Port Codd Palace

Database Management Final Project

By: Kara Lusabia
Instructor: Prof. Alan Labouseur
Due date: April 25, 2014

Table of Contents

Executive Summary

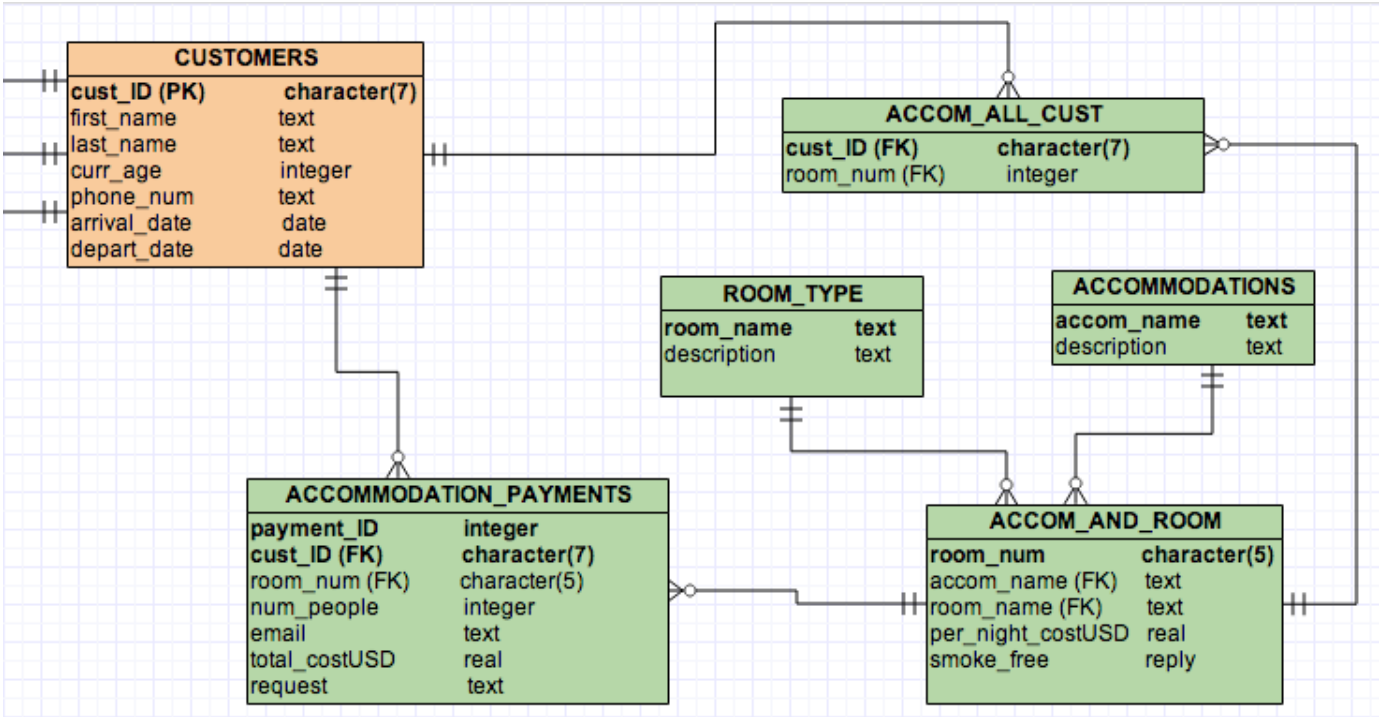
The purpose of this database is to provide a consistent set of information for the customers visiting the Port Codd Palace. Through the implementation of this database, customers are able to do restaurant and accommodation reservations. The resort also offers water activities where the customers can also make reservations to avoid schedule conflicts with the other customers.

