Hotel Database Project

Team 11
Courtney Burns, Mark Makris, Nick Franklin, Sam Bennett, Sharon Soh

IS 420
Dr. Karabatis
4 Dec. 2018

Final Code

Member 1: Courtney Burns

1. **Add a new Hotel:**

```
create sequence newHotelSeq;

create or replace procedure addNewHotel (Hotel_Name in
hotel.HotelName%type, HotelContactFName in
hotel.HotelContactFirstName%type, HotelContactLName in
hotel.HotelContactLastName%type, Hotel_Address in hotel.Address%type,
Hotel_City in hotel.City%type, Hotel_State in hotel.State%type, Hotel_ZipCode in
hotel.ZipCode%type, Hotel_PhoneNumber in hotel.PhoneNumber%type)
Is
--declaring exception
stateFormatException exception;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 1: Courtney Burns. addNewHotel');
--trigger for exception
If length(Hotel_State) > 2 then
Raise stateFormatException;
End if;
--inserting the typed values into the hotel table
Insert into hotel (HotelID, HotelName, HotelContactFirstName,
HotelContactLastName, Address, City, State, ZipCode, PhoneNumber) values
(newHotelSeq.nextval, Hotel_Name, HotelContactFName, HotelContactLName,
Hotel_Address, Hotel_City, Hotel_State, Hotel_ZipCode, Hotel_PhoneNumber);
--what happens when the exception occurs
Exception
When stateFormatException then
dbms_output.put_line('Please enter the two letter abbreviation for your state.');
End;
```

2. **Find hotels using address, output hotel ID:**

```
create or replace procedure findHotels(address_in IN hotel.address%type, city_in in
    hotel.city%type, state_in in hotel.state%type, zipcode_in in hotel.zipcode%type)
    IS
hotel_id number;
cursor findHotelID IS
```

```
--In the original prompt, it was assumed that we would be looking for a whole
    address. In scenario, only partial is given. Therefore, we accounted for this using
    an explicit cursor.
--Since there will be multiple outputs when searching for just city and state, an
    explicit cursor must be used.
  Select HotelID
  from Hotel
  where hotel.city = city_in and hotel.state = state_in;
--Parse the cursor.
hotel_id_row findHotelID%rowtype;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 1: Courtney Burns. findHotels');

--Again checking for address or zipcode missing to make scenario run correctly
if address_in is null or zipcode_in is null
then
  --Run the cursor that may contain multiple hotels in one city
  for hotel_id_row in findHotelID
    loop
    dbms_output.put_line('A hotel at this address has the id: ' ||
   hotel_id_row.HotelID);
    --This line exists to put a space between each print. Though this procedure will
    only ever display one record in our case, if someone were to put in a hotel with
    the same ID, this would separate the records in the printed statements.
    dbms_output.put_line(' ');
    end loop;
else
  -- Full address, run as normal
  select HotelID into hotel_id
  from hotel
  where hotel.address = address_in and hotel.city = city_in and hotel.state =
    state_in and hotel.zipcode = zipcode_in;
  dbms_output.put_line('The hotel at this address has the id: ' || hotel_id);
end if;
--If exception is raised, the system prints error message.
EXCEPTION
  when No_Data_Found then
  dbms_output.put_line('No hotels at this address were found.');
End;
```

3. **Get hotel info, output all information about a hotel:**

```
create or replace procedure getHotelInfo(Hotel_ID IN integer) IS
```

```
cursor hotelInfo IS
--Since there will be multiple outputs, an explicit cursor must be used.
   Select HotelID, HotelName, HotelContactFirstName, HotelContactLastName,
    Address, City, State, ZipCode, PhoneNumber
   from Hotel
   where Hotel.HotelID = Hotel_ID;
--Parse the cursor.
hotel_info_row hotelInfo%rowtype;
--Setting a boolean to determine if any data exists for this particular query in order to
    determine whether an exception will occur or not.
hotelIDFound boolean := false;
noHotelIDFound exception;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 1: Courtney Burns. getHotelInfo');
--Looping through the hotelInfo cursor into hotel_info_row rowtype until no more
    records can be pulled, returning the name, address, contact information, and
    phone number of the hotel with the given ID.
for hotel_info_row IN hotelInfo
loop
   hotelIDFound := true;
   dbms_output.put_line('Hotel with ID ' || hotel_info_row.HotelID || ': ');
   dbms_output.put_line('Hotel Name: ' || hotel_info_row.HotelName);
   dbms_output.put_line('Hotel Contact: ' || hotel_info_row.HotelContactFirstName || '
    ' || hotel_info_row.HotelContactLastName);
   dbms_output.put_line('Address: ' || hotel_info_row.Address || ', ' ||
    hotel_info_row.City || ', ' || hotel_info_row.State || ', ' || hotel_info_row.ZipCode);
   dbms_output.put_line('Hotel Phone Number: ' || hotel_info_row.PhoneNumber);
--This line exists to put a space between each print. Though this procedure will only
    ever display one record in our case, if someone were to put in a hotel with the
    same ID, this would separate the records in the printed statements.
   dbms_output.put_line(' ');
end loop;
--Exception is raised if the state entered by the user does not have hotels.
if hotelIDFound = false then
   raise noHotelIDFound;
end if;
--If exception is raised, the system prints error message.
EXCEPTION
   when noHotelIDFound then
   dbms_output.put_line('No hotels with this ID were found.');
end;
```

4. **Add an Event Room:**

```
create sequence eventRoomSeq;

create or replace procedure addEventRoom (Hotel_ID in room.HotelID%type,
    Room_Type in room.RoomType%type) is
--initializing values for the capacity and price since they will not be editable for users
capacityOf number;
priceOf number;
room_name room.RoomType%type := Room_Type;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 1: Courtney Burns. addEventRoom');
--trims whitespace from words to allow for cleaner checking
room_name := TRIM(room_name);
--makes words entirely lowercase to make sure that the system can properly check
    for the room type
room_name := LOWER(room_name);
--This case statement allows capacity and prices to be automatically applied for our
    three room types. Since the room types must have specific numeric values for
    both capacity and price, this makes it impossible for users to input any values
    outside of those given parameters.
CASE
When room_name = 'small hall' then
    capacityOf := 100;
    priceOf := 500;
When room_name = 'medium hall' then
    capacityOf := 250;
    priceOf := 1000;
When room_name = 'large hall' then
    capacityOf := 500;
    priceOf := 2000;
End case;
--makes the first letter of each word uppercase
room_name := initcap(room_name);
--inserting the typed values into the room table
Insert into room (RoomID, HotelID, RoomType, Capacity, Price) values
    (eventRoomSeq.nextval, Hotel_ID, Room_Type, capacityOf, priceOf);
End;
```

