**Team**

**Jabberwocky**



**Team Members**

Johnny Hsu

Justin Lebreck

Taoran Ma

Sameer Patil

Josh Ziegler

Michelle Ziegler

# Table of Contents

[Table of Contents 2](#_Toc437637894)

[Chapter 1: Requirements Document 4](#_Toc437637895)

[Chapter 2: ER Diagram 7](#_Toc437637896)

[Chapter 2: ER Data Dictionary 8](#_Toc437637897)

[Chapter 3: Relational Schema 14](#_Toc437637898)

[Chapter 3: Relational Data Dictionary 16](#_Toc437637899)

[Chapter 4: Data population and Queries 25](#_Toc437637900)

[Query 1**:** Top 10 destination cities for Co.Host overall 25](#_Toc437637901)

[Query 2**:** Companies' respective spends through the Co.Host platform over given time periods 26](#_Toc437637902)

[Query 3**:** Departmental spends for a given company through the Co.Host platform over a given time period 26](#_Toc437637903)

[Query 4**:** Top 5 destination cities for a given company over a given time period 27](#_Toc437637904)

[Query 5**:** Co.Host incentives paid to each user over given time periods 28](#_Toc437637905)

[Query 6**:** Current awards levels of individual users 29](#_Toc437637906)

[Query 7**:** Top 10 booked rooms and addresses 30](#_Toc437637907)

[Query 8: Rooms and addresses that have never been booked over time periods 31](#_Toc437637908)

[Query 9**:** Number of bookings for every city over a given period of time 32](#_Toc437637909)

[Query 10: How often do employees stay together ? 33](#_Toc437637910)

[Chapter 5: Triggers and Procedures 36](#_Toc437637911)

[Trigger 1: Cancellation 36](#_Toc437637912)

[Trigger 2: Department And Company Spend 39](#_Toc437637913)

[Procedure 1: Make Payment 43](#_Toc437637914)

[Procedure 2 Revenue Per Period 47](#_Toc437637915)

[Procedure 3: Available Rooms 49](#_Toc437637916)

[Procedure 4: Room Rate 58](#_Toc437637917)

[Chapter 6: Interface and Reports 62](#_Toc437637918)

[Chapter 7: Conclusions and implementation plan 69](#_Toc437637919)

[Implementation plan: 69](#_Toc437637920)

[Hours and Costs Breakdown 70](#_Toc437637921)

[/\*Appendix to Chapter 3: Create Statements, Sequences, Sequence Triggers, Insert Statements, Alter Table Constraints\*/ 71](#_Toc437637922)

[/\*Create Table Statements\*/ 71](#_Toc437637923)

[/\*Sequence Creation Script\*/ 77](#_Toc437637924)

[/\*Sequence Trigger Creation Script\*/ 79](#_Toc437637925)

[/\*Table Insert Script\*/ 83](#_Toc437637926)

[/\*Alter Table Statements\*/ 294](#_Toc437637927)

[/\*Appendix to Chapter 3: Triggers and Procedures\*/ 297](#_Toc437637928)

[/\*Trigger 1 – Cancellation\*/ 297](#_Toc437637929)

[/\*Trigger 2 – Department and Company Spend\*/ 298](#_Toc437637930)

[/\*Procedure 1 – Make Payment\*/ 302](#_Toc437637931)

[/\*Procedure 2 – Revenue Per Period\*/ 305](#_Toc437637932)

[/\*Procedure 3 – Available Rooms\*/ 307](#_Toc437637933)

[/\*Procedure 4 – Room Price\*/ 315](#_Toc437637934)

# Chapter 1: Requirements Document

**Introduction:**

Our client, Co.Host, is an internet startup headquartered in Tucson, AZ that organizes employee travel between a Company’s locations around the country, and some day, the world. They have specific ideas about how they intend for their startup to facilitate this process, but they have not been able to find someone to set up their database. They expect to start booking travel for their pilot customers within the next few months. It is important to them that they have a working solution soon.  We are working with the CEO, Dustin Cox.

Co.Host currently does not have any database set up. They were planning to use an Excel spreadsheet before we approached them for this project. Other than Travelers and Hosts (discussed below), the 3 partners of Co.Host will need to have administrative rights to the system and the ability to run or view reports.

Co.Host will need to track employees who are signed up for their program. They will have to login to access their account. Each account will be tied to a unique email address.  The user’s first and last name, chosen password, gender, birth date, company, company department, and assigned office will also be recorded. To assure the security of the system the password information will be tracked in a separate entity than the rest of the login and employee information and cannot be the same as the last 3 passwords. To log in to the website, employees will use their email address and password to securely access their profile and available listings for their company. The employee will have a unique ID number assigned by Co.Host. Finally, Co.Host wants to maintain a photo of each employee to display on the website.

Co.Host needs to track the company by name and will have to assign it a unique ID. The system needs to keep a running total spent by each company, and have that total broken down by company department such as finance, sales, and operations.

Each company may have an appointed company contact. This contact could change over time and Co.Host may need to know who was a company contact at a specific moment in time. For company contact people Co.Host would like to track contact name (first and last), contact phone, contact begin date, contact end date, and contact email. Contact email will be unique for each contact, but a contact ID will be assigned to each contact by Co.Host to serve as the unique identifier.

For each company, there will be at least two offices. The address (including longitude and latitude) will be tracked in order to calculate distance to the chosen stay location. They will assign a unique number to that office.

The employees will be broken down into two categories: travelers and hosts. Employees can sign up with only the intent to participate in the future without actually hosting or traveling. Participating employees may also be a part of either or both groups. Co.Host needs to track ratings specific to each role.

Each host will offer accommodations at their home. They may have more than one home to offer to travelers.  Each home will have certain characteristics that apply to the entire building, such as wifi, any pets, partner/spouse in the home, children, smoking, building type, kitchen access, washer/dryer access, etc. It will also belong to a certain type of neighborhood. Neighborhoods will be categorized by name and assigned a unique ID. Additional information relating to nearby attractions, restaurants, parks, etc. can be tracked in an open comment section. There may be several addresses located in the same neighborhood.

Co.Host uses the US Federal per diem rates to set room prices. The per diem rates are based on county (referred to as regions), so host addresses must include the county. The rates are also based on travel month.

Each house will offer rooms to travelers. The room must have a private bathroom to be eligible. Each room will have specific characteristics like square footage, number of beds, price, and size of the bed(s). Each room will have pictures. The rooms will be identified based on an assigned room number and the address of the building.

A traveler reserves a room for a certain time period at a specific rate. The rate will be determined by applying a set discount to the region price. A Traveler can reserve many rooms at a time and a room can be reserved by many travelers over time.

At the time a reservation is made, the company will provide payment. For each payment Co.Host would like to track payment date and payment price. A payment must be cancelled more than 24 hours before the beginning of the stay to receive a refund. A cancellation may cover all or part of a reservation. For example a reservation that is made for a business trip may cover many locations and a traveler may find out they no longer need to travel to one of those locations.

Once a stay has ended both the traveler and the host receive a survey. Upon completion of the questionnaire the traveler/host receive their chosen reward currency. Co.Host then uses the results of the questionnaire to calculate the traveler rating and host rating.  If an employee’s host or traveler rating goes below a predetermined threshold, the employee is no longer eligible to book or offer accommodations.

Co.Host needs to track relationships between travelers and hosts such as who has hosted whom and lodged with whom. For this relationship the date of the stay will be used to differentiate multiple stays within the same relationship.

Rewards are tracked by a reward system, i.e. Choice Privileges, Marriott Rewards...etc. An employee may only choose one preferred reward system at a time and the same reward system could be chosen by many employees. Each reward system needs a unique reward ID, reward type, and reward name. Highest reward level and current reward level will need to be tracked for each employee as well as their current number of reward points. The number of points required to achieve the next reward level also needs to be maintained.

# Chapter 2: ER Diagram



## Chapter 2: ER Data Dictionary

|  |  |  |
| --- | --- | --- |
| **ADDRESS** | Entity Class to model the address of host's place |  |
| • addrs\_city | The city of the address | Attribute should not contain NULL values |
| • addrs\_country | The country of the address | Attribute should not contain NULL values |
| • addrs\_ID | The identifying number of the address | Identifying Attribute |
| • addrs\_latitude | The latitude of the address | Attribute should not contain negative and NULL values |
| • addrs\_longitude | The longitude of the address | Attribute should not contain negative and NULL values |
| • addrs\_state | The state of the address | Attribute should not contain NULL values |
| • addrs\_street | The street of the address | Attribute should not contain NULL values |
| • addrs\_zip | The zipcode | Attribute should not contain NULL values |
| • avail\_rms | How many rooms available | Attribute should not contain negative values |
| • bldg\_type | The type of the building | Attribute should not contain NULL values |
| • children | If there is children | Attribute should not contain NULL values |
| • common\_area | The common area for the rooms | Attribute should not contain NULL values |
| • kitchen | If there is a kitchen | Attribute should not contain NULL values |
| • other | If there is other amentities | Attribute should not contain NULL values |
| • pets | If there is pet | Attribute should not contain NULL values |
| • pool | If there is a pool | Attribute should not contain NULL values |
| • smoking | If the travaler can smoke | Attribute should not contain NULL values |
| • television | If there is a television | Attribute should not contain NULL values |
| • washer\_dryer | If there is a washer and dryer | Attribute should not contain NULL values |
| • wifi | If there is a wifi | Attribute should not contain NULL values |
| **AUTHORIZES** | Relationship that models the password anthorizes employee |  |
| **BELONGS TO** | Relationship that models that an employee belongs to a department |  |
| **CAN HAVE** | Relationship that models the reservation can have cancellation |  |
| **CANCELLATION** | Entity Class to model the the cancellation mechanism |  |
| • canc\_date | The date when traveler cancels the reservation | Attribute should not contain NULL values |
| • canc\_ID | Identifying number of the cancellation | Identifying Attribute |
| • canc\_limit | The deadlind of the cancellation refund | Attribute should not contain NULL values |
| • canc\_reason | The reason why traveler cancels the reservation | Attribute should not contain NULL values |
| • ref\_amt | The refund amount of the cancellation | Attribute should not contain NULL values |
| **CO\_CONTACT** | Entity Class to model the contact information of client company contact |  |
| • cont\_begin\_date | The begin date that a contact person work in the company | Attribute should not contain NULL values |
| • cont\_email | The email address of contact person | The phone or email address either should not be NULL |
| • cont\_end\_date | The end date that a contact person works in the company |  |
| • cont\_first | The first name of contact person | Attribute should not contain NULL values |
| • cont\_ID | The identifying number of the contact person | Identifying Attribute |
| • cont\_last | The last name of contact person | Attribute should not contain NULL values |
| • cont\_name | The name of contact person | Attribute should not contain NULL values |
| • cont\_phone | The phone number of contact person | The phone or email address either should not be NULL |
| **COMPANY** | Entity Class to model the company information |  |
| • co\_ID | Identifying number of the company | Identifying Attribute |
| • co\_name | The name of company | Attribute should not contain NULL values |
| • total\_co\_spend | The total spend of each company | Attribute should not contain negative values |
| **COMPLETES** | Relationship that models employees complete surveys to get traveler/host results |  |
| **CONTAINS** | Relationship that models the address contains rooms |  |
| **DETERMINES** | Relationship that models the region determins the region rate |  |
| **DEPARTMENT** | Entity Class to model the department information |  |
| • dept\_ID | Identifying number of the department | Identifying Attribute |
| • dept\_name | The name of a department in a company | Attribute should not contain NULL values |
| • total\_dept\_spend | The total spend of each department | Attribute should not contain negative values |
| **EMPLOYEE** | Entity Class to model the employee information |  |
| • active\_flg | A boolean value to determine wether the employee is currentlly employed | Attribute should not contain NULL values |
| • banned | The function used if the employee is banned to access the system, yes or no | Attribute should not contain NULL values, constraining attribute for room reservation |
| • current\_rew\_lv | Current reward level, Gold, Silver, Platinum | Attribute should not contain negative values |
| • email | The email of employee | Attribute should not contain NULL values |
| • emp\_dob | The birthday of the employee | Attribute should not contain NULL values |
| • emp\_first | The first name of employee | Attribute should not contain NULL values |
| • emp\_ID | Identifying number of the employee | Identifying Attribute |
| • emp\_last | The last name of employee | Attribute should not contain NULL values |
| • emp\_name | The name of employee | Attribute should not contain NULL values |
| • emp\_photo | The address of each employee's photo album | Attribute should not contain NULL values |
| • emp\_points | The points earned by employee (host& travelers) | Attribute should not contain negative values |
| • highest\_rew\_lv | Highest reward level, Gold, Silver, Platinum | Attribute should not contain NULL values |
| • pts\_to\_nxt\_rwd | Points to next reward | Attribute should not contain negative values |
| **GENERATES** | Relationship that models how the reservation generates payments |  |
| **HAS** | Relationship that models that each company has department(s) |  |
| **GENDER** | Entity Class to model the gender of the employee |  |
| • gender\_ID | Identifying number of the gender | Identifying Attribute |
| • gender\_descr | The description of gender |  |
| **HOST** | Sub Class to model the host |  |
| • host\_rtg | The rating of the host | Attribute should not contain negative values |
| **HOST\_RESULTS** | Sub Class to model the host questionaire results |  |
| •host\_rtg | The rating of the host | Attribute should not contain NULL values |
| **IDENTIFIES AS** | Relationship that employee identifies as wich gender |  |
| **IN** | Relationship that models office in the region |  |
| **LOCATED IN** | Relationship that models the address located in a neighborhood |  |
| **MAINTAINS** | Relationship that models how a company may maintains a company contact |  |
| **MAKES** | Relationship that models the department makes payment(s) |  |
| **NEIGHBORHOOD** | Entity Class to model the the neighborhood attributes |  |
| • nbhd\_feat | The features of the neighborhood, eg., "parking", "restuarant" |  |
| • nbhd\_ID | Identifying number of the neighborhood | Identifying Attribute |
| • nbhd\_name | The name of the neighborhood |  |
| • nbhd\_type | The type of the neighborhood, eg., "suburbs", "urban" |  |
| **OFFERS** | Relationship that models the address offered by host |  |
| **OFFICE** | Entity Class to model the office information. |  |
| • off\_address | The address of office | Attribute should not contain NULL values |
| • off\_city | The city contained in the address of office | Attribute should not contain NULL values |
| • off\_country | The country contained in the address of office | Attribute should not contain NULL values |
| • off\_ID | Identifying number of the office | Identifying Attribute |
| • off\_latitude | The latitude derived from the address of office | Attribute should not contain NULL values |
| • off\_longitude | The longitude derived from the address of office | Attribute should not contain NULL values |
| • off\_state | The state contained in the address of office | Attribute should not contain NULL values |
| • off\_street | The street contained in the address of office | Attribute should not contain NULL values |
| • off\_zip | The zip code contained in the address of office | Attribute should not contain NULL values |
| **PASSWORD** | Entity Class to model login in password |  |
| • passward\_date | The date of creating the password | Attribute should not contain NULL values |
| • sequence\_ID | Identifying number of the password | Identifying Attribute |
| • user\_password | The log in password of each employee | Attribute should not contain NULL values |
| **PAYMENT** | Entity Class to model the traveler |  |
| • pmnt\_date | The date of the payment | Attribute should not contain NULL values |
| • pmnt\_ID | Identifying number of the payment | Identifying Attribute |
| • pmnt\_price | The price of payment | Attribute should not contain negative values |
| **REGION** | Composite Class to model the region nearby the host's address |  |
| • county | The county of the region | Attribute should not contain NULL values |
| • reg\_ID | Identifying number of the region | Identifying Attribute |
| **REGION\_RATE** | Weak Entity Class to model the region rate |  |
| • region\_begin\_date | Identifying number of the payment | Identifying Attribute |
| • region\_rate | The rate of each region | Attribute should not contain NULL values |
| **RESERVATION** | Weak Entity Class to model the reservation traveler makes |  |
| • dept\_ID | Identifying number of the department | User enter attribute |
| • isCanc | Whether the reservation is canclled | Default 'N' value |
| • reserv\_end | The end date of the reservation | Attribute should not contain NULL values, triggering attribute for surveys. |
| • reserv\_ID | The identifying number of the reservation | Identifying Attribute |
| • reserv\_price | The price of reservation | Attribute should not contain NULL values |
| • reserv\_start | The start date of the reservation | Attribute should not contain NULL values |
| **REWOARDS** | Relationship that models the employee rewards by the payment they make |  |
| **REWARD\_LOG** | Entity Class to model the reward system |  |
| • reward\_amt | Derived attribute that tracks each reward disbursement | Attribute cannot be negative |
| • reward\_date | Partial Identifier in the reward log to differentiate reward disbursements | Identifying Attribute |
| **REWARD\_SYSTEM** | Entity Class to model the reward system chosen by the employee |  |
| • reward\_ID | The identifying number of the reward system | Identifying Attribute |
| • reward\_type | The type of reward eg., "Cash","Points", etc... | Attribute should not contain NULL values |
| • reward\_name | The name of the specific reward eg., "Marriott Rewards", "Cash", "Wyndam Rewards" etc... | Attribute should not contain NULL values |
| **ROOM** | Entity Class to model the room in the host's place |  |
| • bed\_count | How many beds available | Attribute should not contain negative values |
| • bed\_size | The size of the bed | Attribute should not contain NULL values |
| • room\_ID | The identifying number of room | Identifying Attribute |
| • room\_picture | Multivalued attribute to store the address(es) of the picture(s) of the room being offered. | Multiple Attribute should not contain NULL values |
| • sqr\_feet | How many squre feet of the room | Attribute should not contain negative values |
| **RUNS** | Relationship that models the company runs offices |  |
| **SELECTS** | Relationship that models the employee selects the reward |  |
| **SURVEY** | Typing Class to model the surveys that each traveler and host fills in |  |
| • answers | The answers of the survey | Multiple Attribute should not contain NULL values |
| • survey\_id | The identifying number of a survey | Identifying Attribute |
| • comp\_date | Completion date of the survey | Attribute should not contain NULL values |
| **SURVEY\_QUESTIONS** | Instantiation Class to model the what questions are in each survey |  |
| • isActive | Whether the question is active | Attribute should not contain NULL values |
| • question | The question of the survey | Attribute should not contain NULL values |
| • question\_add\_date | The adding date of the question to survey | Attribute should not contain NULL values |
| • question\_ID | The identifying number of a survey | Identifying Attribute |
| • survey\_type | The type of the survey | Attribute should not contain NULL values |
| **TRAVELER** | Sub Class to model the traveler |  |
| • traveler\_ID | Identifying number of the traveler | Identifying Attribute |
| • traveler\_rtg | The rating of the traveler | Attribute should not contain negative values |
| **TIES TO** | Relationship that models reservations tie to room |  |
| **TRIGGERS** | Relationship that models how the survey is triggered by the reservation |  |
| **WITHIN** | Relationship that address within the region |  |
| **WORK IN** | Relationship that models employee works in office |  |

# Chapter 3: Relational Schema

ADDRESS (addrs\_ID, addrs\_country, addres\_zip, addrs\_state, addrs\_county, addr\_street, addrs\_city, addrs\_longitude, addrs\_latitude, Avail\_rms, nbhd\_ID, host\_ID, children, common\_area, kitchen, other, owner\_occ, pets, pool, smoking, television, washer\_dryer, wifi)

Foreign Key (nbhd\_ID) references to NEIGHBORHOOD

Foreign Key (host\_ID) references to HOST

ANSWERS (question\_ID, survey\_ID, answer)

Foreign Key (question\_ID) references to SURVEY\_QUESTIONS

Foreign Key (survey\_ID) references to SURVEY

CANCELLATION (canc\_ID, canc\_limit, canc\_reason, canc\_date, ref\_amt, reserve\_ID)

Foreign Key reserve\_ID references to RESERVATION

CO\_CONTACT (cont\_ID, cont\_first, cont\_last, cont\_begin\_date, cont\_end\_date, cont\_email, co\_ID)

