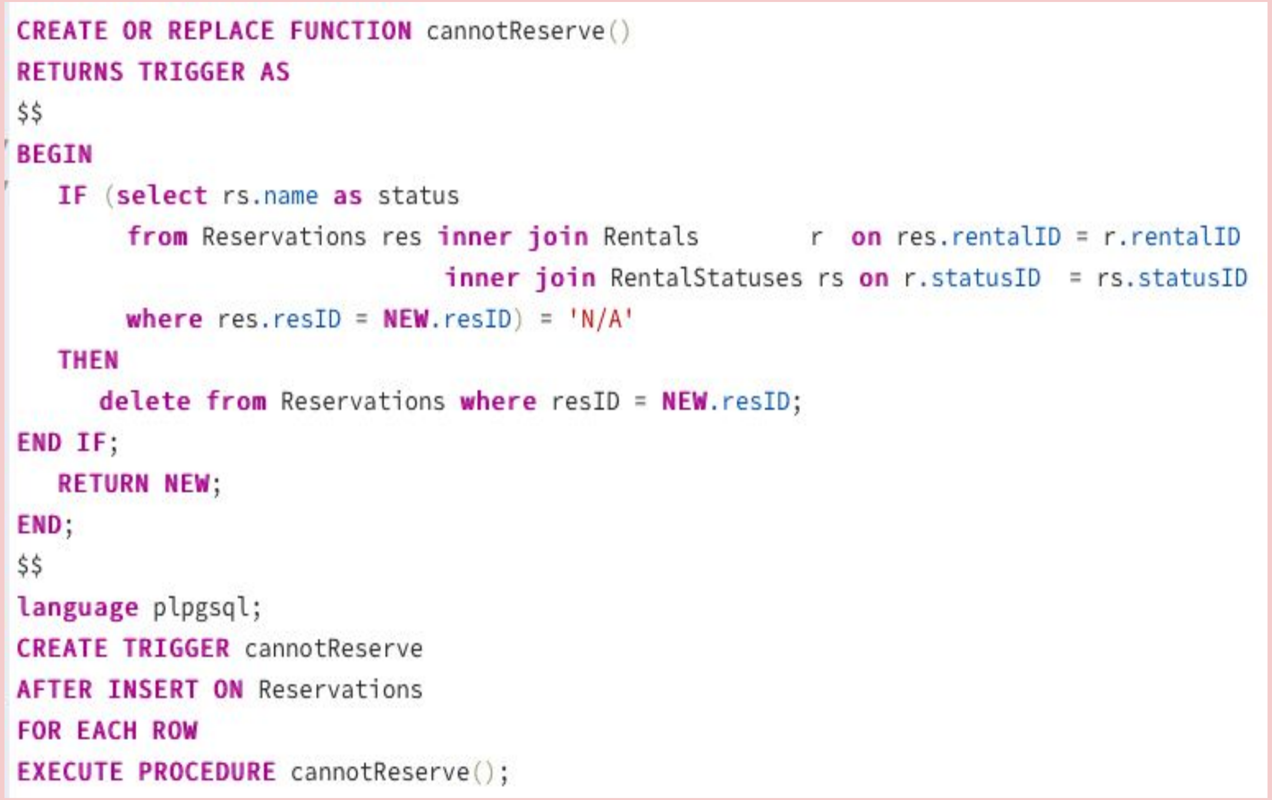


```
select findAmenities (033, 'res1');
fetch all from res1;
```



	amenityname text
1	Wi-Fi
2	Covered Porch
3	Screened Porch
4	Sun Deck
5	Outside Shower
6	Roof Deck
7	Grill