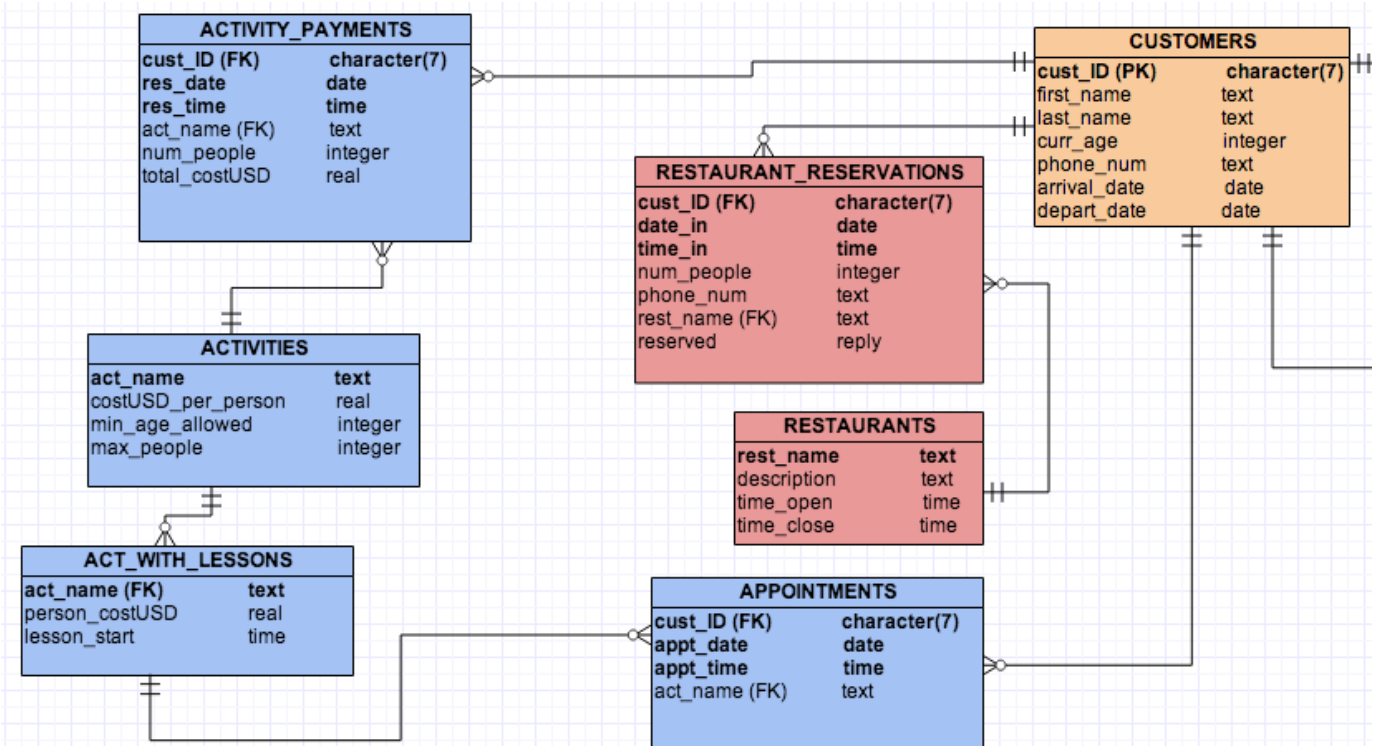
The following tables provide convenience to the customers and the employees as well, allowing the employees to keep track of the customers' bookings. Through these tables employees can keep track of:

- which rooms are currently available and will be available
- how many customers will be arriving and leaving given a certain date
- how many people are living in each accommodation
- how many children are currently in the resort
- which restaurants are currently open
- which of the water activities were popular to the customers

This document will provide the queries and an ER diagram to physically represent the database. There will also be functional dependencies, views, stored procedures and security. All of these are factors in flourishing this resort, because these queries will provide convenience for the employees and especially the customers.

Entity-Relationship Diagram




ACTIVITY_PAYMENTS
cust_ID (FK) character(7)
res_date date
res_time time
act_name (FK) text
num_people integer
total_costUSD real
CUSTOMERS
cust_ID (PK) character(7)
first_name text
last_name text
curr_age integer
phone_num text
arrival_date date
depart_date date
RESTAURANT_RESERVATIONS
cust_ID (FK) character(7)
date_in date
time_in time
num_people integer
phone_num text
rest_name (FK) text
reserved reply
ACTIVITIES
act_name text
costUSD_per_person real
min_age_allowed integer
max_people integer
RESTAURANTS
rest_name text
description text
time_open time
time_close time
ACT_WITH_LESSONS
act_name (FK) text
person_costUSD real
lesson_start time
APPOINTMENTS
cust_ID (FK) character(7)
appt_date date
appt_time time
act_name (FK) text

Tables

Overview, functional dependencies, create statements and sample data

Customers Table

Overview: It contains all the customers that are booked in the resort, whether they will be staying overnight or not. Basic information are also included in this table.