Foreign Key (co\_ID) references to COMPANY

COMPANY (co\_ID, co\_name, total\_co\_spend)

CONTACT\_PHONE (cont\_ID, cont\_phone, current\_ph, phone\_type)

Foreign Key (cont\_ID) references to CO\_CONTACT

DEPARTMENT (dept\_ID, dept\_name, tot\_dept\_spend, co\_ID)

Foreign Key (co\_ID) references to COMPANY

EMPLOYEE (emp\_ID, emp\_first, emp\_last, banned, current\_rew\_lv, emp\_points, emp\_DOB, gender\_ID, emp\_photo, highest\_rew\_lv, pts\_to\_nxt\_rwd, reward\_ID, dept\_ID, off\_ID, active\_flg, email)

Foreign Key (login\_ID) references to LOGIN

Foreign Key (reward\_ID) references to REWARD\_SYSTEM

Foreign Key (dept\_ID) references to DEPARTMENT

Foreign Key (off\_ID) references to OFFICE

Foreign Key (gender\_ID) references to GENDER

GENDER (gender\_ID, gender\_descr)

HOST (host\_ID, host\_rtg)

Foreign Key (host\_ID) references to EMPLOYEE (emp\_ID)

NEIGHBORHOOD (nbhd\_ID, nbhd\_type, nbhd\_feat, nbhd\_name)

OFFICE (off\_ID, off\_longitude, off\_latitude, off\_country, off\_zip, off\_state, off\_city, off\_street, co\_ID, reg\_ID)

Foreign Key (co\_ID) references to COMPANY

Foreign Key (reg\_ID) references to REGION

PASSWORD (emp\_ID, sequence ID, password\_date, user\_password)

Foreign Key (emp\_ID) references to EMPLOYEE

PAYMENT (pmnt\_ID, pmnt\_date, pmnt\_price, dept\_ID)

Foreign Key (pmnt\_ID) references to DEPARTMENT

REGION\_RATE (reg\_ID, rate\_begin\_date, region\_rate)

Foreign Key (reg\_ID) references to REGION

REGION (reg\_ID, county)

RESERVATION (reserve\_ID, reserve\_start, reserve\_end, traveler\_ID, pmnt\_ID, room\_ID, dept\_ID, Iscanc, off\_ID, room\_price)

Foreign Key (traveler\_ID) references to TRAVELER

Foreign Key (pmnt\_ID) references to PAYMENT

Foreign Key (room\_ID) references to ROOM

Foreign Key (dept\_ID) references to DEPARTMENT

Foreign Key (off\_ID) references to OFFICE

REWARD\_LOG (emp\_ID, pmnt\_ID, reward\_date, reward\_amt)

Foreign Key (pmnt\_ID) references to DEPARTMENT

Foreign Key (emp\_ID) references to EMPLOYEE

REWARD\_SYSTEM (reward\_ID, reward\_type, reward\_name)

ROOM (room\_ID, sqr\_feet, bed\_count, addrs\_ID, private\_bthrm, room\_comments)

Foreign Key (addrs\_ID) references to ADDRESS

ROOM\_BED (bed\_no, room\_ID, bed\_size)

Foreign Key (room\_ID) references to ROOM

ROOM\_PICTURE (room\_ID, room\_pictures)

Foreign Key (room\_ID) references to ROOM

SURVEY (survey\_ID, comp\_date, survey\_type, emp\_ID, reserve\_ID)

Foreign Key (emp\_ID) references to EMPLOYEE

Foreign Key (reserve\_ID) references to RESERVATION

SURVEY\_QUESTIONS (question\_ID, question, question\_add\_date, survey\_type, isActive)

TRAVELER (traveler\_ID, traveler\_rtg)

Foreign Key (traveler\_ID) references to EMPLOYEE (emp\_ID)

## Chapter 3: Relational Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Schema Construct** | **Data type** | **Constraint** |
| ADDRESS | Relation representing the entity class ADDRESS | |
| addrs\_city | VARCHAR(200) | Not Null |
| addrs\_county | VARCHAR(100) | Not Null |
| addrs\_country | VARCHAR(200) | Not Null |
| addrs\_ID | VARCHAR(12) | Primary Key |
| addrs\_latitude | VARCHAR(100) | Not Null |
| addrs\_longitude | VARCHAR(100) | Not Null |
| addrs\_state | VARCHAR(100) | Not Null |
| addrs\_street | VARCHAR(200) | Not Null |
| addrs\_zip | VARCHAR(10) | Not Null |
| avail\_rms | VARCHAR(2) | Not Null |
| bldg\_type | VARCHAR(200) | Not Null |
| children | VARCHAR(2) | Not Null |
| common\_area | VARCHAR(2) | Not Null |
| host\_ID | VARCHAR(12) | Foreign Key references NEIGHBORHOOD |
| kitchen | VARCHAR(2) | Not Null |
| nbhd\_ID | VARCHAR(6) | Foreign Key references HOST |
| other | VARCHAR(2000) | Not Null |
| owner\_occ | VARCHAR(2) | Not Null |
| pets | VARCHAR(2) | Not Null |
| pool | VARCHAR(2) | Not Null |
| smoking | VARCHAR(2) | Not Null |
| television | VARCHAR(2) | Not Null |
| washer\_dryer | VARCHAR(2) | Not Null |
| wifi | VARCHAR(2) | Not Null |
| FD: addrs\_ID --> addrs\_city, addrs\_county, addrs\_country, addrs\_distance, addrs\_latitude, addrs\_longitutde, addrs\_state, addrs\_street, addrs\_zip, avail\_rms, host\_ID, nbhd\_ID, wifi, pets, children, kitchen, common\_area, owner\_occ, television, boldg\_type, sasher\_dryer, pool, other | | |
|
|
|
|  |  |  |
| ANSWERS | Relation representing the multiple attribute of ANSWERS | |
| answer | VARCHAR(2000) | Not Null |
| question\_ID | VARCHAR(10) | Foreign Key references SURVEY\_QUESTIONS |
| survey\_ID | VARCHAR(10) | Foreign Key references SURVEY |
| Primary Key Constraints:question\_ID, survey\_ID | | |
| FD: question\_ID, survey\_ID --> answer | | |
|  |  |  |
| AVAILABLE\_ROOM | Relation representing **staging table** for external of data | |
| addrs\_city | VARCHAR(200) |  |
| addrs\_latitude | VARCHAR(100) |  |
| addrs\_longitude | VARCHAR(100) |  |
| addrs\_state | VARCHAR(100) |  |
| addrs\_street | VARCHAR(200) |  |
| avail\_check\_date | DATE |  |
| avail\_ID | VARCHAR(12) | Primary Key |
| bldg\_type | VARCHAR(200) |  |
| children | VARCHAR(2) |  |
| common\_area | VARCHAR(2) |  |
| kitchen | VARCHAR(2) |  |
| nvhd\_feat | VARCHAR(2000) |  |
| nbhd\_name | VARCHAR(200) |  |
| hbhd\_type | VARCHAR(100) |  |
| other | VARCHAR(2000) |  |
| owner\_occ | VARCHAR(2) |  |
| pets | VARCHAR(2) |  |
| pool | VARCHAR(2) |  |
| reserv\_end | DATE |  |
| reserv\_start | DATE |  |
| room\_ID | VARCHAR(12) | Foreign Key references ROOM |
| room\_price | DECIMAL(24,6) |  |
| smoking | VARCHAR(2) |  |
| television | VARCHAR(2) |  |
| traveler\_ID | VARCHAR(12) | Foreign Key references TRAVELER |
| washer\_dryer | VARCHAR(2) |  |
| smoking | VARCHAR(2) |  |
| Primary Key Constraints:room\_ID, travaler\_ID | | |
| FD: avail\_ID, room\_ID, travaler\_ID --> reserv\_start, reserv\_end, room\_price, addrs\_street, addrs\_cit, addrs\_state, nbhd\_name, nabhd\_type, nbhd\_feat, wifi, pets, children,kitchen, common\_area, owner\_occ, television, bldg\_type, smoking, washer\_dryer, pool, other, addrs\_latitude, addrs\_longitude, avail\_check\_date | | |
|
|
|
|  |  |  |
| CANCELLATION | Relation representing the entity calss CANCELLATION | |
| canc\_date | DATE | Not Null |
| canc\_ID | VARCHAR(12) | Primary Key |
| canc\_limit | DATE | Not Null |
| canc\_reason | VARCHAR(2000) |  |
| ref\_amt | DECIMAL(24,6) | Not Null |
| reserve\_ID | VARCHAR(12) | Foreign key references RESERVATION |
| FD: canc\_ID --> canc\_date, canc\_limit, canc\_reasaon, ref\_amt, reserve\_ID | | |
|  |  |  |
| CO\_CONTACT | Relation representing the entity calss CO\_CONTACT | |
| co\_ID | VARCHAR(8) | Foreign key references COMPANY |
| cont\_begin\_date | DATE | Not Null |
| cont\_email | VARCHAR(200) | Not Null |
| cont\_end\_date | DATE |  |
| cont\_first | VARCHAR(200) | Not Null |
| cont\_ID | VARCHAR(8) | Primary Key |
| cont\_last | VARCHAR(200) | Not Null |
| FD: cont\_ID --> cont\_begin\_date, cont\_email, cont\_end\_date, cont\_first, cont\_last, co\_ID | | |
|  |  |  |
| COMPANY | Relation representing the entity calss COMPANY | |
| co\_ID | VARCHAR(8) | Primary Key |
| co\_name | VARCHAR(200) | Not Null |
| total\_co\_spend | DECIMAL(24,6) | Not Null |
| FD: co\_ID --> co\_name, total\_co\_spend | | |
|  |  |  |
| CONTACT\_PHONE | Relation representing the multiple attribute CONTACT\_PHONE | |
| cont\_ID | VARCHAR(8) | Foreign Key references CO\_CONTACT |
| cont\_phone | VARCHAR(20) |  |
| current\_ph | VARCHAR(2) |  |
| phone\_type | VARCHAR(50) |  |
| Primary Key Constraints: cont\_ID | | |
| FD: cont\_ID --> cont\_phone, current\_ph, phone\_type | | |
|  |  |  |
| DEPARTMENT | Relation representing the entity calss DEPARTMENT | |
| co\_ID | VARCHAR(8) | Foreign Key references COMPANY |
| dept\_ID | VARCHAR(9) | Primary Key |
| dept\_name | VARCHAR(100) | Not Null |
| tot\_dept\_spend | DECIMAL(24,6) | Not Null |
| FD: dept\_ID --> dept\_name, tot\_dept\_spend | | |
|  |  |  |
| EMPLOYEE | Relation representing the entity class EMPLOYEE | |
| active\_flg | VARCHAR(2) | Default 'Y' |
| banned | VARCHAR(2) | Not Null |
| current\_rew\_lv | VARCHAR(50) | Not Null |
| dept\_ID | VARCHAR(9) | Foreign Key references DEPARTMENT |
| email | VARCHAR(100) | Not Null |
| emp\_dob | DATE | Not Null |
| emp\_first | VARCHAR(40) | Not Null |
| emp\_ID | VARCHAR(12) | Primary Key |
| emp\_last | VARCHAR(40) | Not Null |
| emp\_photo | VARCHAR(2000) | Not Null |
| emp\_points | DECIMAL(24,6) | Not Null |
| gender\_ID | VARCHAR(12) | Foreign Key references GENDER |
| highest\_rew\_lv | VARCHAR(100) | Not Null |
| off\_ID | VARCHAR(8) | Foreign Key references OFFICE |
| points\_to\_nxt\_rwd | DECIMAL(24,6) | Not Null |
| reward\_ID | VARCHAR(12) | Foreign Key references REWARD\_SYSTEM |
| FD: emp\_ID --> banned, current\_rew\_lv, dept\_ID, email, emp\_dob, emp\_first, emp\_last, emp\_photo, emp\_points, gender\_ID, highest\_rew\_lv, off\_ID, points\_to\_nxt\_rwd, reward\_ID, active\_flg | | |
|
|
|  |  |  |
| GENDER | Relation representing the entity class GENDER | |
| gender\_descr | VARCHAR(100) |  |
| gender\_ID | VARCHAR(12) | Primary Key |
| FD: gender\_ID --> descr | | |
|  |  |  |
| HOST | Relation represeting the subclass HOST | |
| host\_ID | VARCHAR(12) | Foreign Key references EMPLOYEE |
| host\_rtg | DECIMAL(24,6) | Not Null |
| Primary Key Constraint: host\_ID | | |
| FD: host\_ID --> host\_rtg | | |
|  |  |  |
|  |  |  |
| NEIGHBORHOOD | relation representing the entity class NEIGHBORHOOD | |
| nbhd\_feat | VARCHAR(2000) |  |
| nbhd\_ID | VARCHAR(6) | Primary Key |
| nbhd\_name | VARCHAR(200) |  |
| nbhd\_type | VARCHAR(100) |  |
| FD: nbhd\_ID --> nbhd\_feat, nbhd\_name, nbhd\_type | | |
|  |  |  |
| OFFICE | relation representing the entity class OFFICE | |
| co\_ID | VARCHAR(8) | Foreign Key references COMPANY |
| off\_city | VARCHAR(200) | Not Null |
| off\_country | VARCHAR(200) | Not Null |
| off\_ID | VARCHAR(8) | Primary Key |
| off\_latitude | VARCHAR(100) | Not Null |
| off\_longitude | VARCHAR(100) | Not Null |
| off\_state | VARCHAR(100) | Not Null |
| off\_street | VARCHAR(200) | Not Null |
| off\_zip | VARCHAR(10) | Not Null |
| reg\_ID | VARCHAR(12) | Foreign Key references REGION |
| FD: off\_ID--> co\_ID, off\_city, off\_country, off\_latitude, off\_longitude, off\_state, off\_street, reg\_ID | | |
|
|  |  |  |
| PASSWORD | Relation representing the weak entity class of PASSWORD | |
| emp\_ID | VARCHAR(12) | Foreign Key references EMPLOYEE |
| password\_date | DATE | Not Null |
| sequence\_ID | VARCHAR(10) | Primary Key |
| user\_password | VARCHAR(254) | Not Null |
| Primary Key Constraint: emp\_ID | | |
| FD: sequence\_ID, emp\_ID --> password\_date, user\_password | | |
|  |  |  |
| PAYMENT | Relation representing the entity class of PAYMENT | |
| dept\_ID | VARCHAR(9) | Foreign Key references DEPARTMENT |
| pmnt\_date | DATE | Not Null |
| pmnt\_ID | VARCHAR(12) | Primary Key |
| pmnt\_price | DECIMAL(24,6) | Not Null |
| FD: pmnt\_ID --> pmnt\_date, pmnt\_price | | |
|  |  |  |
| REGION\_RATE | Relation representing entity class of REGION\_RETA | |
| rate\_begin\_date | DATE | Primary Key |
| reg\_ID | VARCHAR(12) | Foreign Key references REGION |
| region\_rate | DECIMAL(24,6) |  |
| Primary Key Constraint: reg\_ID | | |
| FD: rate\_begin\_date, reg\_ID --> region\_rate | | |
|  |  |  |
| REGION\_RAW | Relation representing **staging table** for external of data | |
| apr16 | DECIMAL(24,6) |  |
| aug16 | DECIMAL(24,6) |  |
| county | VARCHAR(300) |  |
| dec15 | DECIMAL(24,6) |  |
| feb16 | DECIMAL(24,6) |  |
| jan16 | DECIMAL(24,6) |  |
| jul16 | DECIMAL(24,6) |  |
| jun16 | DECIMAL(24,6) |  |
| mar16 | DECIMAL(24,6) |  |
| may16 | DECIMAL(24,6) |  |
| nov15 | DECIMAL(24,6) |  |
| oct15 | DECIMAL(24,6) |  |
| primary\_dest | VARCHAR(300) |  |
|  |  |  |
| REGION | Relation representing the composite class of REGION | |
| county | VARCHAR(300) | Not Null |
| reg\_ID | VARCHAR(12) | Primary Key |
| FD: reg\_ID --> county | | |
|  |  |  |
| RESERVATION | Relation representing the entity class of RESERVATION | |
| dept\_ID | VARCHAR(9) | Foreign Key references to DEPARTMENT |
| Iscanc | VARCHAR(2) | Default 'N' |
| off\_ID | VARCHAR(8) | Foreign Key references to OFFICE |
| pmnt\_ID | VARCHAR(12) | Foreign Key references to PAYMENT |
| reserve\_end | DATE |  |
| reserve\_ID | VARCHAR(12) | Primary Key |
| reserve\_start | DATE | Not Null |
| room\_ID | VARCHAR(12) | Foreign Key references to ROOM |
| room\_price | DECIMAL(24,6) | Not Null |
| traveler\_ID | VARCHAR(12) | Foreign Key references to TRAVELER |
| FD: reserve\_ID --> dept\_ID, iscanc, off\_ID, pmnt\_ID, reserve\_end, reserve\_start, room\_ID, room\_price, traveler\_ID | | |
|
|  |  |  |
| REVENUE\_PER\_CITY | relation representing **staging table** for external of data | |
| city | VARCHAR(200) |  |
| enddate | DATE |  |
| revenue\_per\_city\_ID | VARCHAR(12) | Primary Key |
| startdate | DATE |  |
| totalrevenue | DECIMAL(24,6) |  |
| FD: revenue\_per\_city\_ID --> city, enddate, startdate, totalrevenue | | |
|  |  |  |
| REWARD\_LOG | Relation representing the weak entity class of REWOARD\_LOG | |
| emp\_ID | VARCHAR(12) | Foreign Key references to EMPLOYEE |
| pmnt\_ID | VARCHAR(12) | Foreign Key references to PAYMENT |
| reward\_amt | DECIMAL(24,6) | Not Null |
| reward\_date | DATE | Primary Key |
| Primary Key Constraint: emp\_ID, pmnt\_ID | | |
| FD: reward\_date, emp\_ID, pmnt\_ID --> reward\_amt | | |
|  |  |  |
| REWARD\_SYSTEM | Relation representing the entity class of REWARD\_SYSTEM | |
| reward\_ID | VARCHAR(12) | Primary Key |
| reward\_name | VARCHAR(254) | Not Null |
| reward\_type | VARCHAR(40) | Not Null |
| FD: reward\_ID --> reward\_name, reward\_type | | |
|  |  |  |
| ROOM | Relation representing the entity class of ROOM | |
| addrs\_ID | VARCHAR(12) | Foreign Key references to ADDRESS |
| bed\_count | DECIMAL(3,0) | Not Null |
| private\_bthrm | VARCHAR(3) | Not Null |
| room\_comments | VARCHAR(2000) | Not Null |
| room\_ID | VARCHAR(12) | Primary Key |
| sqr\_feet | DECIMAL(24,6) |  |
| FD: room\_ID --> bed\_count, private\_bthrm, room\_comments, sqr\_feet, addrs\_ID | | |
|  |  |  |
| ROOM\_BED | Relation representing the multiple attribute of ROOM\_BED | |
| bed\_no | VARCHAR(6) | Primary Key |
| bed\_size | VARCHAR(20) | Not Null |
| room\_ID | VARCHAR(12) | Foreign Key references to ROOM |
| Primary Key Constraint: room\_ID | | |
| FD: bed\_no, room\_ID --> bed size | | |
|  |  |  |
| ROOM\_PICTURE | Relation representing the multiple attribute of ROOM\_PICTURE | |
| room\_ID | VARCHAR(12) | Foreign Key references to ROOM |
| room\_pictures | VARCHAR(2000) | Not Null |
| Primary Key Constraint: room\_ID | | |
| FD: pitrue\_no, room\_ID --> room\_prictures | | |
|  |  |  |
| ROOM\_RATE | Relation representing the **stating table** of ROOM\_RATE | |
| rate\_query\_ID | VARCHAR(12) | Primary Key |
| room\_ID | VARCHAR(12) | Foreign Key references to ROOM |
| room\_date | DATE |  |
| room\_rate | DECIMAL(24,6) |  |
| run\_date | DATE |  |
| traveler\_ID | VARCHAR(12) | Foreign Key references to TRAVELER |
| Primary Key Constraint: room\_ID, traveler\_ID | | |
| FD: rate\_query\_ID, room\_ID, traveler\_ID --> room\_date, room\_rate, run\_date | | |
|  |  |  |
| SURVEY | Relation representing the typing class of SURVEY | |
| emp\_ID | VARCHAR(12) | Foreign Key references to EMOLOYEE |
| comp\_date | DATE | Not Null |
| survey\_ID | VARCHAR(10) | Primary Key |
| survey\_type | VARCHAR(100) | Not Null |
| reserve\_ID | VARCHAR(12) | Foreign Key references to RESERVATION |
| FD: survey\_ID --> emp\_ID, survey\_comp\_date, survey\_type, reserve\_ID | | |
|  |  |  |
| SURVEY\_QUESTIONS | Relation representing the instantiation class of SURVEY | |
| isActive | VARCHAR(2) | Not Null |
| question | VARCHAR(2000) | Not Null |
| question\_add\_date | DATE | Not Null |
| question\_ID | VARCHAR(10) | Primary Key |
| survey\_type | VARCHAR(10) | Not Null |
| FD: question\_ID --> isActive, question, question\_add\_date, survey\_type | | |
|  |  |  |
| TRAVELER | Relation representing the subclass of TRAVELER | |
| traveler\_ID | VARCHAR(12) | Primary Key |
| traveler\_rtg | DECIMAL(24,6) | Not Null |
| FD: traveler\_ID --> traveler\_rtg | | |

# Chapter 4: Data population and Queries

**Chapter 4**

## Query 1**:** Top 10 destination cities for Co.Host overall

**Explanation:**

Our client, Co.Host, wants to know the top destination cities for its travelers. We query this data based on the reservation data from table RESERVATION excluding the data in the CANCELLATION table, which contains cancelled reservations. By joining the Table ADDRESS, ROOM, and RESERVATION, we can get all of the details of reservation info for all cities, such as city and reservation ID. Then by using the Group By statement with the aggregation function Count, we can obtain the total number of reservations per city. In the end, by using the function Rownum, we select the top 10 most traveled to cities.