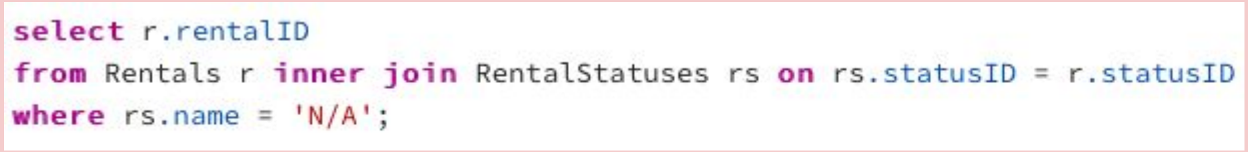
Triggers: cannotReserve



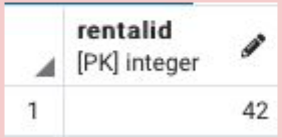
```
CREATE OR REPLACE FUNCTION cannotReserve()
RETURNS TRIGGER AS
$$
BEGIN
   IF (select rs.name as status
        from Reservations res inner join Rentals        r  on res.rentalID = r.rentalID
                              inner join RentalStatuses rs on r.statusID  = rs.statusID
        where res.resID = NEW.resID) = 'N/A'
   THEN
      delete from Reservations where resID = NEW.resID;
END IF;
   RETURN NEW;
END;
$$
language plpgsql;
CREATE TRIGGER cannotReserve
AFTER INSERT ON Reservations
FOR EACH ROW
EXECUTE PROCEDURE cannotReserve();
```

This triggers ensures that a rental with the status 'N/A' cannot be reserved.

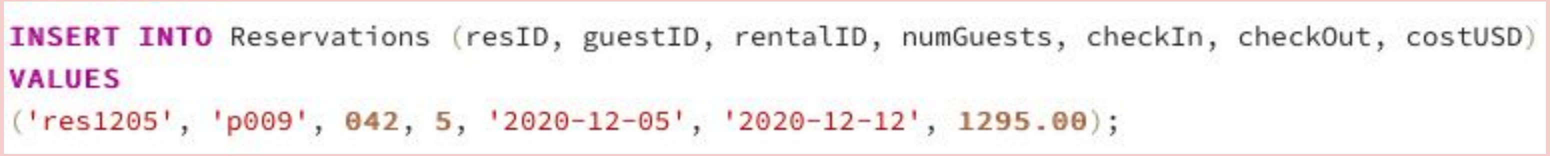
Testing: cannotReserve



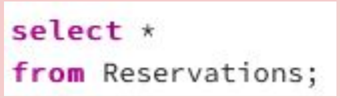
```
select r.rentalID
from Rentals r inner join RentalStatuses rs on rs.statusID = r.statusID
where rs.name = 'N/A';
```



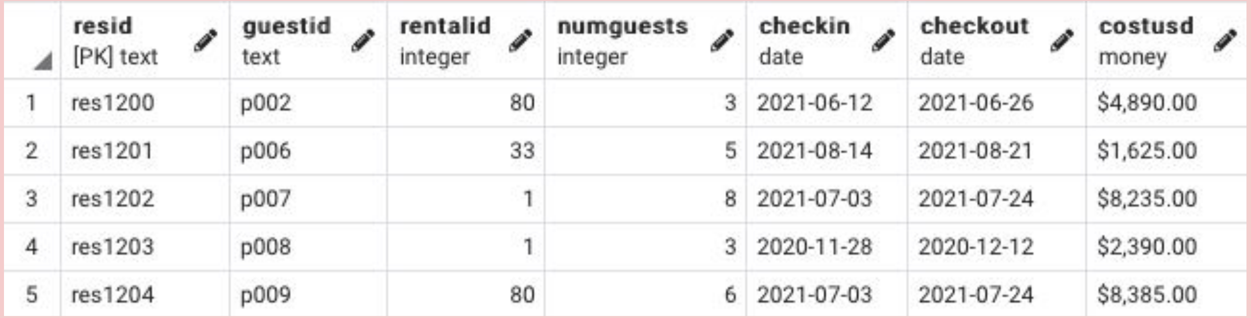
	rentalid [PK] integer
1	42



```
INSERT INTO Reservations (resID, guestID, rentalID, numGuests, checkIn, checkOut, costUSD)
VALUES
('res1205', 'p009', 042, 5, '2020-12-05', '2020-12-12', 1295.00);
```