Functional Dependency: cust_ID -> first_name, last_name, curr_age, phone_num, arrival_date, depart_date

Create statements:

```
DROP TABLE IF EXISTS customers;
CREATE TABLE customers (
   cust_ID       char(7) not null,
   first_name    text,
   last_name     text,
   curr_age      integer,
   phone_num     text,
   arrival_date  date,
   depart_date   date,
  primary key(cust_ID)
);
```

Sample data:

	cust_id character(7)	first_name text	last_name text	curr_age integer	phone_num text	arrival_date date	depart_date date
1	AB20000	Alexis	Davis	24	917-083-6882	2012-03-23	2012-03-25
2	AB20001	Alley	Davis	22	917-759-2551	2012-03-23	2012-03-25
3	AB20002	John	Smith	36	845-892-1398	2014-03-23	2014-03-28

Accommodations Table

Overview: The resort includes four hotels where the customers can stay according to their needs. Each accommodation is distinguished through their location.

Functional Dependency: accom_name -> description

Create statements:

```
DROP TABLE IF EXISTS accomodations;
CREATE TABLE accommodations (
   accom_name    text not null,
   description   text,
  primary key(accom_name)
);
```

Sample data:

accom_name text	description text
Royal Escapade	Great view of the beach
Grand Villa	Best place to stay in with the kids
Royal Park	Close distance to restaurant and bars
Bel-air	Located at the centermost of the resort

Room_type Table

Overview: This table denotes the room types that the customers will be staying in. The room types differ depending on their size and what each room type encompasses. Each of them is also available in each of the accommodations in the hotel.

Functional Dependency: room_name -> description

Create statements:

```
DROP TABLE IF EXISTS room_type;
CREATE TABLE room_type (
   room_name     text not null,
   description   text,
  primary key(room_name)
);
```

Sample data:

room_name text	description text
Double Standard	Queen size bed with one bathroom
Upper Standard	2 queen size beds with a kitchen and a bathroom
Family Suite	2 bedrooms each having a queen sized bed. Also has a common kitchen and 2 bathrooms
Executive Suite	2 bedrooms each having a queen sized bed. Also has a common kitchen, living room and 2 bathrooms

Accom_and_room Table

Overview: Each room in each accommodation has a different room type. To identify the characteristics of each, the room number is used to uniquely identify these types of characteristics, including the cost of that room per night. The table also states whether the room is smoke free or not, in order to please the customers.

Functional Dependency: room_num -> accom_name, room_name, per_night_costUSD, smoke_free

Create statements:

```
DROP TABLE IF EXISTS accom_and_room;
CREATE TYPE reply AS ENUM ('yes', 'no');
CREATE TABLE accom_and_room (
  room_num              char(5) not null,
  accom_name            text references accommodations(accom_name),
  room_name             text references room_type(room_name),
  per_night_costUSD     real,
  smoke_free            reply,
 primary key(room_num)
);
```

Sample data:

	room_num character(5)	accom_name text	room_name text	per_night_costusd real	smoke_free reply
8	RE204	Royal Escapade	Executive Suite	150	no
9	GV101	Grand Villa	Upper Standard	110	yes
10	GV102	Grand Villa	Family Suite	125	yes
11	GV103	Grand Villa	Family Suite	125	yes
12	GV201	Grand Villa	Executive Suite	150	yes
13	GV202	Grand Villa	Executive Suite	150	yes
14	RP101	Royal Park	Double Standard	99	yes

Accommodation_payments Table

Overview: When doing reservations, one person could pay for the accommodation or the group could split up the payment. This table denotes the customers that paid for their room. Since multiple customers could pay for the same room, a composite primary key was used mainly to solve this occurrence. The total cost in this table determines the total accommodation cost, which is based on how long the customers are staying. If more than one person is paying for the room, it is assumed that those people evenly distribute the total amount of payment. The request column allows the customers to provide additional request they want to include in their room.