**Output:**

|  |  |
| --- | --- |
| **City** | **Total\_No\_Reservation** |
| San Diego | 4 |
| Del Mar | 3 |
| Palo Alto | 3 |
| Los Angeles | 3 |
| San Francisco | 3 |
| Costa Mesa | 2 |
| Mountain View | 2 |
| Granite Bay | 2 |
| Pleasanton | 2 |
| Santa Ana | 2 |

**SQL Script:**

Select \*

from

(

Select addrs\_city City, count(reserv\_ID) Number\_Of\_Reservation

From ADDRESS, ROOM, RESERVATION

Where ADDRESS.addrs\_ID = ROOM.addrs\_ID

And ROOM.room\_ID = RESERVATION.room\_ID

AND reserv\_ID not in (Select reserv\_ID from CANCELLATION)

Group by addrs\_city

Order by count(reserv\_ID) desc

)

where rownum <= 10;

## Query 2**:** Companies' respective spends through the Co.Host platform over given time periods

**Explanation:**

Our client, Co.Host, wants to know the respective spends through the Co.Host platform over the given time periods. We query this data based on the payment data from PAYMENT table, excluding the payment data in the CANCELLATION table, which contains cancelled reservations. By joining the Table RESERVATION, PAYMENT, DEPARTMENT, and COMPANY, we get all of the details of the payment/spend information for all companies. These details include the company ID, company name, payment date, and payment amount. We then use the Where clause to choose the payment start date and end date. In the end using the Group By statement and the aggregation function Sum, we obtain the total spend amount per company.

**Output:**

|  |  |  |
| --- | --- | --- |
| **COMPANY\_ID** | **Company\_Name** | **Partial\_Spend\_Amount** |
| CO\_00001 | Intuit | 7462.8 |
| CO\_00004 | Microsoft | 1680.3 |
| CO\_00002 | SAP | 5461.2 |

**SQL Script:**

Select COMPANY.co\_ID Company\_ID, co\_name Company\_Name, sum(pmnt\_price) Partial\_Spend\_Amount

From RESERVATION, PAYMENT, DEPARTMENT, COMPANY

Where PAYMENT.dept\_ID = DEPARTMENT.dept\_ID

And DEPARTMENT.co\_ID = COMPANY.co\_ID

And PAYMENT.pmnt\_ID = RESERVATION.pmnt\_ID

And pmnt\_date Between '01-JAN-15' and '31-DEC-15'

And iscanc = 'N'

Group By COMPANY.co\_ID, co\_name

Order by co\_name;

## Query 3**:** Departmental spends for a given company through the Co.Host platform over a given time period

**Explanation:**

Our client, Co.Host, wants to know departmental spends for a given company through the Co.Host platform over a given time period. We query this starting with the payment data from the PAYMENT table, excluding the payment in the CANCELLATION table. We then join the RESERVATION, PAYMENT, DEPARTMENT, and COMPANY tables to get the payment information for all companies including company name, department name, payment date, and payment amount. Then we use the Where clause to choose the payment start date and end date. Finally, using the Group By statement with the aggregation function Sum, we can obtain the total amount spent per department for a given company.

**Output:**

|  |  |  |
| --- | --- | --- |
| **Company\_Name** | **Department\_Name,** | **Partial\_Spend\_Amount** |
| Intuit | Administration | 900 |
| Intuit | Coporate Strategy and Development | 900 |
| Intuit | Customer Service and Support | 1306.8 |
| Intuit | Legal | 1054.8 |
| Intuit | Marketing | 1054.8 |
| Intuit | Product Management | 900 |
| Intuit | Project and Program Management | 673.2 |
| Intuit | Sales | 673.2 |

**SQL Script:**

Select co\_name COMPANY\_NAME, dept\_name DEPARTMENT\_NAME, Sum (pmnt\_price) Partial\_Spend\_Amount

From PAYMENT, DEPARTMENT, COMPANY

Where PAYMENT.dept\_ID = DEPARTMENT.dept\_ID

And DEPARTMENT.co\_ID = COMPANY.co\_ID

And (pmnt\_date >= '01-JAN-15' And pmnt\_date <= '01-JAN-16')

And Co\_name = 'Intuit'

Group by co\_name, dept\_name

Order by dept\_name ;

## Query 4**:** Top 5 destination cities for a given company over a given time period

**Explanation:**

Our client, Co.Host, wants to know the top destination cities where travelers have stayed for a given company over a given time period. We query this data based on the reservation data from the RESERVATION table, excluding the data in the CANCELLATION table, which contains reservations canceled by travelers. We join the ADDRESS, ROOM, RESERVATION, TRAVELER, EMPLOYEE, DEPARTMENT, and COMPANY tables to get all the pertinent details of the reservation information of a given company. These details include company name, department name, reservation date, and reservation ID. Then we use the Where clause to choose the reservation start date and end date. Next, by using the Group By statement with aggregation function Count, we can obtain the total number of reservations per city. Finally, by using the function Rownum, we select the top 5 most traveled to cities.

**Output:**

|  |  |  |
| --- | --- | --- |
| **Company\_Name** | **Department\_Name,** | **Partial\_No\_Reservation** |
| SAP | Los Angeles | 3 |
| SAP | Costa Mesa | 2 |
| SAP | Santa Ana | 2 |
| SAP | Pleasanton | 2 |
| SAP | Palo Alto | 1 |

**SQL Script:**

Select \*

from

(

Select addrs\_city City, co\_name Company\_Name, count(reserv\_ID) Partial\_No\_Reservation

From ADDRESS, ROOM, RESERVATION, TRAVELER, EMPLOYEE, DEPARTMENT, COMPANY

Where ADDRESS.addrs\_ID = ROOM.addrs\_ID

And ROOM.room\_ID = RESERVATION.room\_ID

And RESERVATION.traveler\_ID = TRAVELER.traveler\_ID

And TRAVELER.traveler\_ID = EMPLOYEE.emp\_ID

And EMPLOYEE.dept\_ID = DEPARTMENT.dept\_ID

And DEPARTMENT.co\_ID = COMPANY.CO\_ID

And co\_name = 'SAP'

And reserv\_start >= '01-JAN-15'

And reserv\_end <= '01-JAN-16'

AND reserv\_ID not in (Select reserv\_ID from CANCELLATION)

Group By addrs\_city, co\_name

Order By count(reserv\_ID) desc

)

where rownum <= 5;

## Query 5**:** Co.Host incentives paid to each user over given time periods

**Explanation:**

Our client, Co.Host, wants to know about the incentives it has paid to each user over a given time period. To find this information we query the reward data from the REWARD\_LOG table. By joining the EMPLOYEE table and REWARD\_LOG able, we can get all the reward details of each employee. The details we are interested in for this query are the employee ID, employee name, and reward amount. Then we use the Where clause to limit the reward date. Next, by using the Group By statement with the aggregation function Sum, we can obtain the total amount of reward for each user, a.k.a. employee over the given time period.

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee\_ID** | **First\_Name** | **Last\_Name** | **Partial\_Incentive\_Paid** |
| EMP000000002 | Richard | Fox | 1200 |
| EMP000000004 | Lawrence | Medina | 1600 |
| EMP000000007 | Alan | Lee | 1600 |
| EMP000000008 | Janet | Grant | 1200 |
| EMP000000009 | Raymond | Myers | 1400 |
| EMP000000010 | Daniel | Ellis | 800 |
| EMP000000011 | Nicholas | Smith | 2100 |
| EMP000000012 | Andrew | Williams | 3200 |
| EMP000000014 | Christopher | Cunningham | 1200 |
| EMP000000015 | James | Walker | 400 |
| … |  |  |  |

**SQL Script:**

Select EMPLOYEE.emp\_ID Employee\_ID, emp\_first First\_Name , emp\_last Last\_Name,

sum(reward\_amt) Partial\_Incentive\_Paid

From REWARD\_LOG, EMPLOYEE

Where EMPLOYEE.emp\_ID = REWARD\_LOG.emp\_ID

And reward\_Date Between '01-JAN-15' And '31-DEC-15'

Group By EMPLOYEE.emp\_ID ,emp\_first, emp\_last

Order By EMPLOYEE.emp\_ID;

## Query 6**:** Current awards levels of individual users

**Explanation:**

Our client, Co.Host, wants to know current awards levels of individual users, so we query this data based on the reward data from table REWARD\_LOG. By joining the Table EMPLOYEE and REWARD\_LOG, we can get all of the details of reward info of every employee, such as employee ID, employee name, email, and reward amount. Then, by using the Group By statement with aggregation function Sum, we can obtain the total amount of reward per each user, a.k.a. employee.

**Output:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Employee\_ID** | **First\_Name** | **Last\_Name** | **Email** | **Reward\_Type** | **Reward\_Name** | **Current\_Reward\_Level** | **Partial\_Incentive\_Paid** |
| EMP000000016 | Brandon | Shaw | bshaw@intuit.com | Points | Best Western Rewards | Gold | 600 |
| EMP000000017 | Jean | White | jwhite@intuit.com | Points | Wyndham Rewards | Silver | 1900 |
| EMP000000019 | Scott | Anderson | sanderson@intuit.com | Points | Best Western Rewards | Platinum | 1200 |
| EMP000000020 | Ethan | Mitchell | emitchell@intuit.com | Miles | Southwest Rapid Rewards | Gold | 800 |
| EMP000000021 | Nicole | Peters | npeters@sap.com | Points | Club Carlson | Gold | 800 |
| EMP000000022 | Paul | Young | pyoung@sap.com | Cash | Cash | Platinum | 2400 |
| EMP000000023 | Frank | Bennett | fbennett@sap.com | Miles | HawaiianMiles | Gold | 1600 |
| EMP000000025 | Andrew | Peterson | apeterson@sap.com | Points | Wyndham Rewards | Platinum | 1200 |
| EMP000000026 | Samuel | Johnson | sjohnson@sap.com | Miles | Southwest Rapid Rewards | Silver | 1600 |
| EMP000000027 | Anthony | Harrison | aharrison@sap.com | Points | Best Western Rewards | Gold | 1600 |
| … |  |  |  |  |  |  |  |

**SQL Script:**

Select EMPLOYEE.emp\_ID EMPLOYEE\_ID, emp\_first First\_Name ,emp\_last Last\_Name ,email,

reward\_type, reward\_name, current\_rew\_lvl,

sum(reward\_amt) TOTAL\_REWARD\_AMOUNT

From EMPLOYEE, REWARD\_LOG , PAYMENT, REWARD\_SYSTEM

Where EMPLOYEE.emp\_ID = REWARD\_LOG.emp\_ID

And PAYMENT.pmnt\_ID = REWARD\_LOG.pmnt\_ID

And REWARD\_SYSTEM.reward\_ID = EMPLOYEE.reward\_ID

Group by EMPLOYEE.emp\_ID, emp\_first, emp\_last, email , reward\_type, reward\_name, current\_rew\_lvl

Order by EMPLOYEE.emp\_ID;

## Query 7**:** Top 10 booked rooms and addresses

**Explanation:**

Our client, Co.Host, wants to know the most booked rooms and addresses across all reservations. We find this data based on the reservation data from the RESERVATION table, excluding data found in the CANCELLATION table. We join ADDRESS, ROOM, and RESERVATION tables to get the necessary details including room ID, address, reservation date, and reservation ID. Then we use the Where clause to filter the reservation start date and end dates. Next using the Group By statement with the aggregation function Count, we can obtain the total number of reservations per room. In the end, by using the function Rownum, we select the top 10 rooms by traveler stays.

**Output:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Room\_ID** | **Zip\_Code** | **Street** | **City** | **State** | **Country** | **Partial\_No\_Reservation** |
| ROO000000015 | 90049 | 11982 Mayfield Ave | Los Angeles | CA | USA | 3 |
| ROO000000023 | 92014 | 377 13th St | Del Mar | CA | USA | 3 |
| ROO000000016 | 92626 | 3082 Molokai Pl | Costa Mesa | CA | USA | 2 |
| ROO000000020 | 94588 | 2163 Alexander Way | Pleasanton | CA | USA | 2 |
| ROO000000013 | 94301 | 2302 Webster St | Palo Alto | CA | USA | 2 |
| ROO000000021 | 95746 | 4460 Cavitt Stallman Rd | Granite Bay | CA | USA | 2 |
| ROO000000017 | 92707 | 431 Curie Ave | Santa Ana | CA | USA | 2 |
| ROO000000022 | 94043 | 2452 Elka Ave | Mountain View | CA | USA | 2 |
| ROO000000011 | 92129 | 13191 Carolee Ave | San Diego | CA | USA | 1 |
| ROO000000012 | 92129 | 7665 Via Cristal | San Diego | CA | USA | 1 |
|  |  |  |  |  |  |  |

**SQL Script:**

Select \*

from

(

Select ROOM.room\_ID ROOM\_ID,

addrs\_zip Zip\_Code, addrs\_street Street,

addrs\_city City, addrs\_state State, addrs\_country Country,

count(reserv\_ID) Partial\_No\_Reservation

From ADDRESS, ROOM, RESERVATION

Where ADDRESS.addrs\_ID = ROOM.addrs\_ID

And ROOM.room\_ID = RESERVATION.room\_ID

And reserv\_start >= '01-JAN-15'

And reserv\_end <= '01-JAN-16'

AND reserv\_ID not in(Select reserv\_ID from CANCELLATION)

Group by ROOM.room\_ID,addrs\_zip,addrs\_street, addrs\_city, addrs\_state, addrs\_country

Order by count(reserv\_ID) desc

)

where rownum <= 10;

## Query 8: Rooms and addresses that have never been booked over time periods

**Explanation:**

Our client, Co.Host, wants to know Rooms and addresses that have never been booked over a given time period. We query this data based on the room data from ROOM table, only including rooms which have never been booked. We eliminate booked rooms by using a sub query to obtain the room IDs which exist in the RESERVATION table over the given time periods. The result is a list of rooms that never existed in table RESERVATION table, or in other words a list of rooms that have never been booked. We conclude by joining the limited ROOM table to ADDRESS. The result of this query provides all the details of about un-booked rooms that we need to display including room ID and address.

**Output:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Room\_ID** | **Zip\_Code** | **Street** | **City** | **State** | **Country** |
| ROO000000002 | 94103 | 523 Natoma St | San Francisco | CA | USA |
| ROO000000004 | 94117 | 904 Steiner St | San Francisco | CA | USA |
| ROO000000006 | 94110 | 235 Andover St | San Francisco | CA | USA |
| ROO000000007 | 94110 | 3763 Folsom St | San Francisco | CA | USA |
| ROO000000008 | 92126 | 11468 Vela Dr | San Diego | CA | USA |
| ROO000000014 | 94025 | 2077 Valparaiso Ave | Menlo Park | CA | USA |
| ROO000000019 | 94301 | 2200 Byron St | Palo Alto | CA | USA |
| ROO000000024 | 92014 | 377 13th St | Del Mar | CA | USA |

**SQL Script:**

Select room\_ID,

addrs\_zip Zip\_Code, addrs\_street Street,

addrs\_city City, addrs\_state State, addrs\_country Country

from ROOM, ADDRESS

where

ROOM.addrs\_ID = ADDRESS.addrs\_ID

AND

room\_ID Not in

(

Select ROOM.room\_ID

From ADDRESS, ROOM, RESERVATION

Where ADDRESS.addrs\_ID = ROOM.addrs\_ID

And ROOM.room\_ID = RESERVATION.room\_ID

And reserv\_start >= '01-JAN-15'

And reserv\_end <= '01-JAN-16'

AND reserv\_ID not in (Select reserv\_ID from CANCELLATION)

)

order by room\_ID;

## Query 9**:** Number of bookings for every city over a given period of time

**Explanation:**

Our client, Co.Host, wants to know number of bookings for every city over a given period of time. We query this data based on the reservation data found in the RESERVATION table, excluding the data which exists in the CANCELLATION table. By joining the Table ADDRESS, ROOM, and RESERVATION, we get the pertinent details of reservation information for all cities, including city and reservation ID. We conclude by using the Group By statement with the aggregation function Count, which allows us to obtain the total number of reservations per city.

**Output:**

|  |  |
| --- | --- |
| **City** | **Partial\_No\_Reservation** |
| Costa Mesa | 2 |
| Del Mar | 3 |
| Granite Bay | 2 |
| Los Angeles | 3 |
| Mountain View | 2 |
| Palo Alto | 3 |
| Pleasanton | 2 |
| San Diego | 4 |
| San Francisco | 3 |
| Santa Ana | 2 |

**SQL Script:**

Select addrs\_city CITY, count(reserv\_ID) Partial\_No\_Reservation

From ADDRESS, ROOM , RESERVATION

Where ADDRESS.addrs\_ID = ROOM.addrs\_ID And ROOM.room\_ID = RESERVATION.room\_ID

And reserv\_start >= '01-JAN-15'

And reserv\_end <= '01-JAN-16'

AND reserv\_ID not in(Select reserv\_ID from CANCELLATION)

Group By addrs\_city

Order By addrs\_city;

## Query 10: How often do employees stay together ?

**Explanation:**

Our client, Co.host has main value proposition for companies that their business model will help employees to build a valuable relationship with each other. To help validate their business value proposition, the client has requested a report that shows the number of times employees stay with each other. Having that count will help them analyze stay trends and take appropriate business actions.

If a traveler stays with a host that is one unique group. If that host then becomes a traveler and stays with the former traveler, for our purposes that is the same group. We are interested in finding the number of unique times this group occurs. To simplify this statement we want to count the number of times the group (A,B) or (B,A) exists as the same group.

To achieve this we join the RESERVATION, ROOM, ADDRESS, HOST, and TRAVELER tables. In one set we select “A” as all travelers where the last 4 digits of the TRAVELER\_ID are greater than the last 4 digits of the HOST\_ID that the traveler stayed with. The HOST\_IDs identified in this process are then selected as “B”. We employ this method as a way of splitting the (A,B) and (B,A) sets into groups that do not overlap so that we can combine them later on.