5. **Get report of hotels in the state and all event rooms**

```
create or replace procedure getStateHotelEventRoomReport(Hotel_State IN
    varchar2) IS
```

```
--There will be multiple hotels and rooms, so multiple rows will be occurring which is
     cause to use a cursor.
cursor eventRoomReport IS
   Select Hotel.HotelName, Hotel.HotelID, Hotel.PhoneNumber, Room.RoomID,
     Room.RoomType, Room.Capacity
   from Hotel, Room
   where hotel.hotelid = room.hotelid and Hotel.State = Hotel_State;
--Parse the cursor.
state_hotel_row eventRoomReport%rowtype;
--Setting a boolean to determine if any data exists for this particular query in order to
     determine whether an exception will occur or not.
stateHotelFound boolean := false;
noHotelFoundinState exception;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 1: Courtney Burns. getStateHotelEventRoom
     Report');
--stating before the loop to ensure that this statement prints.
dbms_output.put_line('Hotels in this state are: ');
--Looping through the eventRoomReport cursor into state_hotel_row rowtype until no
     more records can be pulled, returning basic information about the hotel as well as
     more detailed information about the rooms.
for state_hotel_row IN eventRoomReport
loop
   stateHotelFound := true;
   dbms_output.put_line('Hotel Name: ' || state_hotel_row.HotelName);
   dbms_output.put_line('Hotel ID: ' || state_hotel_row.HotelID);
   dbms_output.put_line('Hotel Phone: ' || state_hotel_row.PhoneNumber);
   dbms_output.put_line('Event Room ID: ' || state_hotel_row.RoomID);
   dbms_output.put_line('Event Room Type: ' || state_hotel_row.RoomType);
   dbms_output.put_line('Event Room Capacity: ' || state_hotel_row.Capacity);
--This line exists to put a space between each printed record.
   dbms_output.put_line(' ');
end loop;
--Exception is raised if the state entered by the user does not have hotels.
if stateHotelFound = false then
   raise noHotelFoundinState;
end if;
--If exception is raised, the system prints error message.
EXCEPTION
   when noHotelFoundinState then
   dbms_output.put_line('No hotels were found in this state.');
End;
```

Member 2: Sharon Soh

6. Make an event reservation: Hotel ID, guest's name, start date, end date, event type, date of reservation, number of people attending, etc. Output: event reservation ID (this is called confirmation code in real-life). NOTE: Only one person can make an event reservation. However, the same person can make multiple reservations. Also make sure that the reserved hall has capacity that can hold the number of people attending. ***For example, for a conference of 500 people, a customer must reserve 2 medium halls and a large hall for each day of the conference, usually 3 consecutive days.***

```
Create sequence resSeq;
Create sequence cusSeq;

create or replace procedure newReservation (hotel_id in hotel.hotelID%type, nameF in
customer.NameFirst%type,
nameL in customer.NameLast%type, dateRes in Reservation.DateReserved%type, SDate in
Reservation.startdate%type,
EDate in reservation.enddate%type, Event_Name in reservation.eventname%type, Attending in
reservation.numattending%type)
is
  dateFormatException exception;
  partySizeException exception;
  Cust_exists number;
  Res_seq number;
  cursor availableRooms is (select roomid from room where capacity >= attending and hotelid = hotel_id
        minus
        (select r.roomid from room r, reservedrooms rr, reservation re
        where startdate between to_date(SDate, 'dd/mm/yyyy') and to_date(EDate, 'dd/mm/yyyy') and
r.roomid=rr.roomid and rr.resid = re.resid
        union
        select r.roomid from room r, reservedrooms rr, reservation re
        where startdate between to_date(SDate, 'dd/mm/yyyy') and to_date(EDate, 'dd/mm/yyyy') and
r.roomid=rr.roomid and rr.resid = re.resid)
    );
  availableRoomRow availableRooms%rowtype;
Loop_counter number := 0;
Begin
  dbms_output.put_line ('Team Member 2: Sharon Soh. newReservation');
  --make sure the beginning date is before the end date
  if SDate > EDate then
    Raise dateFormatException;
  End if;

  -- make sure the customer exists already. If not, add customer to database
  Select count(*) into cust_exists from customer where NameFirst = nameF and nameLast = nameL;
  If (cust_exists < 1) then
```

```
    Cust_exists := cusSeq.nextVal;
    Insert into customer (custid, nameFirst, nameLast, PhoneNumber) values (cust_exists, nameF, nameL,
'5555555555');
  End if;
  Select custid into cust_exists from customer where NameFirst = nameF and nameLast = nameL;

  --this will grab the last available room from the cursor
  --why the last? No reason.
  For availableRoomRow in availableRooms
  Loop
    Loop_counter := availableRoomRow.roomid;
  End loop;

  --if we found an available room, it's room id is stored in loop_counter, and we can create the reservation
  If loop_counter > 0 then
  --insert into the reservation table
  Res_seq:=RESSEQ.nextval;
  insert into Reservation (ResID, EventName, DateReserved, StartDate, EndDate, NumAttending,
ActiveStatus, CustID) values
(Res_seq, Event_Name, dateRes, Sdate, EDate, Attending, 1, cust_exists);

    Insert into reservedRooms(RoomID, resID) values (loop_counter, Res_seq);
    dbms_output.put_line('Your reservation id is: ' || Res_seq);

  Else
    Raise partySizeException;
 End If;
  -- exception to run if beginning and end dates are mixed up
  Exception
  When dateFormatException then
    dbms_output.put_line('Please enter the correct start and end dates');
  When partySizeException then
     dbms_output.put_line('Party size too big, please make multiple reservations.');
End;
```