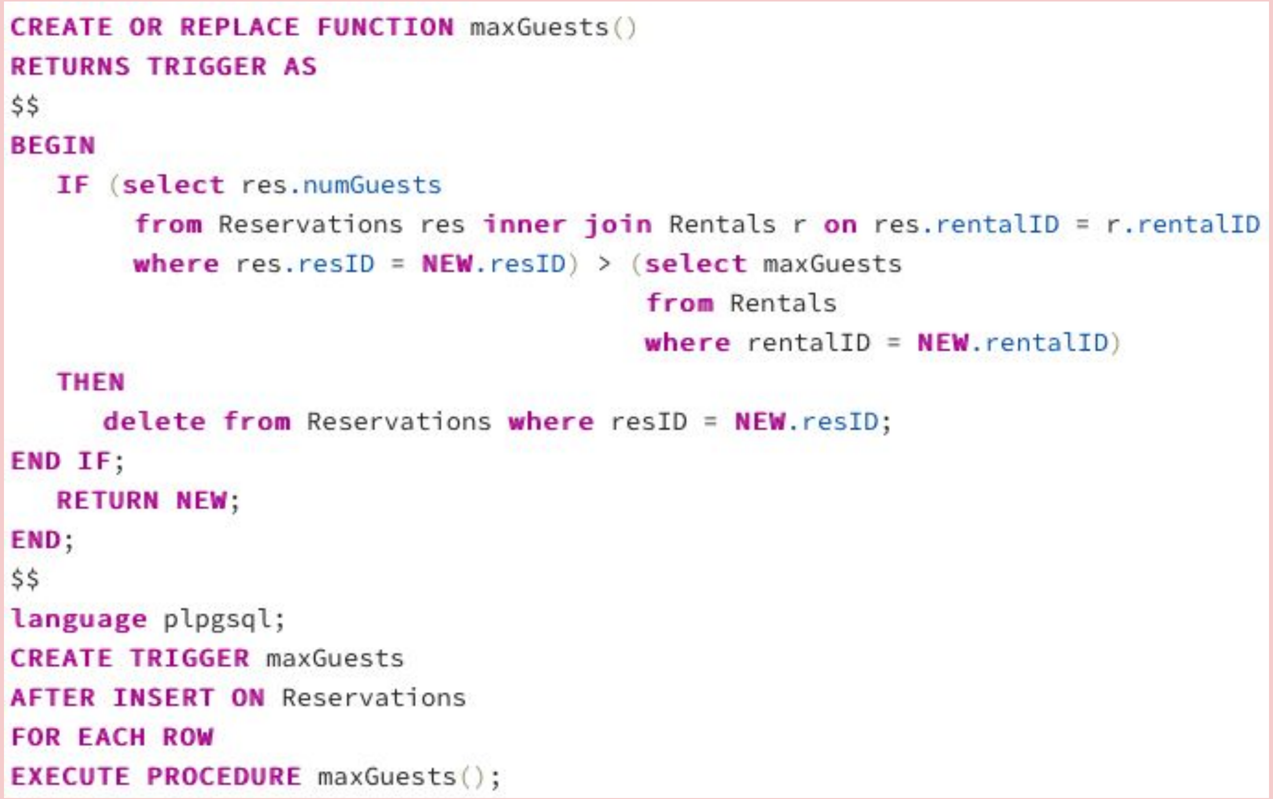


```
select *
from Reservations;
```



	resid [PK] text	guestid text	rentalid integer	numguests integer	checkin date	checkout date	costusd money
1	res1200	p002	80	3	2021-06-12	2021-06-26	$4,890.00
2	res1201	p006	33	5	2021-08-14	2021-08-21	$1,625.00
3	res1202	p007	1	8	2021-07-03	2021-07-24	$8,235.00
4	res1203	p008	1	3	2020-11-28	2020-12-12	$2,390.00
5	res1204	p009	80	6	2021-07-03	2021-07-24	$8,385.00