We then inverse this process and take all HOST\_IDs as “A”, from the same set of tables, where the last 4 digits of the HOST\_ID are greater than the last 4 digits of the TRAVELER\_ID that stayed with that host. The TRAVELER\_IDs identified in this process are then selected as “B”.

We validate that A has stayed with B and that B has stayed with A

We then use UNION ALL to combine the data sets and employ the GROUP BY clause and the COUNT aggregation function to count each unique occurrence of any group.

**Output:**

|  |  |  |
| --- | --- | --- |
| **Shared\_Stay\_Count** | **Employee 1** | **Employee 2** |
| 3 | EMP000000040 | EMP000000038 |
| 2 | EMP000000012 | EMP000000007 |

SQL Script:

select count (\*) as "Shared stay count", final."A" as "Employee 1", final."B" as "Employee 2" from

(select resr.traveler\_ID as "A", A.host\_ID as "B" from reservation resr ,address A, room R, (select distinct AA.traveler\_ID as "T1" ,AA.host\_ID as "H1" from

(select resr.traveler\_ID, A.host\_ID from reservation resr ,address A, room R

where R.addrs\_ID=A.addrs\_ID

and R.room\_ID= resr.room\_ID ) AA,

(select resr.traveler\_ID, A.host\_ID from reservation resr ,address A, room R

where R.addrs\_ID=A.addrs\_ID

and R.room\_ID= resr.room\_ID ) BB

where AA.host\_ID=BB.traveler\_ID and BB.host\_ID=AA.traveler\_ID and

SUBSTR(AA.traveler\_ID, 4, 12) > SUBSTR(AA.host\_ID,4, 12)) grp2

where R.addrs\_ID=A.addrs\_ID

and R.room\_ID= resr.room\_ID

and grp2."T1"=resr.traveler\_ID

and grp2."H1"=A.host\_ID

Union All

select A.host\_ID as "A", resr.traveler\_ID as "B" from reservation resr ,address A, room R, (select distinct AA.traveler\_ID as "T1" ,AA.host\_ID as "H1" from

(select resr.traveler\_ID, A.host\_ID from reservation resr ,address A, room R

where R.addrs\_ID=A.addrs\_ID

and R.room\_ID= resr.room\_ID ) AA,

(select resr.traveler\_ID, A.host\_ID from reservation resr ,address A, room R

where R.addrs\_ID=A.addrs\_ID

and R.room\_ID= resr.room\_ID ) BB

where AA.host\_ID=BB.traveler\_ID and BB.host\_ID=AA.traveler\_ID and

SUBSTR(AA.traveler\_ID, 4, 12) < SUBSTR(AA.host\_ID,4, 12)) grp1

where R.addrs\_ID=A.addrs\_ID

and R.room\_ID= resr.room\_ID

and grp1."T1"=resr.traveler\_ID

and grp1."H1"=A.host\_ID) final

GROUP by final."A", final."B"

# Chapter 5: Triggers and Procedures

## Trigger 1: Cancellation

The purpose of this trigger is to allow travelers to cancel a reservation.

The trigger takes two input values, the reservation ID and the reason for cancellation. It will use the current date and time as the cancellation date, which is also used to determine if a cancellation has been made in time to receive a refund. Our client Co.Host allows refunds for cancellations made at least 24 hours before the start of the reservation. A reservation can still be canceled within 24 hours of the reservation start date, but no refund will be given. The trigger looks in the RESERVATION table to determine the reservation price in order to refund the correct amount of money. Because the company spend and department spend amounts are updated in the COMPANY and DEPARTMENT tables, respectively, when a payment is made, any money refunded due to a cancellation needs to update those fields as well. This trigger performs that task. Finally, the trigger updates the Is\_Canc (cancellation flag) field in the RESERVATION table to indicate that the reservation was canceled. If a reservation has already been canceled, the trigger will stop working without duplicating the cancellation. This situation should not come up, but error handling was included just in case.

CREATE OR REPLACE TRIGGER TRIG\_CAN\_SEQ BEFORE

INSERT ON CANCELLATION FOR EACH ROW DECLARE TEMP\_CAN\_NOCANCELLATION.CANC\_ID%TYPE ;

TEMP\_CANC\_LIMIT CANCELLATION.CANC\_LIMIT%TYPE;

TEMP\_REF\_AMT CANCELLATION.REF\_AMT%TYPE;

TEMP\_RES\_START RESERVATION.RESERV\_START%TYPE;

TEMP\_IS\_CANC RESERVATION.ISCANC%TYPE;

TEMP\_CANC\_DATE CANCELLATION.CANC\_DATE%TYPE;

temp\_tot\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

temp\_tot\_co\_spend COMPANY.total\_co\_spend%type ;

TEMP\_DEPT\_ID DEPARTMENT.DEPT\_ID%TYPE;

TEMP\_CO\_ID COMPANY.CO\_ID%TYPE;

BEGIN

--Insert the cancellation number

SELECT 'CAN'

|| LPAD( TO\_CHAR(CANCELLATION\_CANC\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_CAN\_NO

FROM DUAL;

:new.CANC\_ID := TEMP\_CAN\_NO ;

--Handle duplicate cancellation errors

SELECT ISCANC

INTO TEMP\_IS\_CANC

FROM RESERVATION

WHERE RESERV\_ID = :new.reserv\_ID;

IF (TEMP\_IS\_CANC = 'Y') THEN

raise\_application\_error(-20404, 'This reservation has already been cancelled.');

ELSE

TEMP\_CANC\_DATE := SYSDATE; --assign the canc\_date

TEMP\_IS\_CANC := 'Y'; --mark the reservation as cancelled

SELECT RESERV\_START

INTO TEMP\_RES\_START

FROM RESERVATION

WHERE RESERV\_ID = :new.RESERV\_ID;

TEMP\_CANC\_LIMIT := TEMP\_RES\_START - 1; --determine the last refundable canc\_date

IF (SYSDATE <= TEMP\_CANC\_LIMIT) THEN

SELECT PMNT\_PRICE

INTO TEMP\_REF\_AMT

FROM RESERVATION r

JOIN PAYMENT p

ON r.PMNT\_ID = p.PMNT\_ID

WHERE RESERV\_ID = :new.RESERV\_ID; --determine ref amt for timely canc

ELSE

TEMP\_REF\_AMT := 0; --no refund for late cancellation

END IF;

--insert values for canc\_limit, ref\_amt, and canc\_date in the CANCELLATION table

:new.CANC\_LIMIT := TEMP\_CANC\_LIMIT;

:new.REF\_AMT := TEMP\_REF\_AMT;

:new.CANC\_DATE := TEMP\_CANC\_DATE;

UPDATE RESERVATION

SET isCanc = TEMP\_IS\_CANC --update cancellation state

WHERE RESERV\_ID = :new.RESERV\_ID; --in RESERVATION table

--update the total department spend to subtract refund

SELECT tot\_dept\_spend,

r.DEPT\_ID

INTO temp\_tot\_dept\_spend,

TEMP\_DEPT\_ID

FROM RESERVATION r

JOIN DEPARTMENT d

ON r.DEPT\_ID = d.DEPT\_ID

WHERE r.RESERV\_ID = :new.reserv\_ID;

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend - TEMP\_REF\_AMT;

UPDATE DEPARTMENT

SET tot\_dept\_spend = temp\_tot\_dept\_spend

WHERE dept\_ID = TEMP\_DEPT\_ID;

--update total company spend to subtract refund

SELECT total\_co\_spend,

c.CO\_ID

INTO temp\_tot\_co\_spend,

TEMP\_CO\_ID

FROM RESERVATION r

JOIN DEPARTMENT d

ON r.DEPT\_ID = d.DEPT\_ID

JOIN COMPANY c

ON d.CO\_ID = c.CO\_ID

WHERE r.RESERV\_ID = :new.reserv\_ID;

temp\_tot\_co\_spend := temp\_tot\_co\_spend - TEMP\_REF\_AMT;

UPDATE COMPANY

SET total\_co\_spend = temp\_tot\_co\_spend

WHERE co\_ID = temp\_co\_ID ;

END IF;

END ;

/

## Trigger 2: Department And Company Spend

The purpose of this trigger is to automatically update the total amount each Department and Company has spent after a reservation is made.

The trigger is written to work with new reservations being made and cancellations when they happen. When a reservation is made there is an insert on the RESERVATION table and when a cancellation is made there is an update is made to the RESERVATION table. Therefore the “Department And Company Spend” trigger is written to fire on both an insert or update to the RESERVATION table.

If updating we take the old amounts and change them based on the updated amount.

If inserting we increment the company and department spend amounts the new inserted amount.

**SQL Script**

-------------------------------Trigger -------------------------------

--14—- Automatically Update Totol\_Department\_Spend,Total\_Company\_Spend,

--After Insert or Update of pmnt\_price on PAYMENT

-- Event:

-- Trigger:

Create or Replace Trigger Trig\_Dept\_Company\_Spend Before Insert or

Update Of pmnt\_price,dept\_ID on PAYMENT

For Each Row

Declare

temp\_tot\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

temp\_tot\_co\_spend COMPANY.total\_co\_spend%type ;

temp\_pmnt\_price PAYMENT.pmnt\_price%type;

temp\_co\_ID COMPANY.co\_ID%type;

temp\_tot\_old\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

temp\_tot\_new\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

Begin

/\*

If Updating('pmnt\_ID') Then

If Updating ('dept\_ID') Then

Select tot\_dept\_spend into temp\_tot\_old\_dept\_spend

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select tot\_dept\_spend into temp\_tot\_new\_dept\_spend

From DEPARTMENT Where dept\_ID = : new.dept\_ID ;

Select co\_ID into temp\_co\_ID

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select total\_co\_spend into temp\_tot\_co\_spend

From COMPANY Where co\_ID = temp\_co\_ID ;

-- Update total departmental spend

temp\_tot\_old\_dept\_spend := temp\_tot\_old\_dept\_spend + : old.pmnt\_price;

temp\_tot\_new\_dept\_spend := temp\_tot\_new\_dept\_spend - : new.pmnt\_price;

-- Update total company's spend

temp\_tot\_co\_spend := temp\_tot\_co\_spend - : old.pmnt\_price + : new.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_old\_dept\_spend

Where dept\_ID = : old.dept\_ID;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_new\_dept\_spend

Where dept\_ID = : new.dept\_ID;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID;

End If;

\*/

If (Inserting)Then

Select tot\_dept\_spend,co\_ID into temp\_tot\_dept\_spend, temp\_co\_ID

From DEPARTMENT Where dept\_ID = : new.dept\_ID ;

Select total\_co\_spend into temp\_tot\_co\_spend

From COMPANY Where co\_ID = temp\_co\_ID ;

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend + : new.pmnt\_price;

temp\_tot\_co\_spend := temp\_tot\_co\_spend + : new.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_dept\_spend

Where dept\_ID = : new.dept\_ID;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID ;

End IF;

If (Updating ('pmnt\_price')) Then

Select tot\_dept\_spend, co\_ID into temp\_tot\_dept\_spend, temp\_co\_ID

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select total\_co\_spend into temp\_tot\_co\_spend

From COMPANY Where co\_ID = temp\_co\_ID ;

-- Update total departmental spend

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend - : old.pmnt\_price;

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend + : new.pmnt\_price;

-- Update total company's spend

temp\_tot\_co\_spend := temp\_tot\_co\_spend - : old.pmnt\_price;

temp\_tot\_co\_spend := temp\_tot\_co\_spend + : new.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_dept\_spend

Where dept\_ID = : old.dept\_ID;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID;

End If;

If (Updating ('dept\_ID')) Then

Select tot\_dept\_spend into temp\_tot\_old\_dept\_spend

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select tot\_dept\_spend into temp\_tot\_new\_dept\_spend

From DEPARTMENT Where dept\_ID = : new.dept\_ID ;

-- Update total departmental spend

temp\_tot\_old\_dept\_spend := temp\_tot\_old\_dept\_spend - : old.pmnt\_price;

temp\_tot\_new\_dept\_spend := temp\_tot\_new\_dept\_spend + : old.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_old\_dept\_spend

Where dept\_ID = : old.dept\_ID;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_new\_dept\_spend

Where dept\_ID = : new.dept\_ID;

End If;

End;

/

commit;

-----------------------------Test The Trigger-------------------------------

/\*

-- Test Isert --

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

Insert into PAYMENT(DEPT\_ID,PMNT\_DATE, PMNT\_PRICE) Values ('DEP000007', '01-JAN-16', 1000);

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

--Update DEPARTMENT set tot\_dept\_spend = 10775 where dept\_ID = 'DEP000009';

-- Test Update Payment Price --

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Update PAYMENT Set PMNT\_PRICE = 2000 Where DEPT\_ID = 'DEP000007' And PMNT\_DATE ='01-JAN-16';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

--Update COMPANY set total\_co\_spend = 15337.8 where co\_ID = 'CO\_00001';

--Update DEPARTMENT set tot\_dept\_spend = 8775 where dept\_ID = 'DEP000009';

Rollback;

-- Test Update Department ID --

Select \* from PAYMENT Where DEPT\_ID = 'DEP000009' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000009';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Update PAYMENT Set DEPT\_ID = 'DEP000009' Where PMNT\_ID ='PMN000000059' ;

Select \* from PAYMENT Where DEPT\_ID = 'DEP000009' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000009';

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000010';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

Rollback;

-- Test Update Department ID & Payment Amount--

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007'or DEPT\_ID = 'DEP000009' or DEPT\_ID= 'DEP000010' order by dept\_ID ;

Update PAYMENT Set DEPT\_ID = 'DEP000010', PMNT\_PRICE = 3000 Where PMNT\_ID ='PMN000000059' ;

Select \* from PAYMENT Where DEPT\_ID = 'DEP000010' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000010';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000009' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000009';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

rollback;

\*/

## Procedure 1: Make Payment

The “Make Payment” procedure is used when a reservation is confirmed on the website to update the ROOM\_PRICE and PMNT\_ID field in the RESERVATION table, and insert a new row in the PAYMENT table.

To accomplish this the procedure takes the RESERVATION\_ID of the reservation being made and uses the TRAVELER\_ID, ROOM\_ID, OFF\_ID, DEPT\_ID, RESERV\_START, RESERV\_END to calculate the ROOM\_PRICE by looping trough all days in the reservation timeframe and calculating the total amount of the reservation.

We employ the ADD\_MONTHS function to find the RATE\_START\_DATE in the REGION\_RATE table to find the rate that is current for the date that is being looped through. We add each iterative rate together in the loop and use the result to populate the PMNT\_PRICE in the PAYMENT table and insert a new row into the payment table. Then we use the SEQUENCE id that was just generated and the CURRVAL function to populate in the PMNT\_ID in the RESERVATION table and update the ROOM\_PRICE in the RESERVATION table to be the same as was entered in the PAYMENT table.

Exceptions are handled with custom exception handing to tell the user that the reservation they have entered an invalid reservation ID.

--15—- Insert the Payment based on the Reservation (after execute the procedure)

-- Procedure

-- Input Parameter: reservation ID, reservation start date, reservatin end date

-- Output Parameter: Table: REVENUE\_PER\_CITY

Create or Replace Procedure Make\_Payment(

reservation\_ID RESERVATION.reserv\_ID%type )

AS

--/\*

temp\_start\_date RESERVATION.reserv\_start%type;

temp\_end\_date RESERVATION.reserv\_end%type;

temp\_current\_date RESERVATION.reserv\_start%type;

temp\_dept\_ID DEPARTMENT.dept\_ID%type ;

temp\_off\_ID OFFICE.off\_ID%type;

temp\_reg\_ID REGION.reg\_ID%type;

temp\_room\_ID ROOM.room\_ID%type;

temp\_pmnt\_date PAYMENT.pmnt\_date%type ;

temp\_pmnt\_ID PAYMENT.pmnt\_ID%type ;

temp\_pmnt\_price PAYMENT.pmnt\_price%type ;

temp\_rate\_begin\_date REGION\_RATE.rate\_begin\_date%type;

temp\_region\_rate REGION\_RATE.region\_rate%type;

temp\_room\_price RESERVATION.room\_price%type;

total\_room\_price RESERVATION.room\_price%type;

Begin

-- Delete old pmnt\_ID, if pmnt\_ID is not null

Select pmnt\_ID Into temp\_pmnt\_ID From RESERVATION Where reserv\_ID = reservation\_ID;

If (temp\_pmnt\_ID is not null) Then

Update RESERVATION Set pmnt\_ID = null Where reserv\_ID = reservation\_ID;

Delete From PAYMENT Where pmnt\_ID = temp\_pmnt\_ID;

End If;

-- Set up Initial Values

Select reserv\_start, reserv\_end ,

dept\_ID, off\_ID,room\_ID

Into temp\_start\_date, temp\_end\_date ,

temp\_dept\_ID, temp\_off\_ID, temp\_room\_ID

From RESERVATION

Where reserv\_ID = reservation\_ID;

-- Query the region ID based onthe off\_ID so room rate can be calculated.

Select reg\_ID into temp\_reg\_ID From OFFICE Where off\_ID = temp\_off\_ID;

-- Payment date is the system when execute this procedure

Select sysdate Into temp\_pmnt\_date From DUAL;

temp\_current\_date := temp\_start\_date;

total\_room\_price := 0;

-- Calculate the payment price based on reservation

WHILE (temp\_current\_date < temp\_end\_date) Loop

SELECT RR.REGION\_RATE AS RATE\_CHARGED INTO temp\_region\_rate FROM

REGION\_RATE RR

INNER JOIN REGION RE

ON RR.REG\_ID = RE.REG\_ID

INNER JOIN ADDRESS AA

ON TRIM(LOWER(AA.ADDRS\_COUNTY)) = TRIM(LOWER(RE.COUNTY))

INNER JOIN ROOM RO

ON AA.ADDRS\_ID = RO.ADDRS\_ID

WHERE room\_ID = temp\_room\_ID

And temp\_current\_date >= rate\_begin\_date

and temp\_current\_date <= (ADD\_MONTHS(rate\_begin\_date ,1)-1);

temp\_room\_price := temp\_region\_rate \* 0.9;

total\_room\_price := temp\_room\_price + total\_room\_price;

temp\_current\_date := temp\_current\_date +1;

End Loop;

temp\_pmnt\_price := total\_room\_price;

--Insert date into Table PAYMENT

INSERT INTO PAYMENT(dept\_ID,pmnt\_date, pmnt\_price)

VALUES (temp\_dept\_ID, temp\_pmnt\_date,temp\_pmnt\_price);

-- Update total room price into room price based on reservation.

Update RESERVATION Set room\_price = total\_room\_price Where reserv\_ID = reservation\_ID;

--Update foreign key value, current payment ID into Table RESERVATION

SELECT 'PMN'

|| LPAD( TO\_CHAR(PAYMENT\_PMNT\_ID\_SEQ.currval ),9,'0') into temp\_pmnt\_ID From Dual;

UPDATE RESERVATION SET pmnt\_ID = temp\_pmnt\_ID WHERE reserv\_ID = reservation\_ID;

COMMIT;

EXCEPTION

WHEN no\_data\_found THEN

raise\_application\_error (-20001, reservation\_ID||'does not exist ');

END;

/

-----------------------------Test The Procedure-------------------------------

--/\*

--Test 1 , Insert a new date into Table RAESERVATION and check the Table PAYMENT and RESERVATION

Select \* from RESERVATION where traveler\_ID = 'EMP000000004';

Insert Into RESERVATION (traveler\_ID,room\_ID,off\_ID,dept\_ID,reserv\_start,reserv\_end, iscanc)

Values ('EMP000000004','ROO000000001','OFF00002','DEP000009','10-DEC-15', '14-JAN-16', 'N');

Select \* From RESERVATION Where traveler\_ID ='EMP000000004';

Execute Make\_Payment('RES000000041');

Select \* From PAYMENT order by PMNT\_ID desc;

Select \* From RESERVATION Where traveler\_ID ='EMP000000004';

Rollback;

-- Test 2 , Insert a new date into Table RESERVATION and check the Table PAYMENT and RESERVATION

Select \* from RESERVATION;

Insert Into RESERVATION (traveler\_ID,room\_ID,off\_ID,dept\_ID,reserv\_start,reserv\_end, iscanc)

Values ('EMP000000004','ROO000000013','OFF00003','DEP000009','17-OCT-15', '14-DEC-15', 'N');

Select \* From RESERVATION Where traveler\_ID ='EMP000000004';

Execute Make\_Payment('RES000000041');

Select \* From PAYMENT Order by PMNT\_ID desc;

Select \* From RESERVATION Where traveler\_ID ='EMP000000004' Order By reserv\_ID Desc;

## Procedure 2 Revenue Per Period

This procedure calculates the revenue based on cities for a given time period. The user can select the time period they are interested in by entering the start date and the end date. The procedure must join several tables to relate the city in which addresses are located to the reservations made for rooms in those addresses. It must also determine if any refunds have been given for canceled reservations. It uses one curser to loop through all reservations for the time period and sum up the total reservation amounts while subtracting any refund amounts. It uses a second curser to loop through all of the distinct cities in which host addresses are located to return a table that contains totals for each of those cities.