7. Find an event reservation: Input is a person's name and date, hotel ID. Output is event reservation ID

```
Create or replace procedure findReservation (nameF IN varchar2, nameL IN varchar2,
dateRes in date, Hotel_ID in integer) is
cursor ReservationInfo is

Select  RESID ,
EVENTNAME ,
DATERESERVED ,
STARTDATE ,
ENDDATE ,
NUMATTENDING ,
ACTIVESTATUS ,
CUSTID  from Reservation
where Reservation.DateReserved = dateRes;
```

```
res_info_row ReservationInfo%rowtype;

resIDFound boolean := false;
noResIDFound exception;

Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 2: Sharon Soh. findReservation');

for res_info_row IN ReservationInfo
loop
resIDFound := true;
dbms_output.put_line('Reservation ID: ' || res_info_row.ResId);
end loop;
if resIDFound = false then
raise noResIDFound;
end if;
exception
when noResIDFound then
dbms_output.put_line('No reservations found');
end;
```

8. Cancel events: Input the event reservationID and mark the reservation as cancelled - do not delete

```
Create or replace procedure cancelEvents (res_id int)
Is
noReservationException exception;
Begin
--prints which team member worked on this procedure
dbms_output.put_line('Team Member 2: Sharon Soh. cancelEvents');
    update Reservation set ActiveStatus = 0 where resid = res_id;
    dbms_output.put_line('Reservation has been canceled');
Exception
When noReservationException then
dbms_output.put_line('Please enter a valid Reservation ID');
  end;
```

9. Show cancellations

```
Create or replace procedure showCancelations
IS
cursor eventCancelations IS
Select Hotel.HotelID, Hotel.HotelName, hotel.address, hotel.city, hotel.state,
hotel.zipcode, reservation.eventname, reservation.startdate,
reservation.enddate
from hotel, reservation, ReservedRooms, Room
where
hotel.hotelid = room.hotelid
and reservation.resid = reservedrooms.resid
and reservedrooms.roomid = room.roomid
and reservation.activestatus = 0;
canceledEventsRow eventCancelations%rowtype;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 2: Sharon Soh. showCancelations');
dbms_output.put_line('Canceled events: ');
for canceledEventsRow in eventCancelations
   loop
      dbms_output.put_line('Events cancelled are: ' || canceledEventsRow.hotelID ||
      canceledEventsRow.hotelname || canceledEventsRow.address ||
canceledEventsRow.city|| canceledEventsRow.state ||
      canceledEventsRow.zipcode || canceledEventsRow.eventname ||
canceledEventsRow.startdate||
      canceledEventsRow.enddate);
      end loop;
      end;
Set serveroutput on;
Execute showCancelations;
```

<u>Member 3: Sam Bennett</u>

10. Change an events date

setEventDate(EventID integer, newStartDate date, newEndDate date)

```
create or replace procedure setEventDate(eventId int, newStartDate date, newEndDate date) is
   newRoomId int;
   newRoomSize int;
   rowCount int;
   --The cursor will take our room id
   cursor availableRooms is (select roomid from room where capacity >= (select numattending
from reservation where resid = eventid) and hotelid = (select r.hotelid from room r, reservedrooms
rr where rr.resid = eventid and rr.roomid = r.roomid)
```

```
        minus
        (select r.roomid from room r, reservedrooms rr, reservation re
        where startdate between to_date(newStartDate, 'dd/mm/yyyy') and to_date(newEndDate,
'dd/mm/yyyy') and r.roomid=rr.roomid and rr.resid = re.resid
        union
        select r.roomid from room r, reservedrooms rr, reservation re
        where startdate between to_date(newStartDate, 'dd/mm/yyyy') and to_date(newEndDate,
'dd/mm/yyyy') and r.roomid=rr.roomid and rr.resid = re.resid)
    );
    availableRoomRow availableRooms%rowtype;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 3: Sam Bennett. setEventDate');

    newRoomId :=0;
    for availableRoomRow in availableRooms loop
        if newRoomId = 0 then
            newRoomId := availableRoomRow.roomid;
        end if;
    end loop;
    if newRoomId = 0 then
        --There are no rooms that fit the critera within the date, raise this exception.
        raise no_data_found;
    else
        update reservedrooms set roomid = newRoomId where resid = eventid;
        update reservation set startdate = newStartDate, enddate = newEndDate where resid =
eventid;
        dbms_output.put_line('Your new room number is ' || newRoomId || '.');
        commit;
    end if;
exception
    when no_data_found then
        dbms_output.put_line('There were no available rooms to switch your reservation to.');
end;
```

11. Change event room type

```
create or replace procedure setEventRoomType(eventId int, newRoomType varchar2) is
    newRoomId int;
    newRoomSize int;
    existingStartDate date;
    existingEndDate date;
    rowCount int;
    --The cursor will take the room id look for the hotel the room is in, look for available rooms by
seeing if either the start or end date exist within the date range.
    cursor availableRooms(eventStartDate date, eventEndDate date) is (select roomid from room
where capacity >= (select numattending from reservation where resid = eventid) and roomtype =
newRoomType and hotelid = (select r.hotelid from room r, reservedrooms rr where rr.resid =
eventid and rr.roomid = r.roomid)
        minus
        (select r.roomid from room r, reservedrooms rr, reservation re
        where startdate between to_date(eventStartDate, 'dd/mm/yyyy') and to_date(eventEndDate,
'dd/mm/yyyy') and r.roomid=rr.roomid and rr.resid = re.resid
        union
```

```
        select r.roomid from room r, reservedrooms rr, reservation re
        where startdate between to_date(eventStartDate, 'dd/mm/yyyy') and to_date(eventEndDate,
'dd/mm/yyyy') and r.roomid=rr.roomid and rr.resid = re.resid)
    );
    availableRoomRow availableRooms%rowtype;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 3: Sam Bennett. setEventRoomType');
    newRoomId :=0;
    select startdate, enddate into existingStartDate, existingEndDate from reservation where resid
= eventId;
    for availableRoomRow in availableRooms(existingStartDate,existingEndDate) loop
        if newRoomId = 0 then
            newRoomId := availableRoomRow.roomid;
        end if;
    end loop;
    if newRoomId = 0 then
        --There are no rooms that fit the critera within the date, raise this exception.
        raise no_data_found;
    else
        update reservedrooms set roomid = newRoomId where resid = eventid;
        dbms_output.put_line('Your new room number is ' || newRoomId || '.');
        commit;
    end if;
exception
    when no_data_found then
        dbms_output.put_line('There were no available rooms of the chosen room type that are
available and fit your current required occupance.');
end;
```