Functional Dependency: payment_ID, cust_ID -> room_num, num_people, email, total_costUSD, request

Create statements:

```
DROP TABLE IF EXISTS accommodation_payments;
CREATE TABLE accommodation_payments (
  payment_ID    integer not null,
  cust_ID       char(7) not null references customers(cust_ID),
  room_num      char(5) references accom_and_room(room_num),
  num_people    integer,
  email         text,
  total_costUSD real,
  request       text,
  primary key(payment_ID, cust_ID)
);
```

Sample data:

	payment_id integer	cust_id character(7)	room_num character(5)	num_people integer	email text	total_costusd real	request text
1	1	AB20002	GV102	4	jsmith163@gmail.com	540.62	none
2	2	AB20000	RE104	1	alexisdavis112@yahoo.com	54.47	none
3	2	AB20001	RE104	1	alleydavis213@yahoo.com	54.47	none
4	3	AB20006	RP103	2	andersond115@gmail.com	237.87	3rd Anniversary Package
5	4	AB20008	BA204	1	stonethomas@verizon.net	173	none
6	4	AB20010	BA204	2	thesamnorris@gmail.com	346	none

Accom_all_cust Table

Overview: In most cases, there is more than one person staying in one room. This table identifies the room each customer is staying in.

Functional Dependency: cust_ID -> room_num

Create statements:

```
DROP TABLE IF EXISTS accom_all_cust;
CREATE TABLE accom_all_cust (
  cust_ID       char(7) not null references customers(cust_ID),
  room_num      char(5) references accom_and_room(room_num),
  primary key(cust_ID)
);
```

Sample data:

	cust_id character(7)	room_num character(5)
4	AB20003	RE104
5	AB20004	RE104
6	AB20005	RE104
7	AB20006	RP103
8	AB20007	RP103
9	AB20008	BA204
10	AB20009	BA204

Restaurants Table

Overview: Port Codd Palace also incorporates restaurants within the resort, where each of them serves a variety of food. All the restaurants are enlisted in this table including their hours and more information on the food they serve.

Functional Dependency: rest_name -> description, time_open, time_close

Create statements:

```
CREATE TABLE restaurants (
  rest_name     text not null,
  description   text,
  time_open     time,
  time_close    time,
 primary key(rest_name)
);
```

Sample data:

	rest_name text	description text	time_open time without time zone	time_close time without time zone
1	7th Note Cafe	Coffee and snack area	07:30:00	23:00:00
2	Sea Breeze Restaurant	Mainly seafood and Carribean dishes	11:00:00	23:00:00
3	Garden & Grill	BBQ, smoked and grilled dishes	10:30:00	21:00:00
4	Yellow Cab	American and Italian dishes	07:30:00	22:00:00
5	Cowboy Grill	BBQ, smoked and grilled dishes	11:00:00	21:30:00
6	1 Mile Diner	Diner with all possible dishes	06:00:00	23:00:00

Restaurant_reservations Table

Overview: Since there is a possibility that there are not enough tables to serve the customers, it is highly recommended to get a reservation in these restaurants. This table will store the reservations made by the customers, where each reservation is named under one customer. The purpose of having three primary keys in this table is to allow the customers to make more than one reservation in a day.

Functional Dependency: cust_ID, date_in, time_in -> num_people, phone_num, rest_name, reserved

Create statements:

```
CREATE TABLE restaurant_reservations (
   cust_ID       char(7) not null references customers(cust_ID),
   date_in       date,
   time_in       time,
   num_people    integer,
   phone_num     text,
   rest_name     text references restaurants(rest_name),
   reserved      reply,
  primary key(cust_ID, date_in, time_in)
);
```

Sample data:

	cust_id character(7)	num_people integer	phone_num text	rest_name text	date_in date	time_in time without time zone	reserved reply
5	AB20003	4	845-231-6393	Cowboy Grill	2014-03-23	19:20:00	yes
6	AB20003	3	845-231-6393	1 Mile Diner	2014-03-25	11:10:00	no
7	AB20002	4	845-892-1398	Garden & Grill	2014-03-25	20:15:00	yes
8	AB20002	4	845-892-1398	Sea Breeze Restaurant	2014-03-27	14:30:00	yes
9	AB20002	4	845-892-1398	7th Note Cafe	2014-03-28	09:30:00	yes
10	AB20014	4	722-117-7181	Yellow Cab	2013-07-04	14:30:00	yes

Activities Table

Overview: This resort also includes fun water activities for all ages. This table provides the list of all water activities the resort offers, including specific information for each activity. Knowing the minimum age for each activity increases the resort's reputation, because the lower the minimum age is, the more people are attracted to stay in the resort. Moreover, knowing the maximum number of people for each ride prevents appointment conflicts with the other customers.