--2--Co.Host revenues per city over given time periods

-- Input Parameter: city, start\_date, end\_date

-- Output: Insert into Table REVENUE\_PER\_CITY

-- Procedure Format: Revenue\_Per\_City\_For\_Period ('San Diego','01-JAN-15', '01-FEB-15')

-- Excute the Procedure: Execute Revenue\_Per\_City\_For\_Period ('San Diego','01-JAN-15', '01-FEB-15') ;

Create or Replace Procedure Revenue\_For\_Period

(

start\_date PAYMENT.pmnt\_date%type,

end\_date PAYMENT.pmnt\_date%type

)

--Return PAYMENT.pmnt\_price%type

IS

-- Declare the Variable

Cursor C1 is Select a.addrs\_city, p.pmnt\_date, p.pmnt\_price, c.ref\_amt

From ADDRESS a JOIN ROOM r ON a.addrs\_ID = r.addrs\_ID

JOIN RESERVATION res ON r.room\_ID = res.room\_ID

JOIN PAYMENT p ON res.pmnt\_ID = p.pmnt\_ID

LEFT OUTER JOIN CANCELLATION c ON res.reserv\_ID = c.reserv\_ID

Where pmnt\_date >= start\_date and pmnt\_date <= end\_date

;

Cursor C2 is Select distinct addrs\_city from ADDRESS;

temp\_city ADDRESS.addrs\_city%type;

temp\_revenue PAYMENT.pmnt\_price%type;

total\_revenue PAYMENT.pmnt\_price%type;

Begin

Delete From REVENUE\_PER\_CITY;

For C2\_row in C2 Loop

temp\_revenue:= 0;

total\_revenue:=0;

temp\_city := C2\_row.addrs\_city;

For C1\_row in C1 Loop

If (C1\_row.addrs\_city = temp\_city) Then

temp\_revenue := C1\_row.pmnt\_price - Coalesce(C1\_row.ref\_amt, 0);

total\_revenue := temp\_revenue + total\_revenue ;

End If;

End Loop;

Insert into REVENUE\_PER\_CITY (City,StartDate, EndDate,TotalRevenue)

values(temp\_city, start\_date, end\_date, total\_revenue);

End Loop;

End;

/

--commit;

-----------------------------Test The Procedure-------------------------------

/\*

Execute Revenue\_For\_Period('01-Jan-15','01-JAN-16') ;

Select \* from REVENUE\_PER\_CITY ;

Execute Revenue\_For\_Period( '01-Jan-15','01-SEP-16') ;

Select \* from REVENUE\_PER\_CITY ;

Execute Revenue\_For\_Period('21-May-15','30-SEP-15') ;

Select \* from REVENUE\_PER\_CITY ;

\*/

## Procedure 3: Available Rooms

This procedure is designed to find all available rooms within our database where the active Traveler's company matches the available Hosts' company, the room is in the same region as the destination region, and where the Traveler is not also the Host.

To accomplish this we take ROOM and use MINUS to remove all rooms where PREFFERED\_START date and the PREFFERED\_END date overlap the dates that have been reserved for this room in the RESERVATION table.

Then we get the host information for each available room by joining the ADDRESS, HOST, EMPLOYEE, and DEPARTMENTS tables and further limit the results to only those where the TRAVELER is not also the HOST.

Then we validate that the results are in the preferred county and preferred state. This is to be sure that we do not join to duplicate counties across many states.

Finally we join the traveler, employee, and department tables where the TRAVLER\_ID in the TRAVELER table is the same as the active traveler and limit the results to only those where the traveler and host are in the same company.

We take these results and load them into a **cursor** and use the cursor loop variable to loop through the cursor and load the results in to the table AVAILABLE\_ROOM. The results are effective dated and include the active TRAVELER\_ID to allow the front end to quickly identify pertinent results by choosing all records where the TRAVELER\_ID in AVAILABLE\_ROOM matches the active TRAVELER\_ID and where the AVAIL\_CHECK\_DATE is the maximum from that group.

The AVAILABLE\_ROOM table is intended to hold a history of user queries until such time as it is truncated so that CoHost can analyze user preferences in their searches.

There is a toggle written in to the procedure that will allow the user to activate a delete of the AVAILABLE\_ROOM table each time the procedure is run but this does not actively support more than one user at a time.

Parameters:

TRAV\_CHECKING The traveler Id of the person looking for a room.

OFFICE\_COUNTY The county that the traveler is trying to travel to.

OFFICE\_STATE The state that the traveler is trying to travel to.

Necessary to differentiate duplicate county names.

PREFERRED\_START The prefferred start date the traveler desires.

PREFFERED\_END The prefferred end date that the traveler desires.

Exception Handling:

There are 2 exceptions handled by this procedure.

1. If the user has entered a date before today's date they are prompted to enter a valid date.

2. If any other exception happens the ORA-20005 exception is thrown and the user is given a brief message explaining that no rooms were found for these dates.

Output: This procedure appends a ROOM\_ID and many interesting attributes related to that room including all tracked amenities. The output also includes the TRAVELER\_ID of the active traveler, the dates queried, the date the procedure was run, and the room rate for each room returned over the dates queried.

Future improvements:

Exception reporting could be improved to identify further errors and exceptions that occur at runtime.

**SQL Script**

CREATE OR REPLACE PROCEDURE AVAILABLE\_ROOMS(

TRAV\_CHECKING TRAVELER.TRAVELER\_ID%TYPE,

OFFICE\_COUNTY ADDRESS.ADDRS\_COUNTY%TYPE,

OFFICE\_STATE ADDRESS.ADDRS\_STATE%TYPE,

PREFFERED\_START RESERVATION.RESERV\_START%TYPE,

PREFFERED\_END RESERVATION.RESERV\_END%TYPE)

AS

/\*Variable Declaration\*/

TEMP\_TRAV TRAVELER.TRAVELER\_ID%TYPE;

TEMP\_ROOM\_ID ROOM.ROOM\_ID%TYPE;

TEMP\_STREET ADDRESS.ADDRS\_STREET%TYPE;

TEMP\_CITY ADDRESS.ADDRS\_CITY%TYPE;

TEMP\_STATE ADDRESS.ADDRS\_STATE%TYPE;

TEMP\_NBHD\_NAME NEIGHBORHOOD.NBHD\_NAME%TYPE;

TEMP\_NBHD\_TYPE NEIGHBORHOOD.NBHD\_TYPE%TYPE;

TEMP\_NBHD\_FEAT NEIGHBORHOOD.NBHD\_FEAT%TYPE;

TEMP\_WIFI ADDRESS.WIFI%TYPE;

TEMP\_PETS ADDRESS.PETS%TYPE;

TEMP\_CHILDREN ADDRESS.CHILDREN%TYPE;

TEMP\_KITCHEN ADDRESS.KITCHEN%TYPE;

TEMP\_COMMON\_AREA ADDRESS.COMMON\_AREA%TYPE;

TEMP\_OWNER\_OCC ADDRESS.OWNER\_OCC%TYPE;

TEMP\_TELEVISION ADDRESS.TELEVISION%TYPE;

TEMP\_BLDG\_TYPE ADDRESS.BLDG\_TYPE%TYPE;

TEMP\_SMOKING ADDRESS.SMOKING%TYPE;

TEMP\_WASHER\_DRYER ADDRESS.WASHER\_DRYER%TYPE;

TEMP\_POOL ADDRESS.POOL%TYPE;

TEMP\_OTHER ADDRESS.OTHER%TYPE;

TEMP\_COUNTY ADDRESS.ADDRS\_COUNTY%TYPE;

TEMP\_START RESERVATION.RESERV\_START%TYPE;

TEMP\_END RESERVATION.RESERV\_END%TYPE;

TEMP\_LAT ADDRESS.ADDRS\_LATITUDE%TYPE;

TEMP\_LONG ADDRESS.ADDRS\_LONGITUDE%TYPE;

TEMP\_RATE\_DATE RESERVATION.RESERV\_START%TYPE;

TEMP\_RATE RESERVATION.ROOM\_PRICE%TYPE;

TEMP\_INCREMENT RESERVATION.ROOM\_PRICE%TYPE;

/\*Cursor to hold the resulst to insert into the AVAILABLE\_ROOM table\*/

CURSOR C1

IS

SELECT RO.ROOM\_ID,

AA.ADDRS\_STREET,

AA.ADDRS\_CITY,

NBH.NBHD\_NAME,

NBH.NBHD\_TYPE,

NBH.NBHD\_FEAT,

AA.WIFI,

AA.PETS,

AA.CHILDREN,

AA.KITCHEN,

AA.COMMON\_AREA,

AA.OWNER\_OCC,

AA.TELEVISION,

AA.BLDG\_TYPE,

AA.SMOKING,

AA.WASHER\_DRYER,

AA.POOL,

AA.OTHER,

AA.ADDRS\_LATITUDE,

AA.ADDRS\_LONGITUDE

FROM ROOM RO

INNER JOIN

(SELECT RO.ROOM\_ID FROM ROOM RO

MINUS

(SELECT RE.ROOM\_ID

FROM ROOM RO

INNER JOIN reservation RE

ON RO.ROOM\_ID = RE.ROOM\_ID

INNER JOIN ADDRESS AA

ON RO.ADDRS\_ID = AA.ADDRS\_ID

/\*Find all dates that are not already booked\*/

WHERE ((PREFFERED\_START <= RE.RESERV\_START

AND RE.RESERV\_START <= PREFFERED\_END)

OR(PREFFERED\_START < RE.RESERV\_END

AND RE.RESERV\_END <= PREFFERED\_END))

AND AA.ADDRS\_COUNTY = OFFICE\_COUNTY

AND RE.ISCANC = 'N'

GROUP BY RE.ROOM\_ID

)

) ROOMS ON RO.ROOM\_ID = ROOMS.ROOM\_ID

INNER JOIN ADDRESS AA

ON RO.ADDRS\_ID = AA.ADDRS\_ID

INNER JOIN NEIGHBORHOOD NBH

ON AA.NBHD\_ID = NBH.NBHD\_ID

INNER JOIN (SELECT AA.ADDRS\_ID,DE.CO\_ID,HO.HOST\_ID FROM ADDRESS AA

INNER JOIN HOST HO

ON AA.HOST\_ID = HO.HOST\_ID

INNER JOIN EMPLOYEE EE

ON EE.EMP\_ID = HO.HOST\_ID

INNER JOIN DEPARTMENT DE

ON DE.DEPT\_ID = EE.DEPT\_ID

WHERE HO.HOST\_ID <> TEMP\_TRAV) HO

ON AA.ADDRS\_ID = HO.ADDRS\_ID

/\*Limit the results to just those within the proper region\*/

WHERE AA.ADDRS\_COUNTY = OFFICE\_COUNTY

AND AA.ADDRS\_STATE = OFFICE\_STATE

AND HO.CO\_ID =

(SELECT DE.CO\_ID FROM TRAVELER TR

INNER JOIN EMPLOYEE EE

ON EE.EMP\_ID = TR.TRAVELER\_ID

INNER JOIN DEPARTMENT DE

ON DE.DEPT\_ID = EE.DEPT\_ID

WHERE TR.TRAVELER\_ID = TEMP\_TRAV

GROUP BY DE.CO\_ID)

;

BEGIN

/\*Toggle this command depending on how the AVAIABLE\_ROOM table will be used

if it is used to hold all room availability queries ever toggle it off

if it is used to hold only the current session toggle it on. By default

it is set to off\*/

--delete AVAILABLE\_ROOM;

/\*Assign values to those variables that are not held in the cursor\*/

TEMP\_TRAV := TRAV\_CHECKING;

TEMP\_COUNTY := OFFICE\_COUNTY;

TEMP\_STATE := OFFICE\_STATE;

TEMP\_START := PREFFERED\_START;

TEMP\_END := PREFFERED\_END;

TEMP\_RATE\_DATE := PREFFERED\_START;

TEMP\_RATE := 0;

TEMP\_INCREMENT := 0;

/\*For loop to assign values to all variables inside the cursor\*/

IF TO\_DATE(TEMP\_START) < TO\_DATE(SYSDATE)

THEN

raise\_application\_error

(-20009, 'Please enter a valid date equal to or after today''s date.');

ELSE

FOR C1\_REC IN C1

LOOP

TEMP\_ROOM\_ID := C1\_REC.ROOM\_ID;

TEMP\_STREET := C1\_REC.ADDRS\_STREET;

TEMP\_CITY := C1\_REC.ADDRS\_CITY;

TEMP\_NBHD\_NAME := C1\_REC.NBHD\_NAME;

TEMP\_NBHD\_TYPE := C1\_REC.NBHD\_TYPE;

TEMP\_NBHD\_FEAT := C1\_REC.NBHD\_FEAT;

TEMP\_WIFI := C1\_REC.WIFI;

TEMP\_PETS := C1\_REC.PETS;

TEMP\_CHILDREN := C1\_REC.CHILDREN;

TEMP\_KITCHEN := C1\_REC.KITCHEN;

TEMP\_COMMON\_AREA := C1\_REC.COMMON\_AREA;

TEMP\_OWNER\_OCC := C1\_REC.OWNER\_OCC;

TEMP\_TELEVISION := C1\_REC.TELEVISION;

TEMP\_BLDG\_TYPE := C1\_REC.BLDG\_TYPE;

TEMP\_SMOKING := C1\_REC.SMOKING;

TEMP\_WASHER\_DRYER := C1\_REC.WASHER\_DRYER;

TEMP\_POOL := C1\_REC.POOL;

TEMP\_OTHER := C1\_REC.OTHER;

TEMP\_LAT := C1\_REC.ADDRS\_LATITUDE;

TEMP\_LONG := C1\_REC.ADDRS\_LONGITUDE;

/\* If statement checks to see if date is valid. If date is before todays

date we raise an application error to warn the customer to choose a

date that is actually possible\*/

/\*Assign default values to TEMP\_RATE\_DATE and TEMP\_RATE before the loop

to calculate the total amount of the stay\*/

TEMP\_RATE\_DATE := TEMP\_START;

TEMP\_RATE := 0;

/\*Loop to calculate the total amount that will be charged for each

potential stay.\*/

WHILE TEMP\_RATE\_DATE < TEMP\_END

LOOP

SELECT (RR.REGION\_RATE \* .9) INTO TEMP\_INCREMENT

FROM ROOM RO

INNER JOIN ADDRESS AA

ON RO.ADDRS\_ID = AA.ADDRS\_ID

INNER JOIN REGION RE

ON TRIM(LOWER(AA.ADDRS\_COUNTY)) = TRIM(LOWER(RE.COUNTY))

INNER JOIN REGION\_RATE RR

ON RE.REG\_ID = RR.REG\_ID

WHERE RR.RATE\_BEGIN\_DATE <= TEMP\_RATE\_DATE AND TEMP\_RATE\_DATE <= (ADD\_MONTHS(RATE\_BEGIN\_DATE,1)-1)

AND RO.ROOM\_ID = TEMP\_ROOM\_ID;

/\*Assignment for each value during each loop\*/

TEMP\_RATE := TEMP\_RATE + NVL(TEMP\_INCREMENT,0);

TEMP\_RATE\_DATE := TEMP\_RATE\_DATE + 1;

END LOOP;

/\*Insert each row into a table to capture each value.\*/

INSERT

INTO AVAILABLE\_ROOM

(

TRAVELER\_ID,

ROOM\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_STATE,

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_CHECK\_DATE

)

VALUES

(

TEMP\_TRAV,

TEMP\_ROOM\_ID,

TEMP\_START,

TEMP\_END,

TEMP\_RATE,

TEMP\_STREET,

TEMP\_CITY,

TEMP\_STATE,

TEMP\_NBHD\_NAME,

TEMP\_NBHD\_TYPE,

TEMP\_NBHD\_FEAT,

TEMP\_WIFI,

TEMP\_PETS,

TEMP\_CHILDREN,

TEMP\_KITCHEN,

TEMP\_COMMON\_AREA,

TEMP\_OWNER\_OCC,

TEMP\_TELEVISION,

TEMP\_BLDG\_TYPE,

TEMP\_SMOKING,

TEMP\_WASHER\_DRYER,

TEMP\_POOL,

TEMP\_OTHER,

TEMP\_LAT,

TEMP\_LONG,

SYSDATE

);

END LOOP;

END IF;

/\*Commit all changes after the loop has finished\*/

COMMIT;

/\* Exception handling for the case when there are no rooms available\*/

EXCEPTION

WHEN no\_data\_found THEN

raise\_application\_error (-20005, 'No rooms found for these dates');

END;

/

/\*

EXECUTE AVAILABLE\_ROOMS ('EMP000000037','Santa Clara','CA','10-DEC-15','20-DEC-15');

SELECT \* FROM AVAILABLE\_ROOM;

---query for usefull information from available\_room

SELECT ADDRS\_STREET AS "Address Street",

ADDRS\_CITY AS "Address City",

BLDG\_TYPE AS "Building Type",

RESERV\_START AS "Reservation Start Date",

RESERV\_END AS "Reservation End Date",

ROOM\_PRICE AS "Reservation Amount",

NBHD\_NAME AS "Neighborhood",

NBHD\_TYPE AS "Neighborhood Type",

NBHD\_FEAT AS "Neighborhood Features",

WIFI AS "Wifi included",

PETS AS "Pets",

CHILDREN AS "Children",

KITCHEN AS "Kitchen Available",

COMMON\_AREA AS "Shared Common Area",

OWNER\_OCC AS "Owner Occupied",

TELEVISION AS "Television in Room",

SMOKING AS "Smoking allowed",

WASHER\_DRYER AS "Washer/Dryer Available",

POOL AS "Pool Available",

OTHER AS "Other Features"

FROM AVAILABLE\_ROOM AA

WHERE AA.TRAVELER\_ID = \*ACTIVE\_TRAVELER\*

AND AA.AVAIL\_CHECK\_DATE = (SELECT MAX(AVAIL\_CHECK\_DATE) AS MAX\_CHECK FROM AVAILABLE\_ROOM BB WHERE BB.TRAVELER\_ID = AA.TRAVELER\_ID GROUP BY BB.TRAVELER\_ID);

\*/

## Procedure 4: Room Rate

This procedure is designed to create a list of rooms and dates with the daily price of each room on each date based on the ROOM\_ID that is prompted, the dates the active traveler is interested in, and the active traveler\_id at run time.

To create this list of rooms and dates we join the REGION\_RATE, REGION, ADDRESS, and the ROOM table.

We initialize a variable that uses the date the traveler wants to start their travel and use it as the loop index variable checking all dates from the date they want to start their stay up until one day before the traveler wants to end their stay. Then we find rates by look for where the loop index variable (temp\_date\_start) to make sure it falls between RATE\_BEGIN\_DATE and one month after the RATE\_BEGIN\_DATE. We ind the one month after date by using the ADD\_MONTHS function and adding one month.