12. Show events of a specific type

```
create or replace procedure getEventsByType(eventType VARCHAR2) is
    cursor eventsByType is select r.resid, hotelname, address, city, state, zipcode, startdate,
enddate from reservation r, hotel h, reservedrooms rr, room r where r.resid = rr.resid and
rr.roomid = r.roomid and r.hotelid = h.hotelid and eventname = eventType;
    eventsByTypeRow eventsByType%rowtype;
    rowcount int;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 3: Sam Bennett. getEventsByType');
    rowcount := 0;
--loop through every row where our eventname equals the first argument
    for eventsByTypeRow in eventsByType loop
        dbms_output.put_Line('Reservation: ' || eventsByTypeRow.resid || ', Hotel: ' ||
eventsByTypeRow.hotelname || ', Addr: ' || eventsByTypeRow.address || '' ||
eventsByTypeRow.city || '-' || eventsByTypeRow.state || '-' || eventsByTypeRow.zipcode || ', Date:
' || eventsByTypeRow.startdate || ' thru ' || eventsByTypeRow.enddate);
        rowcount := rowcount + 1;
    end loop;
    if rowcount < 1 then
--raise this exception because we need to know if there were no entries and it wont raise on its
own!
        raise no_data_found;
```

```
    end if;
exception
    when no_data_found then
        dbms_output.put_Line('There are no events with this type!');
End;
```

13. Show events from a specific person by first name and last.

```
create or replace procedure getEventsByCustomer(FirstName VARCHAR2, LastName
VARCHAR2) is
    cursor eventsByType is select r.resid, eventname, hotelname from reservation r, hotel h,
reservedrooms rr, room r, customer c where r.resid = rr.resid and rr.roomid = r.roomid and
r.hotelid = h.hotelid and r.custid = c.custid and namefirst = FirstName and nameLast = LastName;
    eventsByTypeRow eventsByType%rowtype;
    rowcount int;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 3: Sam Bennett. getEventsByCustomer');
    rowcount := 0;
    for eventsByTypeRow in eventsByType loop
        dbms_output.put_Line('Reservation ID#: ' || eventsByTypeRow.resid || ' @ Hotel: ' ||
eventsByTypeRow.hotelname);
        rowcount := rowcount + 1;
    end loop;
    if rowcount < 1 then
--raise this exception because we need to know if there were no entries and it wont raise on its
own!
        raise no_data_found;
    end if;
exception
    when no_data_found then
        dbms_output.put_Line('There are no events under the name ' || FirstName || ' ' || LastName);
end;

execute getEventsByPerson('John','Goodman')
```

14. Display a full income report

```
create or replace procedure getincomreport is
    curyear number;
    nextyear number;
    monthlyTotal number;
    reservationTotal number;
    reservationCost number;
    serviceTotal number;
    serviceCost number;
    discountMult number;
    month varchar2(2000);
Begin
    --prints which team member worked on this procedure
    dbms_output.put_line ('Team Member 3: Sam Bennett. getincomreport');

    select to_char(sysdate, 'YYYY') as year into curyear from dual;
    nextyear := curyear+1;
```

```
--count through everymonth
   for monthcounter in 1..12 loop
--need month in a char format so we can add leading 0 to it
      month := to_char(monthcounter, '00');--lpad(monthcounter, 2, '0');
--trim off any whitespace incase something funky happened
      month := trim(month);
      monthlytotal := 0;
      dbms_output.put_line('Income for Month: ' || month || '-' || curyear || ' by event and service');
      dbms_output.put_line('-----------------------------------------------');
      dbms_output.put_line('Events');
      dbms_output.put_line('------');
      reservationTotal:= 0;
--loop through all reservations in month then print and add up their totals
      for reservation in (select r.resid, r.eventname, r.datereserved, r.startdate,(enddate -
startdate)*price as price from reservation r, reservedrooms rr, room where room.roomid =
rr.roomid and rr.resid = r.resid and to_char(r.datereserved, 'MM-YYYY') = month || '-' || curyear)
loop
         discountMult := 0;
         if months_between(reservation.datereserved, reservation.startdate) > 2 then
--set the discount to 10% off if they have a 2 month distance between the above
            discountMult := .1;
         end if;
         reservationCost := reservation.price - reservation.price*discountMult;
         dbms_output.put_line(reservation.eventname || ' - $' || reservationCost);
         reservationTotal := reservationTotal + reservationCost;
      end loop;
      dbms_output.put_line('');
      dbms_output.put_line('Services');
      dbms_output.put_line('--------');
      serviceTotal := 0;
--do a similar thing here as above but no need to worry about discounts
      for servicetype in (select servicetype,Sum(priceper*numattending) as mulitprice,
Sum(individualprice) as individualprice from service s, reservation r where s.resid = r.resid and
to_char(r.datereserved, 'MM-YYYY') = month || '-' || curyear group by servicetype) loop
         serviceCost := servicetype.mulitprice + servicetype.individualprice;
         dbms_output.put_line(servicetype.servicetype || ' - $' || servicecost);
         serviceTotal:= serviceTotal + serviceCost;
      end loop;
      monthlyTotal := reservationTotal + serviceTotal;
      dbms_output.put_line('');
      dbms_output.put_line('Total Income for Month: ' || month || '-' || curyear || ' - $' ||
monthlyTotal);
      dbms_output.put_line('');
   end loop;
   dbms_output.put_line('-----------------------------------------------');
   dbms_output.put_line('Income for Year: ' || nextyear || ' by event and service');
   dbms_output.put_line('-----------------------------------------------');
--count through everymonth again but in the next year!
   for monthcounter in 1..12 loop
--need month in a char format so we can add leading 0 to it
      month := to_char(monthcounter, '00');--lpad(monthcounter, 2, '0');
--trim off any whitespace incase something funky happened
      month := trim(month);
      monthlytotal := 0;
```

```
        dbms_output.put_line('Income for Month: ' || month || '-' || nextyear || ' by event and service');
        dbms_output.put_line('-----------------------------------------');
        dbms_output.put_line('Events');
        dbms_output.put_line('------');
        reservationTotal:= 0;
--loop through all reservations in month then print and add up their totals
        for reservation in (select r.resid, r.eventname, r.datereserved, r.startdate,(enddate -
startdate)*price as price from reservation r, reservedrooms rr, room where room.roomid =
rr.roomid and rr.resid = r.resid and to_char(r.datereserved, 'MM-YYYY') = month || '-' || nextyear)
loop
            discountMult := 0;
            if months_between(reservation.datereserved, reservation.startdate) > 2 then
--set the discount to 10% off if they have a 2 month distance between the above
                discountMult := .1;
            end if;
            reservationCost := reservation.price - reservation.price*discountMult;
            dbms_output.put_line(reservation.eventname || ' - $' || reservationCost);
            reservationTotal := reservationTotal + reservationCost;
        end loop;
        dbms_output.put_line('');
        dbms_output.put_line('Services');
        dbms_output.put_line('--------');
        serviceTotal := 0;
--do a similar thing here as above but no need to worry about discounts
        for servicetype in (select servicetype,Sum(priceper*numattending) as mulitprice,
Sum(individualprice) as individualprice from service s, reservation r where s.resid = r.resid and
to_char(r.datereserved, 'MM-YYYY') = month || '-' || nextyear group by servicetype) loop
            serviceCost := servicetype.mulitprice + servicetype.individualprice;
            dbms_output.put_line(servicetype.servicetype || ' - $' || servicecost);
            serviceTotal:= serviceTotal + serviceCost;
        end loop;
        monthlyTotal := reservationTotal + serviceTotal;
        dbms_output.put_line('');
        dbms_output.put_line('Total Income for Month: ' || month || '-' || nextyear || ' - $' ||
monthlyTotal);
        dbms_output.put_line('');
    end loop;
end;
```