Functional Dependency: act_name -> costUSD_per_person, min_age_allowed, max_people

Create statements:

```
CREATE TABLE activities (
   act_name              text,
   costUSD_per_person    real,
   min_age_allowed       integer,
   max_people            integer,
  primary key(act_name)
);
```

Sample data:

	act_name text	costusd_per_person real	min_age_allowed integer	max_people integer
1	Snorkeling	45	13	6
2	Parasailing	35	8	2
3	Wakeboarding	15	12	4
4	Banana Boat	15	7	8
5	Windsurfing	25	12	3
6	Flying Fish	35	7	4
7	Water Hopper	35	6	6
8	Water Skiing	20	7	8

Activity_payments Table

Overview: This table helps the employees keep track of which customers paid to do an activity. It is also possible for the same customer to pay for more than one activity, by having the reservation date and time as the primary keys along with the customer ID.

Functional Dependency: cust_ID, res_date, res_time -> act_name, num_people, total_costUSD

Create statements:

```
CREATE TABLE activity_payments (
  cust_ID       char(7) not null references customers(cust_ID),
  res_date      date,
  res_time      time,
  act_name      text references activities(act_name),
  num_people    integer,
  total_costUSD real,
 primary key(cust_ID, res_date, res_time)
);
```

Sample data:

	cust_id character(7)	act_name text	num_people integer	res_date date	res_time time without time zone	total_costusd real
5	AB20002	Water Skiing	2	2014-03-27	17:15:00	40
6	AB20006	Water Hopper	2	2013-07-04	17:30:00	70
7	AB20007	Snorkeling	2	2013-07-05	14:30:00	45
8	AB20013	Water Hopper	1	2013-07-04	17:30:00	35
9	AB20014	Water Hopper	1	2013-07-04	17:30:00	35
10	AB20008	Windsurfing	1	2014-04-13	09:30:00	25
11	AB20009	Windsurfing	1	2014-04-13	09:30:00	25
12	AB20010	Windsurfing	1	2014-04-13	09:30:00	25
13	AB20011	Banana Boat	1	2014-06-04	17:30:00	15

Act_with_lessons Table

Overview: Some of the activities involve taking lessons from a professional. This table lists all the activities where lessons are available for the customers. Having the information of when each lesson starts benefits the customers, so they have a rough estimate on what their itinerary should look like throughout their stay in the resort.

Functional Dependency: act_name -> person_costUSD, lesson_start

Create statements:

```
CREATE TABLE act_with_lessons (
  act_name              text references activities(act_name),
  person_costUSD        real,
  lesson_start          time,
 primary key(act_name)
);
```

Sample data:

	act_name text	person_costusd real	lesson_start time without time zone
1	Snorkeling	75	10:00:00
2	Wakeboarding	40	09:30:00
3	Windsurfing	55	13:30:00

Appointments Table

Overview: The purpose of having lessons for some of the activities is to learn the proper way of doing them and for safety purposes as well. This table lists the customers that took lessons for a certain activity. Likewise, this table has composite primary keys so that customers can take multiple lessons in a day.

Functional Dependency: cust_ID, appt_date, appt_time -> act_name

Create statements:

```
CREATE TABLE appointments (
  cust_ID       char(7) not null references customers(cust_ID),
  appt_date     date,
  appt_time     time,
  act_name      text references activities (act_name),
 primary key(cust_ID, appt_date, appt_time)
);
```

Sample data:

	cust_id character(7)	appt_date date	appt_time time without time zone	act_name text
1	AB20008	2014-04-14	15:00:00	Wakeboarding
2	AB20009	2014-04-14	15:00:00	Wakeboarding
3	AB20010	2014-04-14	15:00:00	Wakeboarding
4	AB20002	2014-03-24	13:30:00	Windsurfing
5	AB20003	2014-03-24	13:30:00	Windsurfing

Views

Accommodation_customers_2014 View

- **This view creates a list of customers who accommodated in the resort for the year of 2014. Depending on the result of this statistic, the management could modify some of its accommodations to attract more customers.**

Create Statements:

```
create view accommodation_customers_2014 as
select c.last_name, c.first_name, c.curr_age, c.arrival_date, a.room_num
from customers as c, accom_all_cust as a, accom_and_room as ac
where c.cust_ID = a.cust_ID
and a.room_num = ac.room_num
and c.arrival_date::text LIKE '2014%'
order by c.last_name

select * from accommodation_customers_2014
```