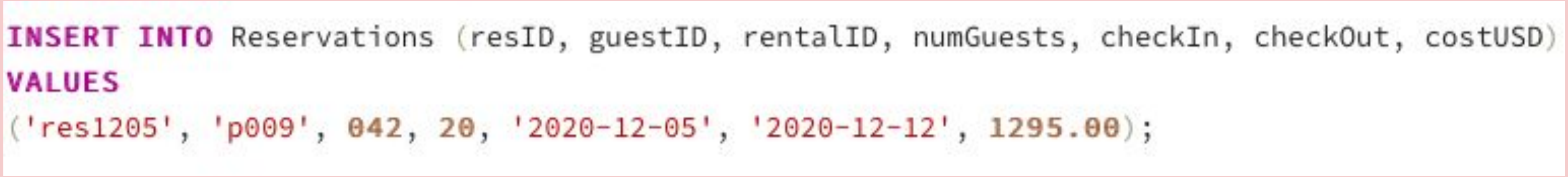
Triggers: maxGuests



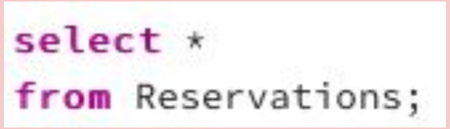
```
CREATE OR REPLACE FUNCTION maxGuests()
RETURNS TRIGGER AS
$$
BEGIN
   IF (select res.numGuests
        from Reservations res inner join Rentals r on res.rentalID = r.rentalID
        where res.resID = NEW.resID) > (select maxGuests
                                         from Rentals
                                         where rentalID = NEW.rentalID)
   THEN
      delete from Reservations where resID = NEW.resID;
END IF;
   RETURN NEW;
END;
$$
language plpgsql;
CREATE TRIGGER maxGuests
AFTER INSERT ON Reservations
FOR EACH ROW
EXECUTE PROCEDURE maxGuests();
```

This trigger prevents a reservation with the number of guests exceeding the capacity of the rental from being made.

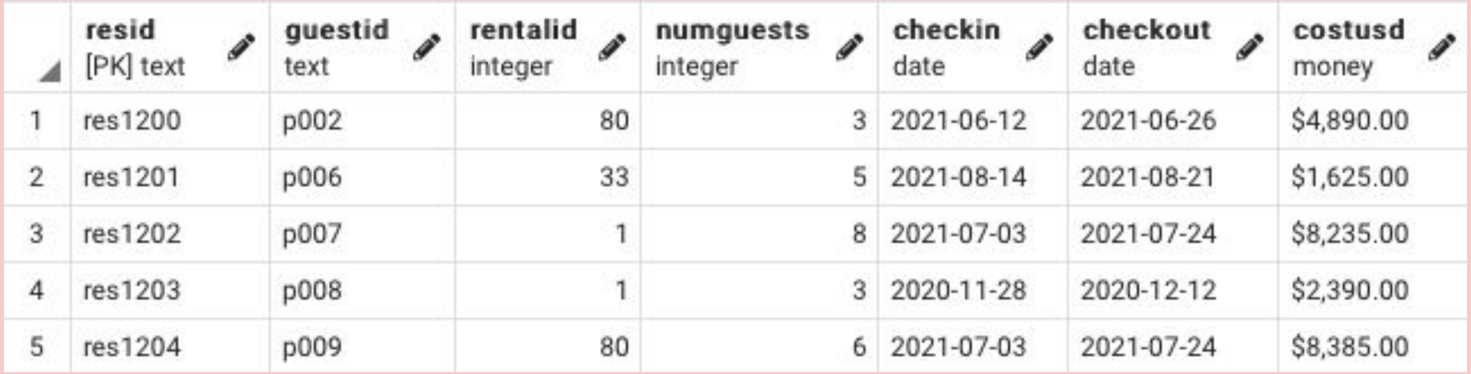
Testing: maxGuests



```sql
INSERT INTO Reservations (resID, guestID, rentalID, numGuests, checkIn, checkOut, costUSD)
VALUES
('res1205', 'p009', 042, 20, '2020-12-05', '2020-12-12', 1295.00);
```