Member 4: Nicholas Franklin

15. Add a Service to an Event

```
create or replace procedure addServiceToReservation( res_id integer, service_type
varchar2, dateOccuring date)
IS
-- how much this should cost, either per-person or one-time payment
costOf number;
dateBegin date;
dateEnd date;
```

```
alreadyExistsCounter number;
serviceNotFound exception;
dateOutOfBounds exception;
alreadyExistsException exception;
service_name service.servicetype%type := service_type;

Begin
  dbms_output.put_line('Member 4: Nicholas Franklin. addServiceToReservation.');
  --check to make sure the service's dateOccuring is between the reservation date
  select startdate, enddate into dateBegin, dateEnd from reservation where resid =
res_id;
  if NOT (dateOccuring BETWEEN dateBegin AND dateEnd) then
    raise dateOutOfBounds;
  end if;

  --perform some string cleaning and normalizing, trimming any whitespace and taking
into account any case issues
  service_name := TRIM(service_name);
  service_name := LOWER(service_name);

  -- since there are only 5 types of events, then we can do a simple case statement for
how much it will cost. Where it will go (PricePer or IndividualPrice) will be determined
after. This keeps memory costs low. Input is case insensitive.
  CASE
    When service_name = 'breakfast' THEN costOf := 10;
    When service_name = 'lunch' THEN costOf := 20;
    When service_name = 'dj' THEN costOf :=500;
    When service_name = 'singer' THEN costOf := 2000;
    When service_name = 'pop band' THEN costOf := 10000;
    ELSE raise serviceNotFound;
    -- If it gets to ELSE, then it's obviously not a service we support, so raise an exception
  END CASE;

  --if the program gets to this point, it's a real service, so let's make it look nice!
  service_name := initcap(service_name);

   --one problem though, what if it already exists in there? We don't want to double
charge!
   select count(*) into alreadyExistsCounter from service where resID = res_id AND
serviceType = service_name AND dateOF = dateOccuring;
   if (alreadyExistsCounter > 0) THEN
    raise alreadyExistsException;
   End if;
```

```
  --If the cost is less than or equal to 20, then it's a price per person total, and should be
put in as such. Otherwise, put it in the IndividualPrice value
  If (costOf <= 20) THEN
    Insert into service values(serSeq.nextVal, service_name, costOf, 0, dateOccuring,
res_id);
  Else
    Insert into service values(serSeq.nextVal, service_name, 0, costOf, dateOccuring,
res_id);
  End if;

  EXCEPTION
    When serviceNotFound THEN
      dbms_output.put_line(service_type || ' is not a service we support.');
    When dateOutOfBounds THEN
      dbms_output.put_line(dateOccuring || ' is not within reservation period.');
    When alreadyExistsException THEN
      dbms_output.put_line('Service already included, do not want to double charge!');
End;
```

16. Reservation Services Report

```
create or replace procedure getReservationServicesReport(Res_ID integer)
is
        -- Multiple services would be multiple rows. Have to use a cursor for that.
        cursor serviceReport is SELECT serviceType FROM service, reservation
WHERE service.resID = reservation.resID and reservation.resID = res_ID;
        --parse the cursor
        serviceRow serviceReport%rowtype;
        -- store the number attending for report, less space than storing a rowtype
        attendingNumber Reservation.NumAttending%type;
begin
    dbms_output.put_line('Member 4: Nicholas Franklin. getReservationServicesReport.');
        -- If there's nothing here, it goes to no_data_found exception
        SELECT numAttending into attendingNumber FROM Reservation WHERE resID
= res_ID;
        dbms_output.put_line('There are ' || attendingNumber || ' people attending
Reservation ' || res_ID || '.');

        --Loop through the cursor into the rowtype until no more available, listing the
ServiceType as we go.
        dbms_output.put_line('Reservation ' || res_ID || ' contains: ');
        FOR serviceRow in serviceReport
        LOOP
```

```
                dbms_output.put_line(serviceRow.ServiceType);
        END LOOP;

        --State that this reservation has no services.
        exception
                when No_Data_Found then
                        dbms_output.put_line('No services for this reservation');
end;
```

17. Show Specific Service Report

```
create or replace procedure getServiceGeneralReport(service_type IN varchar2)
IS
   -- create a cursor with everything needed, again traversing so many tables
   cursor reservationReport is select r.resid, h.hotelname, r.eventname, r. datereserved,
r.startdate, r.enddate, r.numattending, r.activestatus, r.custid
   from service s, reservation r, reservedrooms rr, room o, hotel h
   where h.hotelid = o.hotelid and o.roomid = rr.roomid and rr.resid = r.resid and r.resid =
s.resid
      and s.servicetype = service_type;
   resRepRow reservationReport%rowtype;

   --service rowtype for checking to make sure the service exists
   testForEx service%rowtype;
Begin
   dbms_output.put_line('Member 4: Nicholas Franklin. getServiceGeneralReport.');
   --testing for data, will raise no_data_found exception if not found
   select * into testForEx from service where servicetype = service_type;

   dbms_output.put_line('Service ' || service_type || ' is in:');

   --many reservations will have this service type, so we navigate through the cursor,
displaying each
   FOR resRepRow in reservationReport
   LOOP
      dbms_output.put_line('Reservation ID: ' || resRepRow.resid || ' which is at ' ||
resRepRow.hotelname || ' hosting a ' || resRepRow.eventname || ' from ' ||
resRepRow.startdate || ' until ' || resRepRow.enddate || ' and will have ' ||
resRepRow.numattending || ' people for Customer ID: ' || resRepRow.custid);
   END LOOP;

 Exception
  When No_Data_Found then
   dbms_output.put_line('No reservations with that service');
```

```
End;
```