Output:

	last_name text	first_name text	curr_age integer	arrival_date date	room_num character(5)
1	Norris	Sam	24	2014-04-12	BA204
2	Norris	Alvin	24	2014-04-12	BA204
3	Smith	Cassandra	12	2014-03-23	RE104
4	Smith	John	36	2014-03-23	RE104
5	Smith	Logan	14	2014-03-23	RE104
6	Smith	Leslie	35	2014-03-23	RE104
7	Stone	Thomas	27	2014-04-12	BA204

Views

Popular_restaurants_2014 View

- **This view gives the total number of people that ate in the restaurants during the year of 2014. Its purpose is to improve the restaurants with the least number of customers and maintain the popularity of the restaurant with the most number of customers.**

Create Statements:

```
create view popular_restaurants_2014 as
select rr.rest_name, sum(rr.num_people)
from restaurants as r, restaurant_reservations as rr
where rr.date_in::text LIKE '2014%'
and r.rest_name = rr.rest_name
group by rr.rest_name
order by sum asc;

select * from popular_restaurants_2014
```

Output:

	rest_name text	sum bigint
1	Yellow Cab	3
2	7th Note Cafe	4
3	1 Mile Diner	5
4	Garden & Grill	6
5	Cowboy Grill	7
6	Sea Breeze Restaurant	9

Views

Cust_most_expenses View

- **This lists the customers that paid for accommodation and activities starting from the person that spent the most amount of money. Knowing how much money the customers spent in resort services provides statistics for the management, so they would have an idea whether the prices of the amenities should be readjusted or not.**

Create Statements:

```
create view cust_most_expenses as
select c.last_name, c.first_name, ap.cust_ID, sum(acp.total_costUSD + ap.total_costUSD)
from customers as c, activity_payments as acp, accommodation_payments as ap
where c.cust_ID = ap.cust_ID
and c.cust_ID = acp.cust_ID
group by ap.cust_ID, acp.cust_ID,  c.last_name, c.first_name
order by sum desc;

select * from cust_most_expenses
```

Output:

	last_name text	first_name text	cust_id character(7)	sum real
1	Smith	John	AB20002	1181.24
2	Norris	Sam	AB20010	371
3	Anderson	Derek	AB20006	307.87
4	Stone	Thomas	AB20008	198
5	Davis	Alley	AB20001	124.47
6	Davis	Alexis	AB20000	124.47

Queries

- **Gives the customers that paid for the room they are staying in, the room number and their location. The purpose of this is if there was a malfunction in the hotel's payment system, employees can immediately contact the people that paid for the room.**

Create Statements:

```
select c.last_name, c.first_name, c.cust_ID, ac.room_num
from customers as c, accommodation_payments as ap, accom_all_cust as ac,
     accommodations as a, accom_and_room as ar
where c.cust_ID = ap.cust_ID
and c.cust_ID = ac.cust_ID
and ar.room_num = ap.room_num
and ar.accom_name = a.accom_name
order by c.last_name asc;
```

Output:

	last_name text	first_name text	cust_id character(7)	room_num character(5)
1	Anderson	Derek	AB20006	RP103
2	Davis	Alexis	AB20000	GV102
3	Davis	Alley	AB20001	GV102
4	Norris	Sam	AB20010	BA204
5	Smith	John	AB20002	RE104
6	Stone	Thomas	AB20008	BA204

Queries

- **This query produces the number of people that lives in the occupied rooms. In case of an emergency, it would be a safe procedure if the employees have a rough estimate on how many customers are staying in the resort.**

Create Statements:

```
select a.accom_name, ap.room_num, sum(ap.num_people)
from room_type as r, accommodations as a, customers as c, accommodation_payments as ap,
     accom_and_room as ar
where c.cust_ID = ap.cust_ID
and ap.room_num = ar.room_num
and r.room_name = ar.room_name
and a.accom_name = ar.accom_name
group by a.accom_name, ap.room_num
order by a.accom_name asc;
```

Output:

	accom_name text	room_num character(5)	sum bigint
1	Bel-air	BA204	3
2	Grand Villa	GV102	4
3	Royal Escapade	RE104	2
4	Royal Park	RP103	2

Queries

- **Customers can make certain requests in the accommodations they are staying in. This query will give the customers who made a request, specifically those that requested a certain package deal, and the room they are in.**

Create Statements:

```
select c.last_name, c.first_name, a.room_num
from customers as c right outer join accommodation_payments as a
on c.cust_ID = a.cust_ID
where a.request::text LIKE '%Package'
order by c.last_name asc;
```