On each loop we append one row to the ROOM\_RATE and calculate the daily rate as REGION\_RATE \* .9, CoHost policy is that room rates are locked to 90% of the rate charged for the region the room is in.

Once finished the result is the detail view of the table that the procedure to find available rooms builds. This is used to see each daily price that makes up the room\_price in the reservation and available\_room table.

Parameters

roomno: This is the ROOM\_ID that the active traveler would like to get more

detail about.

date\_want\_start: This is the date that the active traveler is interested in

starting their trip.

date\_want\_end: This is the date that the active traveler is interested in

ending their trip.

query\_trav: this is the active traveler id that is currently using the

website.

Exception Handling:

There are 2 exceptions handled by this procedure.

1. If the user has entered a date before today's date they are prompted to

enter a valid date.

2. If any other exception happens the ORA-20002 exception is thrown and the user

is given a brief message explaining that the traveler id or the room id does

not exist.

Output: This procedure appends data to the room price table.

CREATE OR REPLACE PROCEDURE ROOM\_PRICE

(roomno room.room\_id%type,

date\_want\_start reservation.reserv\_start%type,

date\_want\_end reservation.reserv\_end%type,

query\_trav employee.emp\_id%type)

as

temp\_date\_start reservation.reserv\_start%type;

temp\_rate region\_rate.region\_rate%type;

temp\_room room.room\_id%type;

temp\_date\_end reservation.reserv\_end%type;

temp\_trav employee.emp\_id%type;

BEGIN

DELETE ROOM\_RATE;

temp\_date\_start := date\_want\_start;

temp\_date\_end := date\_want\_end ;

IF TO\_DATE(temp\_date\_end) < TO\_DATE(SYSDATE)

THEN

raise\_application\_error

(-20009, 'Please enter a date greater than or equal to today''s date.');

ELSE

/\*While loop to move through all days that a traveler would be staying with a host.\*/

WHILE temp\_date\_end > temp\_date\_start

LOOP

SELECT roomno into temp\_room from dual;

SELECT query\_trav into temp\_trav from dual;

/\*Calculate each rate based on the date and price of that date\*/

SELECT (RR.REGION\_RATE \* .9) AS RATE\_CHARGED INTO temp\_rate FROM

REGION\_RATE RR

INNER JOIN REGION RE

ON RR.REG\_ID = RE.REG\_ID

INNER JOIN ADDRESS AA

ON TRIM(LOWER(AA.ADDRS\_COUNTY)) = TRIM(LOWER(RE.COUNTY))

INNER JOIN

ROOM RO

ON AA.ADDRS\_ID = RO.ADDRS\_ID

WHERE room\_id = temp\_room

and temp\_date\_start >= RR.RATE\_BEGIN\_DATE

and temp\_date\_start <= (ADD\_MONTHS(RR.RATE\_BEGIN\_DATE,1)-1);

/\*Insert the new values into the ROOM\_RATE table so that they can be displayed

in the front end\*/

INSERT INTO ROOM\_RATE (ROOM\_ID, TRAVELER\_ID, ROOM\_RATE, ROOM\_DATE, RUN\_DATE)

VALUES (TEMP\_ROOM, TEMP\_TRAV, TEMP\_RATE, temp\_date\_start, SYSDATE);

/\*iterate the looping vairable\*/

temp\_date\_start := temp\_date\_start + 1;

END LOOP;

END IF;

/\*Handle the no data found exception\*/

COMMIT;

Exception

When no\_data\_found Then

raise\_application\_error

(-20002, TEMP\_TRAV || ' or ' || TEMP\_ROOM ||' does not exist.');

END;

/

/\*Test Values used to verify procedure\*/

/\*

EXECUTE ROOM\_PRICE ('ROO000000010','30-DEC-15','05-JAN-16','EMP000000004');

COMMIT;

SELECT \* FROM ROOM\_RATE AA

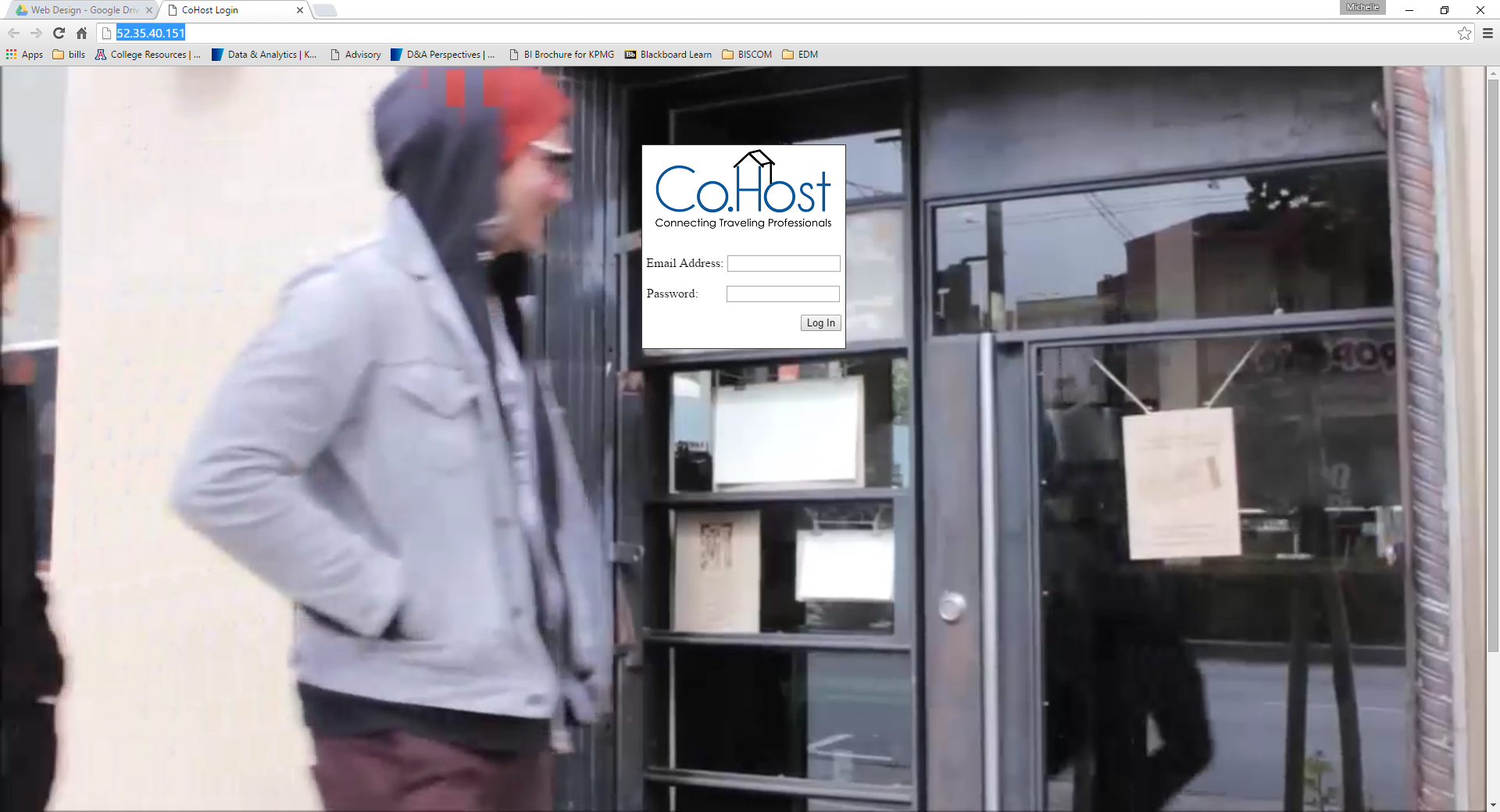
WHERE AA.TRAVELER\_ID = \*ACTIVE TRAVELER\*

AND AA.RUN\_DATE = (SELECT MAX(RUN\_DATE) ROOM\_RATE BB WHERE BB.TRAVELER\_ID = AA.TRAVELER\_ID GROUP BY TRAVELER\_ID);

\*/

# Chapter 6: Interface and Reports

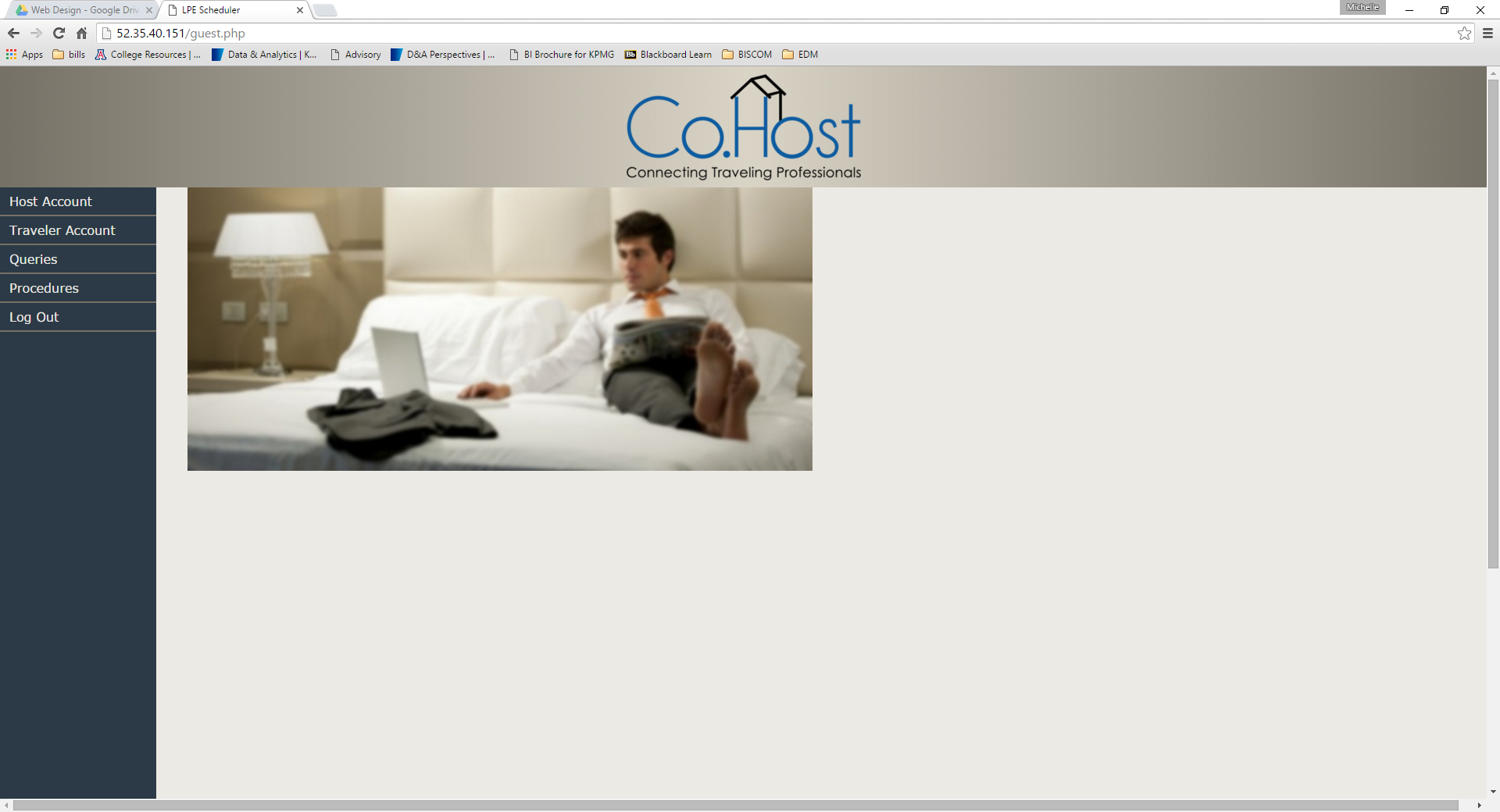
1)



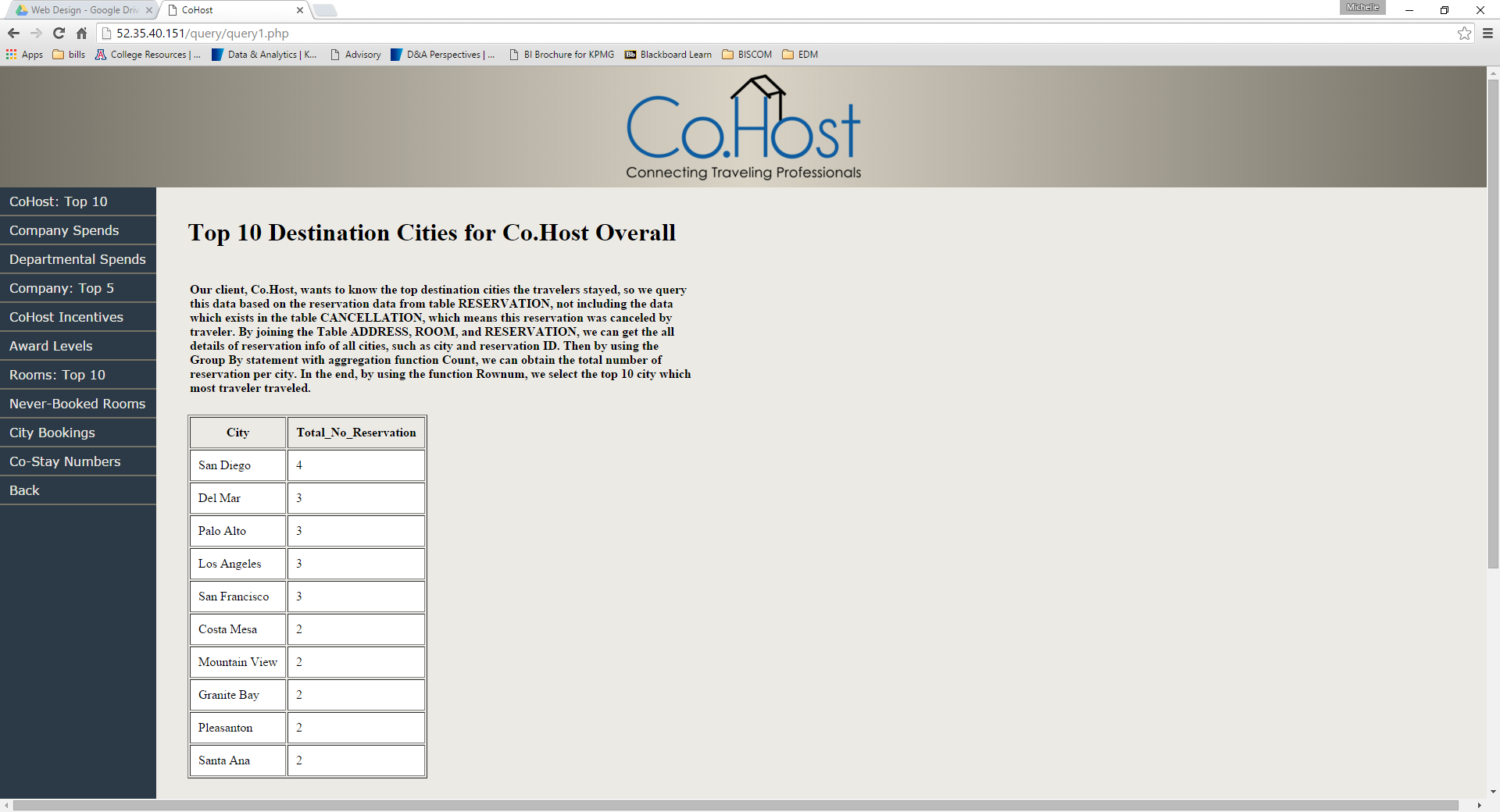
Go to: <http://52.35.40.151/>

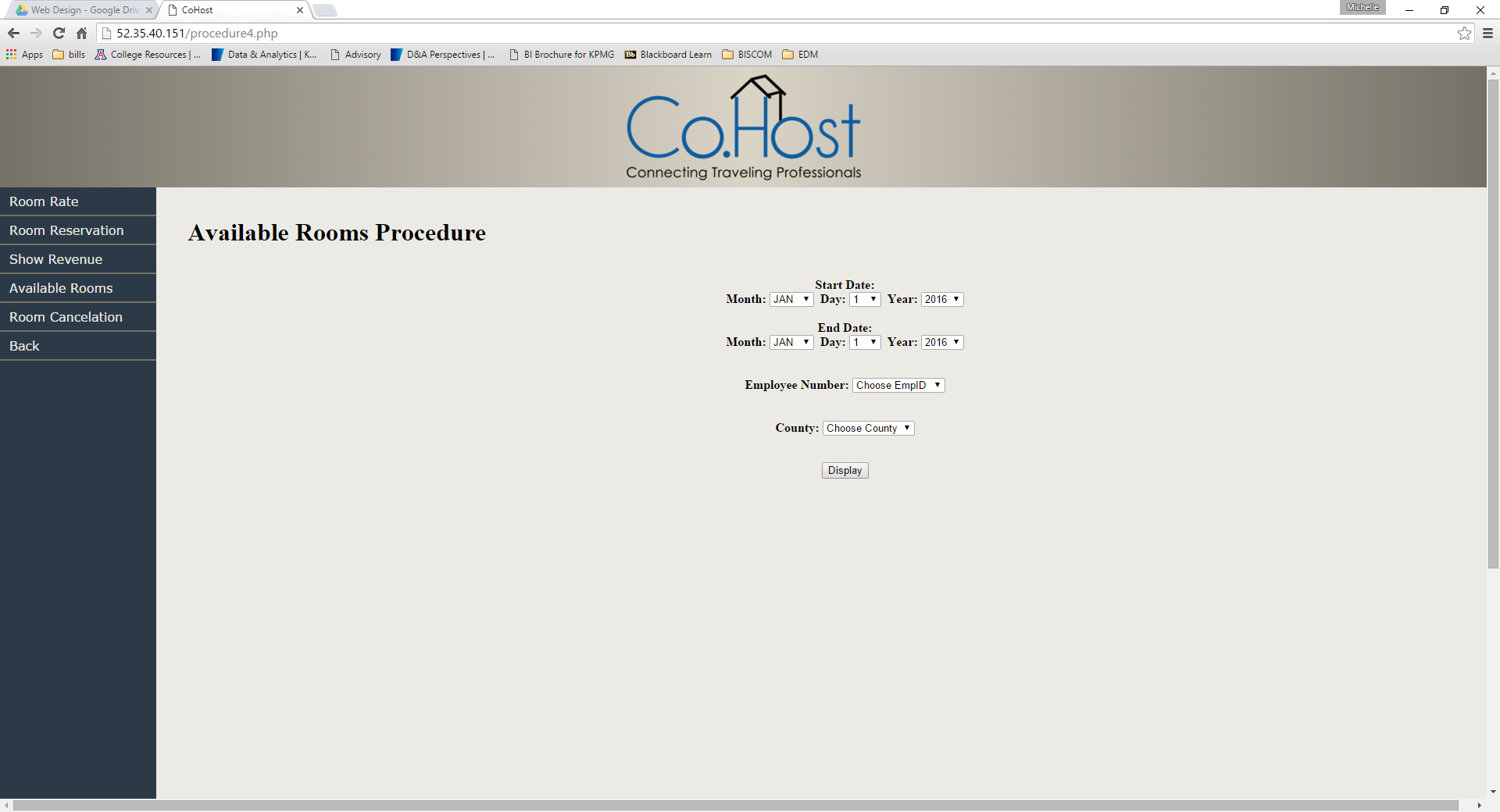
Enter email address: [login@intuit.com](mailto:login@intuit.com)  
Enter password: cohost

2)