18. Services Income Report

```
create or replace procedure getServiceIncomeReport(hotel_id integer)
IS
   --Holy Joins Batman! I have to traverse every single table except customer to get to Service.
        --We need the number attending, the price for the event if not individual (PricePer, and individual price.
        --Also, no point in counting it if the reservation is no longer active!
        cursor incomeReport IS select Reservation.NumAttending, Service.PricePer, IndividualPrice
        from Hotel, Room, ReservedRooms, Reservation, Service
        where Hotel.HotelID = Room.HotelID and ReservedRooms.RoomID = Room.RoomID and Reservation.ResID = ReservedRooms.ResID and Service.ResID = Reservation.ResID
        and Hotel.HotelID = hotel_id and Reservation.ActiveStatus = 1;

        --rowtype to capture each cursor row
        reportRow incomeReport%rowtype;
        --this will be added to throughout the loop
        totalIncome real := 0;
        --this will be the calculator for each service, so the totalIncome is only updated once in the loop and not modified otherwise.
        currentTotal real := 0;
        --test for existing hotel using the smallest memory in the Hotel Table
         testState varchar2(2) := '  ';

begin
   dbms_output.put_line('Member 4: Nicholas Franklin. getServiceIncomeReport.');
        --test for hotel existing, if it doesn't exist then it raises a No_Data_Found exception
   select state into testState from Hotel where HotelID = hotel_id;

   FOR reportRow in incomeReport
        LOOP
                --check if it's individually priced or flat rate
                if reportRow.IndividualPrice > 0 then
                        --flat rate, just put number in currentTotal
                        currentTotal := reportRow.IndividualPrice;
                else
                        --price per person, multiply price by NumAttending
                        currentTotal := reportRow.NumAttending * reportRow.PricePer;
```

```
                end if;

                -- add to totalIncome
                totalIncome := totalIncome + currentTotal;
                --reset currentTotal for next loop
                currentTotal := 0;
        END LOOP;

        dbms_output.put_line('Hotel ' || hotel_id || ' has a Total Income for all the Services
in all the Reservations totalling: $' || totalIncome);

    exception
        when no_data_found then
            dbms_output.put_line('Error: Hotel not found.');
End;
```

Member 5: Mark Makris

19. Shows available events

```
--takes in a hotel ID
create or replace procedure showRooms(HotelIDIn in integer) is
--cursor for all rooms
cursor roomTypes is
select distinct RoomType
from Room;
--row from room
roomT roomtypes%rowtype;
--number of rooms
numRooms integer;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 5: Mark Makris. showRooms');
--loops through all the rooms
for roomT in roomTypes
loop
--gets count for room type
select count(RoomID) into numRooms
from room
where roomType = roomT.roomType and HotelID = HotelIDIn;
--prints room information for hotel
dbms_output.put_line('Hotel: ' || HotelIDIn || ' has ' || numRooms || ' ' || roomT.roomType || 's
free');
end loop;
end;
```

20. Invoice of event

```
create or replace procedure getInvoice(resIDIn in integer) is
    --variable for how far in advance the reservation was made
    reservDist number;
    --variables for customer first and last name
    rNameFirst customer.nameFirst%type;
    rNameLast customer.nameLast%type;
    --variables for room and price
    rRoom room.roomID%type;
    rPriceR room.price%type;
    --variables for service price and date
    rServiceDate service.dateOf%type;
    rService service.serviceType%type;
    rServiceAmt number;
    --variables for cost discount and cost - dics
    subCost number;
    rDicount number;
    totalCost number;
    --variables for reservation dates
    rResDate reservation.dateReserved%type;
    rStartDate reservation.startDate%type;
    rEndDate reservation.endDate%type;
    --amount of people attending
    attendance reservation.numattending%type;
    --cursor for rooms in reservation
    cursor rooms is
        select r.ROOMID ROOMID,
        r.ROOMTYPE ROOMTYPE,
        r.CAPACITY CAPACITY,
        r.PRICE PRICE,
        r.HOTELID HOTELID,
        rr.ROOMID ROOMID_0,
        rr.RESID RESID
        from room r, reservedRooms rr
        where r.roomID = rr.roomID and rr.resID = resIDIn;
        roomRow rooms%rowType;
    --cursor for services in reservations
    cursor services is
        select *
        from service s
        where s.resID = resIDIn;
        serviceRow services%rowType;
    Begin
            --prints which team member worked on this procedure
            dbms_output.put_line ('Team Member 5: Mark Makris. getInvoice');
        --setting costs to 0 to begin
```

```
        subCost := 0;
        rDicount := 0;
        totalCost := 0;
        --gets the reservation start and end dates
        select numattending into attendance
        from reservation
        where resid = residin;
        --gets all the dates
        select dateReserved, startDate, endDate into rResDate, rStartDate, rEndDate
            from reservation r
            where r.resID = resIDIn;
        --get reservation distance
        reservDist := (rResDate-rStartDate)/30;
        --gets the customer first and last name
        select nameFirst, nameLast into rNameFirst, rNameLast
            from customer c, reservation r
            where c.custID = r.custID and r.resID = resIDIn;
        dbms_output.put_line('Customer: ' || rNameFirst || ' ' || rNameLast);
        --loops through rooms in reservation
        for roomRow in rooms
            loop
            --prints room information
            dbms_output.put_line('Room #: ' || roomRow.roomID || ', costs per day: ' ||
roomRow.price);
            --adds to cost of reservation
            subCost := subCost + roomRow.price*(rStartDate-rEndDate);
        end loop;
        --loops through services in reservation
        for serviceRow in services
            loop
            --prints service information and checks the type of event
            if serviceRow.pricePer = 0 then
                dbms_output.put_line('Service: ' || serviceRow.serviceType || ', costs for service: ' ||
serviceRow.individualPrice || ' on date: ' || serviceRow.dateOf);
            else
                dbms_output.put_line('Service: ' || serviceRow.serviceType || ', costs for service: ' ||
serviceRow.pricePer || ' on date: ' || serviceRow.dateOf);
            end if;
            --adds to cost of reservation
            subCost := subCost + serviceRow.pricePer*attendance + serviceRow.individualPrice;
        end loop;
        --checks if the reservation was reserved two months in advance.
        If ((rResDate-rStartDate)/30) >= 2 then
            --applies a 10% discount
            totalCost := subCost*.9;
            dbms_output.put_line('Discount: ' || (subCost*.1));
        else
            --print for no discount
```

```
        dbms_output.put_line('No discount');
    End if;
    --prints total cost
    dbms_output.put_line('Total cost: ' || subCost);
end;
```