Sample Data:

	last_name text	first_name text	room_num character(5)
1	Anderson	Derek	RP103

Stored Procedures

- **The function returns the total number of customers that have done the activity or has made an appointment, which is the passed-in parameter in the function. Through this function, the employees can identify which activities are the most and least popular.**

Create Statements:

```
create or replace function getTotalActivityCust(text, refcursor) returns refcursor as
$$
declare
   act_name text  := $1;
   resultset refcursor  := $2;
begin
open resultset for
        select sum(ap.num_people) as sum_of_customers
        from activity_payments as ap
        where ap.act_name = $1;
        return resultset;
end;
$$ language plpgsql;

select getTotalActivityCust('Banana Boat', 'results');
fetch all from results;
```

Output:

	sum_of_customers bigint
1	6

Stored Procedures

- **This function will constantly be needed in order to improve the quality of the restaurants in the resort. Knowing the total number of customers for each restaurant helps the management keep track of which restaurants to focus on for improvement.**

Create Statements:

```
create or replace function getTotalRestaurantCust(text, refcursor) returns refcursor as
$$
declare
   rest_name text  := $1;
   resultset refcursor  := $2;
begin
open resultset for
        select sum(rr.num_people) as sum_of_restaurant_cust
        from restaurant_reservations as rr
        where rr.rest_name = $1;
        return resultset;
end;
$$ language plpgsql;

select getTotalRestaurantCust('Garden & Grill', 'results');
fetch all from results;
```

Ouput:

	sum_of_restaurant_cust bigint
1	8

Security

- **Activity instructors can only manage the tables related to the activities. Other employees are not allowed to do any types of modification, because the appointments must be based on the schedule of the instructors, and the instructors are the ones that handle the customers' payments.**

```
create role instructor;
grant select, insert, delete, update
on activities, activity_payments, act_with_lessons, appointments
to instructor
```

- **Customers are given the privilege to select the amenities of the resort. This way, they are aware of the services the resort has to offer, and they can make reservations according to their needs.**

```
create role customers;
grant select
on customers, accommodations, room_type, accom_and_room, restaurants, activities
to customers
```

- **Database administrator can fully access the database, because his job is to maintain this database according to the needs of the resort.**

```
create role database_administrator;
grant select, insert, delete, update
on all tables in schema public
to database_administrator
```

Implementation Notes

- As of right now, the arrival date was made so it is before or the same day as the departure date. Similarly, restaurant reservations are made between the arrival and departure date of the customers. However, there can be moments where the employees could make a mistake on inserting these times and dates. Therefore, it would be beneficial to implement the checking of the times and dates, making sure that they are in sync and consistent.

- The total cost for the accommodations has been calculated based on the number of nights the customer has reserved the room. If the customer wants to extend his stay, the total cost must be calculated again. In other words, the total cost is not automatically calculated once the departure date is changed. When a customer decides to extend his stay, the employees have to make sure that the total cost gets updated in addition to the departure date.

Known Problems

- Currently, customers must make reservations in order to have accommodation and similar goes for the water activities. In order for the business to flourish, customers must be able to request for a room or be able to participate in water activities without making reservations. This can occur in cases where people are in the mood for a spontaneous adventure or in the mood for spending lavishly.

- The Restaurant_reservations table only lists the customers who made the reservations and the number of people in the group. The name of the customers that ate at the restaurants are not identified. Having this information can have its benefits. For instance, if someone had a medical emergency, such as a customer having an allergic reaction from the food he ate, the management can confirm whether this incident puts the resort at fault, depending whether his name was in this table or not.

- Under the appointments table, each customer that takes a lesson are listed individually, even if the customers have their own corresponding groups. Unlike the activity_payments table, it lists the customer that paid for that activity, along with the number of people the customer paid for. It would have been easier if the appointments table was implemented similar to the activity_payments table, so the number of customers doing the lessons can be calculated easily.

Future Enhancements

- To prevent any conflict, it would be helpful to combine the customers that took lessons for an activity and the customers that paid for an activity. From there, create a new table where it only contains the customers who took lessons.

- For the lessons as well, adding a column for the instructors and the activity they teach would cause less havoc when making appointments. This way, the customers are not making appointments when the instructors are not available.

- It would also help if the employees are listed and the accommodation reservations they have made for the customers. Besides avoiding more stress to the employees and customers, this provides an efficient way to achieve great customer service, because the employees would have an idea on how to assist the customers they helped in the first place.