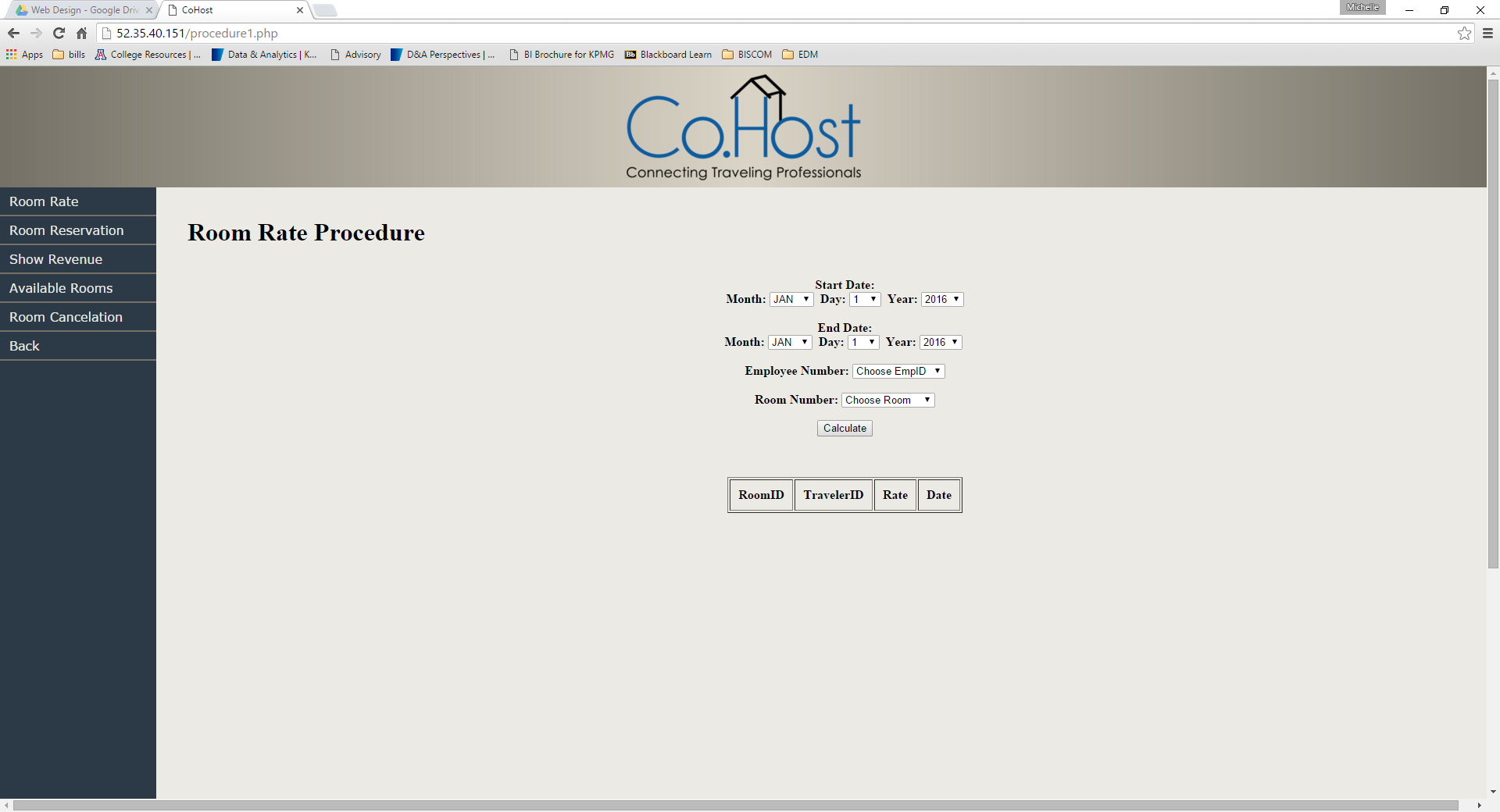


Host Account and Traveler Account do not perform any operations at this time. Select Queries.

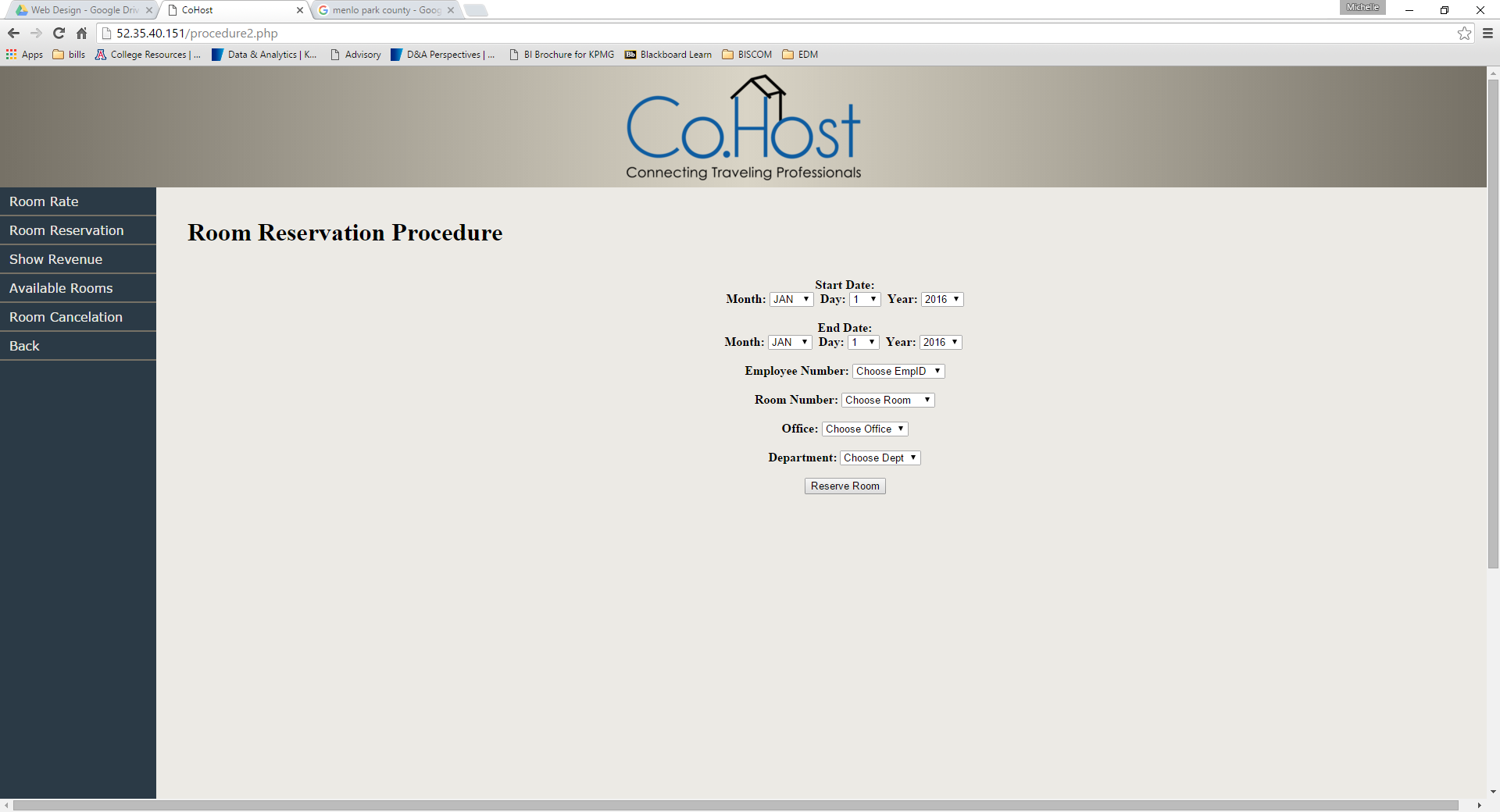
3)You can pull up each of the reports listed on the Queries menu. You will be able to see updates after running the procedures/triggers in the following steps. Select room 3 when creating the reservation for employee 4 if you would like to see the Co-Stay Numbers report change.

4)

Available Rooms shows open rooms for your travel dates. Enter the start date and end date (start date must be before end date). Select the employee number (please select employee 4 for the best results throughout the site tour). Select the County (addresses are located in the following counties for employee 4’s company: San Francisco, San Diego, San Mateo, and Santa Clara) Please remember the selected county for step 6.

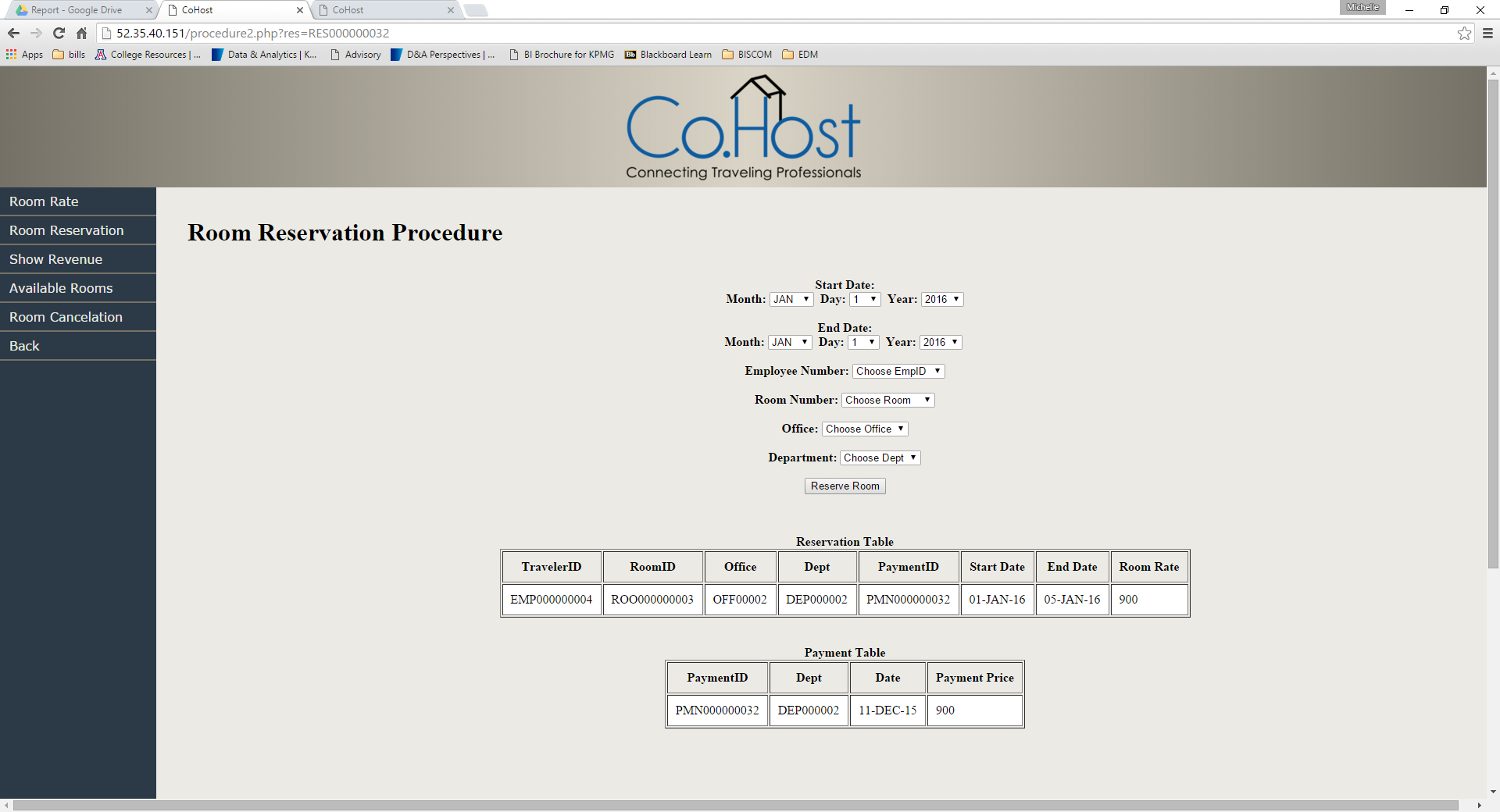
5)

For the Room Rate Procedure, enter a start date and an end date: the end date must be after the start date. Select the employee number (again, please select employee 4 for best results) and a room number. Rooms 1-7, 9-14, 23, and 24 belong to Intuit employees (employee 4’s company), so one of these rooms should be selected in order to avoid any integrity issues.

6) 

To make a reservation, select start and end date (start date must be before end date). Choose employee number (4), the room number that you would like from the previous Room Rate Procedure, the Office you will be working from (list of offices by county follows so that you can select an office in the county you selected in step 4 above), and the department (departments 1-14 belong to Intuit).

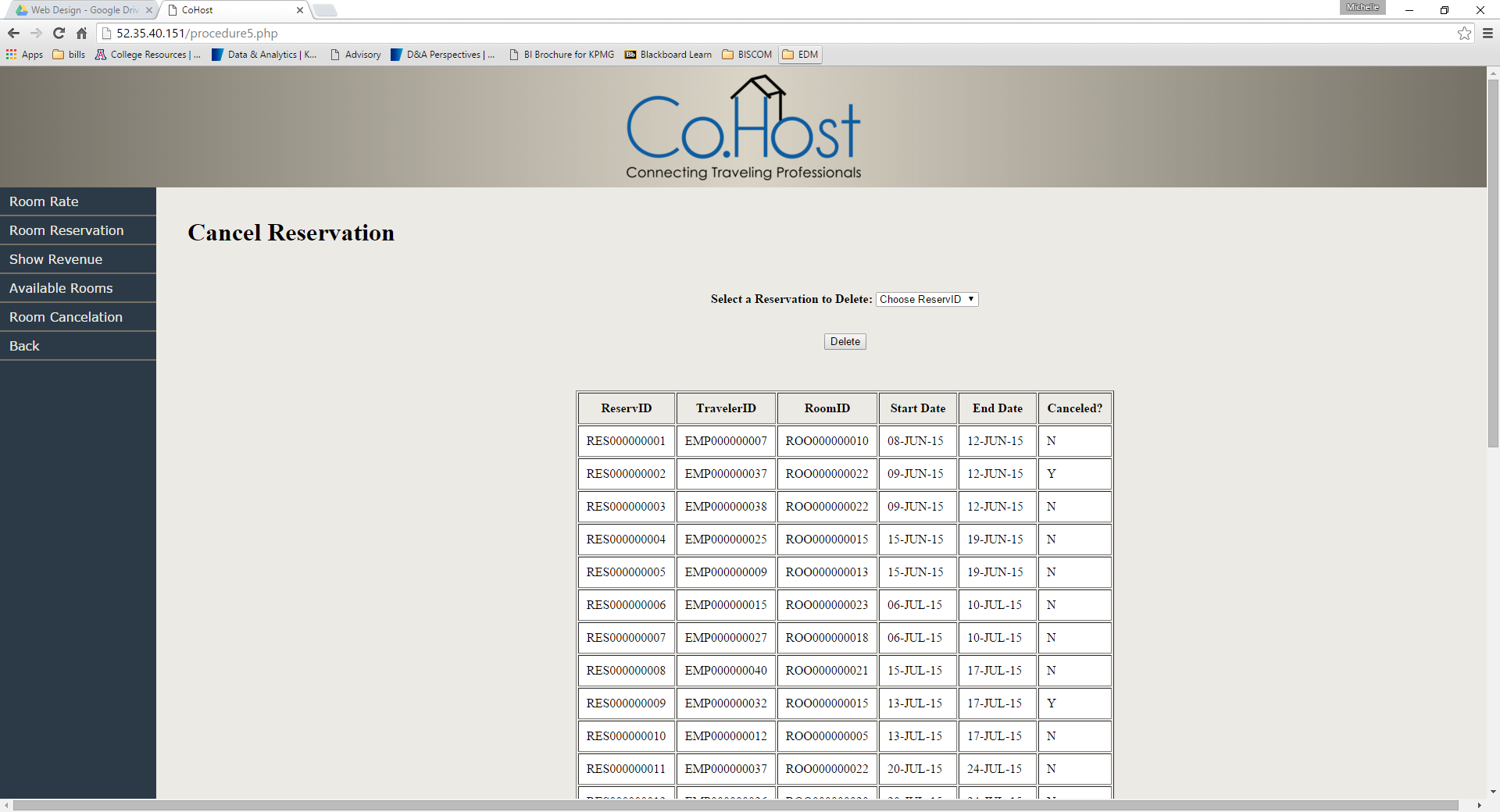
|  |  |
| --- | --- |
| County | Office # |
| San Francisco | 1, 2 |
| San Diego | 4 |
| San Mateo | 3 |
| Santa Clara | 6 |



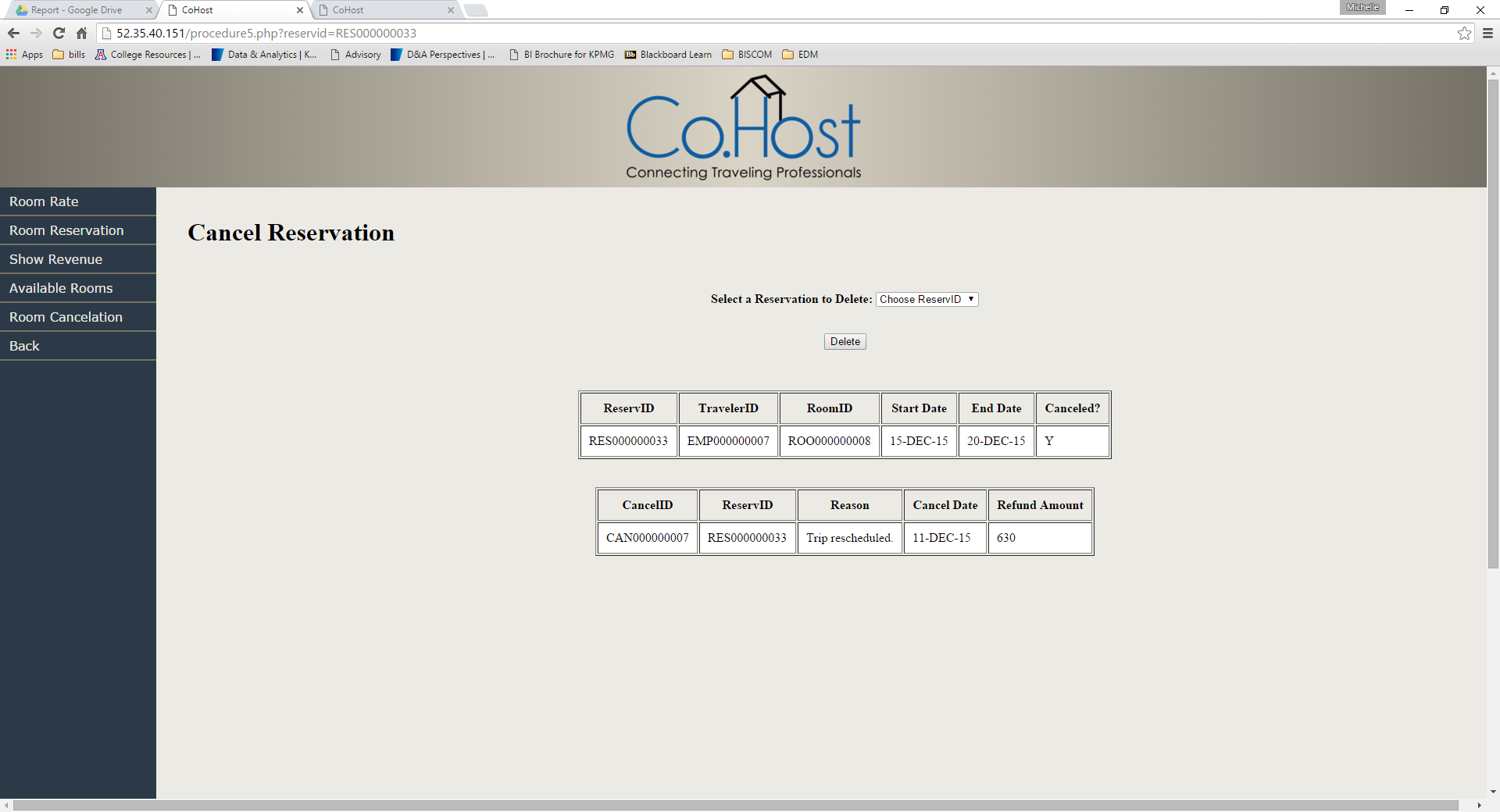
The new reservation and payment records will be displayed after reserving the room.