21. Income for state

```
create or replace procedure getStateIncomeReport(stateIn in varchar2) is
--variables to output
eventIdO integer;
roomPriceO real;
serviceCostsOpp real;
serviceCostsOind real;
totalEventO real;
grandTotalO real;
hasEvent number;
--get event id for events in state
cursor eventids is
select rr.resid
from reservedrooms rr, room r, hotel h
where stateIn = h.state and h.hotelID = r.hotelID and r.roomID = rr.roomID;
--variables for reservation dates
rResDate reservation.dateReserved%type;
rStartDate reservation.startDate%type;
Begin
--prints which team member worked on this procedure
dbms_output.put_line ('Team Member 3: Mark Makris. getStateIncomeReport');
grandTotalO := 0;
--loops through the event ids of the events in the desired state
for idi in eventids
loop
--gets the reservation start and end dates
select dateReserved, startDate into rResDate, rStartDate
from reservation r
where r.resID = idi.resID;
--gets the cost of the rooms for that event
select sum(r.price) into roomPriceO
from reservedrooms rr, room r
where resID = idi.resid and r.roomID = rr.roomID;
--gets the cost of the services for that event
select sum(s.priceper)*r.numattending into serviceCostsOpp
from service s, reservation r
where s.resID = idi.resid and r.resID = s.resid;
--gets the cost of individually priced services
select sum(s.individualprice) into serviceCostsOind
from service s
where resID = idi.resid;
```

```
--output line of id, rooms, services and total cost of event
dbms_output.put_line('id: ' || idi.resid || ', room costs ' || roomPriceO || ', price costs ' ||
(serviceCostsOpp + serviceCostsOind) || ', total costs ' || (roomPriceO + (serviceCostsOpp +
serviceCostsOind)));
--adds a discount if qualified
If ((rResDate-rStartDate)/30) >= 2 then
grandTotalO := (roomPriceO + (serviceCostsOpp + serviceCostsOind)) * .9;
else
grandTotalO := (roomPriceO + (serviceCostsOpp + serviceCostsOind));
end if;
end loop;
--if there is a discount add it
If ((rResDate-rStartDate)/30) >= 2 then
grandTotalO := grandTotalO * .9;
end if;
--prints grand total and state
dbms_output.put_line('Grand total: ' || grandTotalO);
dbms_output.put_line('For State: ' || stateIn);
end;
```

Test Scenario:

```
dbms_output.put_line(Mark);
--M5: Show available halls at each hotel (multiple calls)
execute showRooms(0);
execute showRooms(1);
execute showRooms(2);
execute showRooms(3);
--M5: Provide event invoice for Mrs. Brown
--declare
--pid integer;
--rid integer;
--begin
--Select custID into pid
--From customer
--Where nameLast = 'Brown';
--Select resID into rid
--From reservation
--Where custID = pid;
--getInvoice(pid);
--End;
getInvoice(1);
--declare
--pid integer;
--rid integer;
--begin
--Select custID into pid
--From customer
```

```
--Where nameLast = 'Zero';
--Select resID into rid
--From reservation
--Where custID = pid;
--getInvoice(pid);
--end;
getInvoice(2);
--M5: Income for AK, MD (two calls)
execute getStateIncomeReoprt('AK');
execute getStateIncomeReoprt('MD');
```

--M1:

--Add a new hotel, H0 in Fairbanks Alaska. Find it, and add 2 small, 2 medium and 2 large halls.

execute addNewHotel('H0', 'James', 'Jones', '60 Berkshire Ave.', 'Fairbanks', 'AK', 99701, '9074505088');

```
Team Member 1: Courtney Burns. addNewHotel
```

execute findHotels ('60 Berkshire Ave.', 'Fairbanks', 'AK', 99701);

```
Team Member 1: Courtney Burns. findHotels
The hotel at this address has the id: 1


PL/SQL procedure successfully completed.
```

execute addEventRoom(1, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(1, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(1, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(1, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(1, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(1, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

--Add a new hotel, H1 in Honolulu, Hawaii. Find it and add 5 large halls.

execute addNewHotel('H1', 'Kevin', 'Hart', '4477 Indiana Avenue', 'Honolulu', 'HI', 96814, '8085936393');

```
Team Member 1: Courtney Burns. addNewHotel


PL/SQL procedure successfully completed.
```

execute findHotels ('4477 Indiana Avenue', 'Honolulu', 'HI', 96814);

```
Team Member 1: Courtney Burns. findHotels
The hotel at this address has the id: 2
```

execute addEventRoom(2, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(2, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(2, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(2, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(2, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

--Add a new hotel, H2 in Baltimore, MD. Add 1 small, 1 medium and 1 large hall.

execute addNewHotel('H2', 'Abigail', 'Sherwood', '2618 Five Points', 'Baltimore', 'MD', 21250, '4432937553');

```
Team Member 1: Courtney Burns. addNewHotel


PL/SQL procedure successfully completed.
```

execute findHotels ('2618 Five Points', 'Baltimore', 'MD', 21250);

```
Team Member 1: Courtney Burns. findHotels
The hotel at this address has the id: 3


PL/SQL procedure successfully completed.
```

execute addEventRoom(3, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(3, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(3, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

--Add a new hotel, H3 in Annapolis, MD. Add 5 small, 5 medium and 5 large hall.