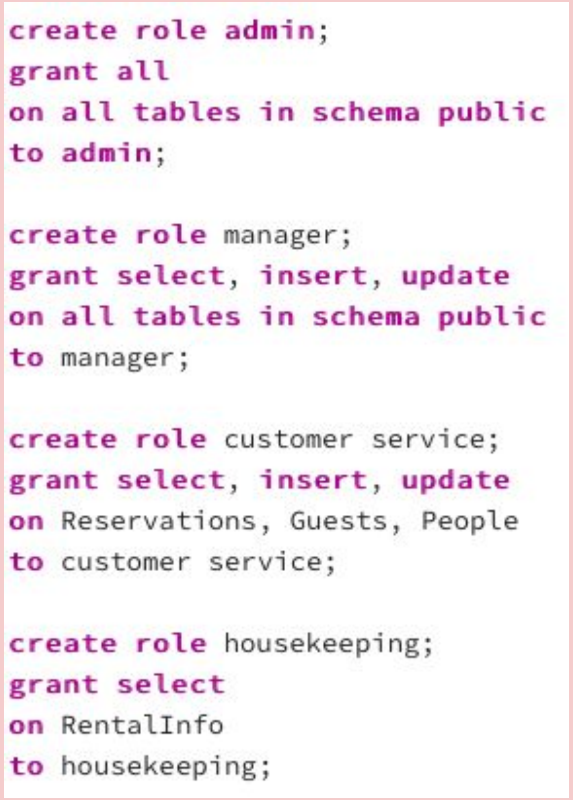


```sql
select *
from Reservations;
```



	resid [PK] text	guestid text	rentalid integer	numguests integer	checkin date	checkout date	costusd money
1	res1200	p002	80	3	2021-06-12	2021-06-26	$4,890.00
2	res1201	p006	33	5	2021-08-14	2021-08-21	$1,625.00
3	res1202	p007	1	8	2021-07-03	2021-07-24	$8,235.00
4	res1203	p008	1	3	2020-11-28	2020-12-12	$2,390.00
5	res1204	p009	80	6	2021-07-03	2021-07-24	$8,385.00

Security



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

create role manager;
grant select, insert, update
on all tables in schema public
to manager;

create role customer service;
grant select, insert, update
on Reservations, Guests, People
to customer service;

create role housekeeping;
grant select
on RentalInfo
to housekeeping;
```

Admin: This role grants access to all aspects of the database, likely the owner of Sunset Vacations.

Manager: The manager role grants similar access to that of the admin role, however they cannot delete anything within the database.

Customer Service: This allows customer service employees to make reservations and update any necessary personal information.

Housekeeping: This role is given access to a view that identifies all important rental information, which is necessary to allow housekeeping to do their job

Known Problems/Future Enhancements

- ❑ For the purposes of this project, I limited the sample data to only what was necessary. Much more data would be required to fully utilize all aspects of this database.
- ❑ After creating the Seasons table, I realized I probably should have used better names since some of them repeated, even though it doesn't make a significant difference since they have unique ID's. I tried to mirror the actual reservation process offered by Sunset Vacations, so I just used the information they provided (including season names).
- ❑ Reservation costs are simplified to only include the sum of the weekly rates. In the future, I would account for additional costs such as taxes, damage insurance, fees, etc. I was not really sure of the best way to implement this, and I felt it added a layer of complexity that was not necessary at this time.
- ❑ Similarly, I feel it would also be necessary to allow the option for nightly rates instead of only weekly rates. Sunset Vacations doesn't offer nightly rates on all of their properties, so I decided to just omit that for the purposes of this project.
- ❑ More views and stored procedures would also likely be necessary in order to provide a more simplified user experience.

<u>Final Thoughts:</u>

Sunset Beach has always been my favorite vacation spot, so I really enjoyed the process of designing this database, especially since my family and I visit almost every summer.
It was definitely interesting to learn more about Sunset Vacations and how their business operates, being that my family has used their services for nearly 20 years now.