execute addNewHotel('H3', 'Janet', 'Toth', '2651 Blue Spruce Lane', 'Annapolis', 'MD', 21401, '4102953704');

```
Team Member 1: Courtney Burns. addNewHotel


PL/SQL procedure successfully completed.
```

execute findHotels ('2651 Blue Spruce Lane', 'Annapolis', 'MD', 21401);

```
Team Member 1: Courtney Burns. findHotels
The hotel at this address has the id: 4


PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'small hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'medium hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom

PL/SQL procedure successfully completed.
```

execute addEventRoom(4, 'large hall');

```
Team Member 1: Courtney Burns. addEventRoom


PL/SQL procedure successfully completed.
```

--Find hotel in Annapolis, MD

execute findHotels(null, 'Annapolis','MD', null);

```
Team Member 1: Courtney Burns. findHotels
A hotel at this address has the id: 4



PL/SQL procedure successfully completed.
```

--Display information for hotel H0

execute getHotelInfo(1);

```
Team Member 1: Courtney Burns. getHotelInfo
Hotel with ID: 1
Hotel Name: H0
Hotel Contact: James Jones
Address: 60 Berkshire Ave., Fairbanks, AK, 99701
Hotel Phone Number: 9074505088



PL/SQL procedure successfully completed.
```

--Display information for hotel H55 (which does not exist!)

-- if Primary Key 1 = H0, then Primary Key 56 = H55

execute getHotelInfo(56);

```
Team Member 1: Courtney Burns. getHotelInfo
No hotels with this ID were found.


PL/SQL procedure successfully completed.
```

--M2: Mrs. Brown makes a reservation at hotel H2 for a wedding for Dec 7. 100 people are attending.

Execute newReservation(2, 'Jane', 'Brown', '5-Dec-2018', '7-Dec-2019', '7-Dec-2019', 'Wedding', 100);

--M2: Mr. Zero makes a reservation at hotel H0 for a Birthday for March 9. 10 people are attending.

Execute newReservation (1, 'John', 'Zero', '5-Dec-2018', '9-Mar-2018', '9-Mar-2018', ‘Birthday’, 10);

--M2: Mr. Zero makes a reservation at hotel H1 for a Birthday for July 9. 50 people are attending.

```
        Execute newReservation(2, 'John', 'Zero', '5-Dec-2018', '9-Jul-2019', '9-Jul-2019',
'Birthday', 50);
        --M2: Mrs. Brown makes a reservation at hotel H3 for a Conference for March 10-13.
1000 people are attending.
                Execute newReservation(4, 'Jane', 'Brown', '5-Dec-2018', '10-Mar-2019',
'13-Mar-2019', 'Conference', 500);
                Execute newReservation(4, 'Jane', 'Brown', '5-Dec-2018', '10-Mar-2019',
'13-Mar-2019', 'Conference', 500);
        --[new] M2: Mr. Cyber makes a reservation at hotel H2 for a Hackathon for May 1. 100
people are attending
                Execute newReservation(1, 'Joseph', 'Cyber', '5-Dec-2018', '1-May-2019',
'1-May-2019', 'Hackathon', 100);

        --M3: Show events reserved by Mr. Zero
                execute getEventsByCustomer('John','Zero');
        --M3: Show events reserved by Mr. Brown
        execute getEventsByCustomer('John','Brown');

        --M2: Cancel Mr. Zero's reservation at H0.
                Execute cancelEvents (2);
        --M3: Show events reserved by Mr. Zero
                execute getEventsByCustomer('John','Zero');
        --M2: Show cancellations
                Execute showCancelations;

        --M1: Report hotels and rooms in MD including total capacity per event room type per
hotel.
                execute getStateHotelEventRoomReport ('MD');
        --M3: Produce total monthly income report
                execute getincomreport();
        --M3: Change event Date of the Hackathon from May 1, to Dec 20.
        execute setEventDate(6, to_date('01/05/2019', 'dd/mm/yyyy'), to_date('20/12/2019',
'dd/mm/yyyy'));
        --M3: Produce total monthly income report (again)
        execute getincomreport();
        --M3: Change room type of Hackathon to a large room
                setEventRoomType(6,'Large Hall');
        --M3: Produce total monthly income report
        execute getincomreport();
        --M4: Add breakfast for all attendees of the Conference
                execute addServiceToReservation(4, 'breakfast', '10-Mar-19');
                execute addServiceToReservation(4, 'breakfast', '11-Mar-19');
                execute addServiceToReservation(4, 'breakfast', '12-Mar-19');
                execute addServiceToReservation(4, 'breakfast', '13-Mar-19');
                execute addServiceToReservation(5, 'breakfast', '10-Mar-19');
                execute addServiceToReservation(5, 'breakfast', '11-Mar-19');
                execute addServiceToReservation(5, 'breakfast', '12-Mar-19');
                execute addServiceToReservation(5, 'breakfast', '13-Mar-19');
```

```
--M4: Show reservation services report for the Conference event
        execute getReservationServicesReport(4);
--M4: Add a DJ to all birthday events (you may need to make multiple calls)
        execute addServiceToReservation(2, 'dj', '9-Mar-19');
        execute addServiceToReservation(3, 'dj', '9-Jul-19');
--M4: Add a pop band to the wedding
        execute addServiceToReservation(1, 'pop band', '7-Dec-2018');
--M4: Show Specific service report for DJ
        execute getServiceGeneralReport('dj');
--M4: Show services income report
        execute getServiceIncomeReport(0);
        execute getServiceIncomeReport(1);
        execute getServiceIncomeReport(2);
        execute getServiceIncomeReport(3);



--M5: Show available halls at each hotel (multiple calls)
execute showRooms(1);
execute showRooms(2);
execute showRooms(3);
execute showRooms(4);
--M5: Provide event invoice for Mrs. Brown
        execute getInvoice(1);
        Execute getInvoice(4);
--M5: Provide event invoice for Mr. Zero
        execute getInvoice(2);
        Execute getInvoice(3);
--M5: Income for AK, MD (two calls)
        execute getStateIncomeReport('AK');
execute getStateIncomeReport('MD');
```

```
delete from reservedrooms;
delete from room;
delete from service;
delete from hotel;
delete from reservation;
delete from customer;

drop sequence newHotelSeq;
drop sequence cusseq;
drop sequence eventseq;
drop sequence hotelseq;
drop sequence resseq;
drop sequence serseq;

create sequence newHotelSeq;
create sequence cusseq;
create sequence eventseq;
create sequence hotelseq;
create sequence resseq;
create sequence serseq;
```