7) 

For Show Revenue, select the start date and end date for the period you are interested in to see the revenue broken down by city. Be sure to select a start date before the end date. Additionally, because revenue is recorded based on payments, the Total Revenue will be based on the payment date rather than the reservation date. If you reserved for January but booked in December, the revenue will show up in December. If a reservation is cancelled, the refunded amount will be deducted from the revenue. If the reservation is cancelled less than 24 hours before the start of the reservation, the refund amount will be $0 and the revenue data will not be reduced.

8) 

For Cancel Reservation, select a reservation from the list that has a Canceled? value of “N”. You can cancel any reservation that has not yet been canceled, however refunds will only be given if the cancellation is made at least 24 hours before the reservation start date. You will then see the updated record in the reservation and cancellation tables:



You can then return to step 7 to see the change in the Revenue table.

The above procedures/triggers allow update and/or insert on the following tables:

CANCELLATION (insert of cancellation record upon cancellation)  
RESERVATION (insert/update of reservation record upon making reservation or cancellation)  
 \*A cancellation will change the status of a reservation to Canceled? = “Y”  
DEPARTMENT (update to total department spend record upon cancellation)  
COMPANY (update to total company spend record upon cancellation)  
PAYMENT (insert of payment record upon making reservation)

Additionally, staging tables are created when running the following procedures: Available Rooms, Room Rates, and Show Revenue.

We do not want to delete any records because this could cause issues with foreign keys and impact the reliability of analytical data. Instead, we use active flags where needed (e.g. the Canceled? Y/N flag).

# Chapter 7: Conclusions and implementation plan

During the course of this project, we were able to build a database from the ground up. Our original Entity Relationship model gave us a good starting point for our project. After creating the tables in the database, the relationships between the entities and the weaknesses in our original model became apparent. This gave us a better understanding of how the information was connected and how data should flow between tables. We were able to update our diagram to reflect the better designs discovered during table-creation. Conversely, we were also able to use the model to identify tables that could be combined to prevent redundancy.

We learned that building a functioning database requires iterative revisions to each part of the process. We found that even with a good plan and understanding of the basic needs and structure, it was impossible to predict all of the changes that each new step would introduce. We made countless revisions during the process to ensure that all parts remained consistent. This meant keeping up with changes to every part of the project as they became necessary so that we did not have missing information in our data. We grew to despise changes more and more as the project went on because of the increase in the amount of time even small changes would take.

We also learned that a good team dynamic and frequent communication was a great asset. We met on a weekly basis during the entire semester to ensure that we were able to keep tabs on our progress. We were lucky to be able to meet with our client almost every week as well so that any concerns or new requirements they had could be addressed quickly.

## Implementation plan:

**Phase 1**: Walk client through the process of spinning up a server. We will suggest using Amazon Web Service as it allows for great flexibility at a reasonable price. (Dec. 28)

**Phase 2**: Help the client set up the database. We will have them run the table creation statements that we will provide via an Excel document that helps to streamline the process. We will help walk them through this process, but we want to ensure that they understand the process in case they encounter any problems and need to start the process over again in the future. This phase will also include setting up the user rights within the system so that they can ensure the security of their data. Our client is still getting ready to officially launch their pilot program, so there will not be any data to migrate into the tables. (Dec. 29-30)

**Phase 3**: Integrate the front-end. As a web-based company, the front-end will be very important to our client. They have set aside funds to pay for a professional service to set up their website, so we will have minimal involvement with this part of the process. We will be happy to answer any questions they might have regarding the connectivity of the database that we have developed for them and any potential front-end solutions they are looking into. (January 2016)

## Hours and Costs Breakdown

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Number of People (including client) | Total Person Hours | Hardware/ Software Needed | Hardware/ Software Costs\* | Total Cost |
| Phase 1 | 3 | 15 | -AWS Server | $370 | $370 |
| Phase 2 | 3 | 24 | -Excel  -Oracle SQL Developer | $0  $0 | $0 |
| Phase 3 | As needed | As needed | Unknown | Unknown | Unknown |

\*Server cost represents monthly cost to run 15G AWS db.r3.large server. This should be adequate for the first 1-2 years of operation.

# /\*Appendix to Chapter 3: Create Statements, Sequences, Sequence Triggers, Insert Statements, Alter Table Constraints\*/

## /\*Create Table Statements\*/

DROP TABLE JABBERWOCKY.EMPLOYEE CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.EMPLOYEE

(

EMP\_ID VARCHAR(12) CONSTRAINT EMPLOYEE\_EMPID\_PK PRIMARY KEY ,

DEPT\_ID VARCHAR(9),

OFF\_ID VARCHAR(8),

REWARD\_ID VARCHAR(12),

GENDER\_ID VARCHAR(12),

BANNED VARCHAR(2) NOT NULL,

CURRENT\_REW\_LVL VARCHAR(50) NOT NULL,

EMP\_POINTS DECIMAL(24,6) NOT NULL,

EMP\_DOB DATE NOT NULL,

EMP\_PHOTO VARCHAR(2000) NOT NULL,

HIGEST\_REW\_LV VARCHAR(100) NOT NULL,

POINTS\_TO\_NXT\_RWD DECIMAL(24,6) NOT NULL,

EMP\_FIRST VARCHAR(40) NOT NULL,

EMP\_LAST VARCHAR(40) NOT NULL,

EMAIL VARCHAR(100) NOT NULL,

ACTIVE\_FLG VARCHAR(2) DEFAULT 'Y'

);

DROP TABLE JABBERWOCKY.PASSWORD CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID VARCHAR(10),

EMP\_ID VARCHAR(12),

PASSWORD\_DATE DATE NOT NULL,

USER\_PASSWORD VARCHAR(254) NOT NULL

);

DROP TABLE JABBERWOCKY.REWARD\_SYSTEM CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_ID VARCHAR(12) CONSTRAINT REWARD\_SYSTEM\_REWARDID\_PK PRIMARY KEY ,

REWARD\_TYPE VARCHAR(40) NOT NULL,

REWARD\_NAME VARCHAR(254) NOT NULL

);

DROP TABLE JABBERWOCKY.DEPARTMENT CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.DEPARTMENT

(

DEPT\_ID VARCHAR(9) CONSTRAINT DEPARTMENT\_DEPTID\_PK PRIMARY KEY ,

CO\_ID VARCHAR(8),

DEPT\_NAME VARCHAR(100) NOT NULL,

TOT\_DEPT\_SPEND DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.REWARD\_LOG CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.REWARD\_LOG

(

EMP\_ID VARCHAR(12),

PMNT\_ID VARCHAR(12),

REWARD\_DATE DATE NOT NULL,

REWARD\_AMT DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.TRAVELER CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.TRAVELER

(

TRAVELER\_ID VARCHAR(12) CONSTRAINT TRAVELER\_TRAVELERID\_PK PRIMARY KEY ,

TRAVELER\_RATING DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.HOST CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.HOST

(

HOST\_ID VARCHAR(12) CONSTRAINT HOST\_HOSTID\_PK PRIMARY KEY ,

HOST\_RATING DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.SURVEY\_QUESTIONS CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.SURVEY\_QUESTIONS

(

QUESTION\_ID VARCHAR(10) CONSTRAINT SURVEY\_QUESTIONS\_QUESTIONID\_PK PRIMARY KEY ,

SURVEY\_TYPE VARCHAR(10) NOT NULL,

QUESTION VARCHAR(2000) NOT NULL,

QUESTION\_ADD\_DATE DATE NOT NULL,

IS\_ACTIVE VARCHAR(2) NOT NULL

);

DROP TABLE JABBERWOCKY.RESERVATION CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.RESERVATION

(

RESERV\_ID VARCHAR(12) CONSTRAINT RESERVATION\_RESERVID\_PK PRIMARY KEY ,

TRAVELER\_ID VARCHAR(12),

ROOM\_ID VARCHAR(12),

OFF\_ID VARCHAR(8),

PMNT\_ID VARCHAR(12),

DEPT\_ID VARCHAR(9),

RESERV\_START DATE NOT NULL,

RESERV\_END DATE NOT NULL,

ROOM\_PRICE DECIMAL(24,6) DEFAULT 0,

ISCANC VARCHAR(2) DEFAULT 'N'

);

DROP TABLE JABBERWOCKY.CANCELLATION CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.CANCELLATION

(

CANC\_ID VARCHAR(12) CONSTRAINT CANCELLATION\_CANCID\_PK PRIMARY KEY ,

RESERV\_ID VARCHAR(12),

CANC\_LIMIT DATE NOT NULL,

CANC\_REASON VARCHAR(2000),

CANC\_DATE DATE NOT NULL,

REF\_AMT DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.ROOM CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.ROOM

(

ROOM\_ID VARCHAR(12) CONSTRAINT ROOM\_ROOMID\_PK PRIMARY KEY ,

ADDRS\_ID VARCHAR(12),

SQR\_FEET DECIMAL(24,6),

BED\_COUNT DECIMAL(3,0) NOT NULL,

PRIVATE\_BTHRM VARCHAR(3) NOT NULL,

ROOM\_COMMENTS VARCHAR(2000) NOT NULL

);

DROP TABLE JABBERWOCKY.ADDRESS CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.ADDRESS

(

ADDRS\_ID VARCHAR(12) CONSTRAINT ADDRESS\_ADDRSID\_PK PRIMARY KEY ,

HOST\_ID VARCHAR(12),

NBHD\_ID VARCHAR(6),

ADDRS\_STREET VARCHAR(200) NOT NULL,

ADDRS\_CITY VARCHAR(200) NOT NULL,

ADDRS\_COUNTY VARCHAR(100) NOT NULL,

ADDRS\_STATE VARCHAR(100) NOT NULL,

ADDRS\_ZIP VARCHAR(10) NOT NULL,

ADDRS\_COUNTRY VARCHAR(200) NOT NULL,

ADDRS\_LATITUDE VARCHAR(100) NOT NULL,

ADDRS\_LONGITUDE VARCHAR(100) NOT NULL,

AVAIL\_RMS VARCHAR(2) NOT NULL,

WIFI VARCHAR(2) NOT NULL,

PETS VARCHAR(2) NOT NULL,

CHILDREN VARCHAR(2) NOT NULL,

KITCHEN VARCHAR(2) NOT NULL,

COMMON\_AREA VARCHAR(2) NOT NULL,

OWNER\_OCC VARCHAR(2) NOT NULL,

TELEVISION VARCHAR(2) NOT NULL,

BLDG\_TYPE VARCHAR(200) NOT NULL,

SMOKING VARCHAR(2) NOT NULL,

WASHER\_DRYER VARCHAR(2) NOT NULL,

POOL VARCHAR(2) NOT NULL,

OTHER VARCHAR(2000)

);

DROP TABLE JABBERWOCKY.SURVEYS CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.SURVEYS

(

SURVEY\_ID VARCHAR(10) CONSTRAINT SURVEYS\_SURVEYID\_PK PRIMARY KEY ,

RESERV\_ID VARCHAR(12),

EMP\_ID VARCHAR(12),

SURVEY\_TYPE VARCHAR(10) NOT NULL,

COMP\_DATE DATE NOT NULL

);

DROP TABLE JABBERWOCKY.ANSWERS CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.ANSWERS

(

QUESTION\_ID VARCHAR(10),

SURVEY\_ID VARCHAR(10),

ANSWER VARCHAR(2000) NOT NULL

);

DROP TABLE JABBERWOCKY.COMPANY CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.COMPANY

(

CO\_ID VARCHAR(8) CONSTRAINT COMPANY\_COID\_PK PRIMARY KEY ,

CO\_NAME VARCHAR(200) NOT NULL,

TOTAL\_CO\_SPEND DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.CO\_CONTACT CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.CO\_CONTACT

(

CONT\_ID VARCHAR(8) CONSTRAINT CO\_CONTACT\_CONTID\_PK PRIMARY KEY ,

CO\_ID VARCHAR(8),

CONT\_FIRST VARCHAR(200) NOT NULL,

CONT\_LAST VARCHAR(200) NOT NULL,

CONT\_BEGIN\_DATE DATE NOT NULL,

CONT\_END\_DATE DATE,

CONT\_EMAIL VARCHAR(200) NOT NULL

);

DROP TABLE JABBERWOCKY.OFFICE CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.OFFICE

(

OFF\_ID VARCHAR(8) CONSTRAINT OFFICE\_OFFID\_PK PRIMARY KEY ,

CO\_ID VARCHAR(8),

REG\_ID VARCHAR(12),

OFF\_STREET VARCHAR(200) NOT NULL,

OFF\_CITY VARCHAR(200) NOT NULL,

OFF\_STATE VARCHAR(100) NOT NULL,

OFF\_ZIP VARCHAR(10) NOT NULL,

OFF\_COUNTRY VARCHAR(200) NOT NULL,

OFF\_LATITUDE VARCHAR(100) NOT NULL,

OFF\_LONGITUDE VARCHAR(100) NOT NULL

);

DROP TABLE JABBERWOCKY.CONTACT\_PHONE CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.CONTACT\_PHONE

(

CONT\_ID VARCHAR(8) CONSTRAINT CONTACT\_PHONE\_CONTID\_PK PRIMARY KEY ,

CONT\_PHONE VARCHAR(20) DEFAULT 'Y',

CURRENT\_PH VARCHAR(2),

PHONE\_TYPE VARCHAR(50)

);

DROP TABLE JABBERWOCKY.REGION\_RAW CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST VARCHAR(300),

COUNTY VARCHAR(300),

OCT15 DECIMAL(24,6),

NOV15 DECIMAL(24,6),

DEC15 DECIMAL(24,6),

JAN16 DECIMAL(24,6),

FEB16 DECIMAL(24,6),

MAR16 DECIMAL(24,6),

APR16 DECIMAL(24,6),

MAY16 DECIMAL(24,6),

JUN16 DECIMAL(24,6),

JUL16 DECIMAL(24,6),

AUG16 DECIMAL(24,6),

SEP16 DECIMAL(24,6)

);

DROP TABLE JABBERWOCKY.REGION CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.REGION

(

REG\_ID VARCHAR(12) CONSTRAINT REGION\_REGID\_PK PRIMARY KEY ,

COUNTY VARCHAR(300) NOT NULL

);

DROP TABLE JABBERWOCKY.NEIGHBORHOOD CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_ID VARCHAR(6) CONSTRAINT NEIGHBORHOOD\_NBHDID\_PK PRIMARY KEY ,

NBHD\_NAME VARCHAR(200),

NBHD\_TYPE VARCHAR(100),

NBHD\_FEAT VARCHAR(2000) NOT NULL

);

DROP TABLE JABBERWOCKY.PAYMENT CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.PAYMENT

(

PMNT\_ID VARCHAR(12) CONSTRAINT PAYMENT\_PMNTID\_PK PRIMARY KEY ,

DEPT\_ID VARCHAR(9),

PMNT\_DATE DATE NOT NULL,

PMNT\_PRICE DECIMAL(24,6) NOT NULL

);

DROP TABLE JABBERWOCKY.ROOM\_BEDS CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.ROOM\_BEDS

(

BED\_NO VARCHAR(6),

ROOM\_ID VARCHAR(12),

BED\_SIZE VARCHAR(20) NOT NULL

);

DROP TABLE JABBERWOCKY.ROOM\_PICTURES CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE VARCHAR(40),

ROOM\_ID VARCHAR(12)

);

DROP TABLE JABBERWOCKY.REGION\_RATE CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.REGION\_RATE

(

RATE\_BEGIN\_DATE DATE,

REG\_ID VARCHAR(12),

REGION\_RATE DECIMAL(24,6)

);

DROP TABLE JABBERWOCKY.ROOM\_RATE CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.ROOM\_RATE

(

RATE\_QUERY\_ID VARCHAR(12) CONSTRAINT ROOM\_RATE\_RATEQUERYID\_PK PRIMARY KEY ,

ROOM\_ID VARCHAR(12),

TRAVELER\_ID VARCHAR(12),

ROOM\_RATE DECIMAL(24,6),

ROOM\_DATE DATE,

RUN\_DATE DATE

);

DROP TABLE JABBERWOCKY.GENDER CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.GENDER

(

GENDER\_ID VARCHAR(12) CONSTRAINT GENDER\_GENDERID\_PK PRIMARY KEY ,

GENDER\_DESCR VARCHAR(100)

);

DROP TABLE JABBERWOCKY.AVAILABLE\_ROOM CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.AVAILABLE\_ROOM

(

AVAIL\_ID VARCHAR(12) CONSTRAINT AVAILABLE\_ROOM\_AVAILID\_PK PRIMARY KEY ,

TRAVELER\_ID VARCHAR(12),

ROOM\_ID VARCHAR(12),

RESERV\_START DATE,

RESERV\_END DATE,

ROOM\_PRICE DECIMAL(24,6),

ADDRS\_STREET VARCHAR(200),

ADDRS\_CITY VARCHAR(200),

ADDRS\_STATE VARCHAR(100),

NBHD\_NAME VARCHAR(200),

NBHD\_TYPE VARCHAR(100),

NBHD\_FEAT VARCHAR(2000),

WIFI VARCHAR(2),

PETS VARCHAR(2),

CHILDREN VARCHAR(2),

KITCHEN VARCHAR(2),

COMMON\_AREA VARCHAR(2),

OWNER\_OCC VARCHAR(2),

TELEVISION VARCHAR(2),

BLDG\_TYPE VARCHAR(200),

SMOKING VARCHAR(2),

WASHER\_DRYER VARCHAR(2),

POOL VARCHAR(2),

OTHER VARCHAR(2000),

ADDRS\_LATITUDE VARCHAR(100),

ADDRS\_LONGITUDE VARCHAR(100),

AVAIL\_CHECK\_DATE DATE

);

DROP TABLE JABBERWOCKY.REVENUE\_PER\_CITY CASCADE CONSTRAINTS;

CREATE TABLE JABBERWOCKY.REVENUE\_PER\_CITY

(

REVPER\_ID VARCHAR(12) CONSTRAINT REVENUE\_PER\_CITY\_REVPERID\_PK PRIMARY KEY ,

CITY VARCHAR(200),

STARTDATE DATE,

ENDDATE DATE,

TOTALREVENUE DECIMAL(24,6)

);

## /\*Sequence Creation Script\*/

DROP SEQUENCE EMPLOYEE\_EMP\_ID\_SEQ;

CREATE SEQUENCE EMPLOYEE\_EMP\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE REWARD\_SYSTEM\_REWARD\_ID\_SEQ;

CREATE SEQUENCE REWARD\_SYSTEM\_REWARD\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE DEPARTMENT\_DEPT\_ID\_SEQ;

CREATE SEQUENCE DEPARTMENT\_DEPT\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99998;

DROP SEQUENCE SURVEY\_QUESTIONS\_QUE\_SEQ;

CREATE SEQUENCE SURVEY\_QUESTIONS\_QUE\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 999998;

DROP SEQUENCE RESERVATION\_RESERV\_ID\_SEQ;

CREATE SEQUENCE RESERVATION\_RESERV\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE CANCELLATION\_CANC\_ID\_SEQ;

CREATE SEQUENCE CANCELLATION\_CANC\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE ROOM\_ROOM\_ID\_SEQ;

CREATE SEQUENCE ROOM\_ROOM\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE ADDRESS\_ADDRS\_ID\_SEQ;

CREATE SEQUENCE ADDRESS\_ADDRS\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE SURVEYS\_SURVEY\_ID\_SEQ;

CREATE SEQUENCE SURVEYS\_SURVEY\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 999998;

DROP SEQUENCE COMPANY\_CO\_ID\_SEQ;

CREATE SEQUENCE COMPANY\_CO\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 9998;

DROP SEQUENCE CO\_CONTACT\_CONT\_ID\_SEQ;

CREATE SEQUENCE CO\_CONTACT\_CONT\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 9998;

DROP SEQUENCE OFFICE\_OFF\_ID\_SEQ;

CREATE SEQUENCE OFFICE\_OFF\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 9998;

DROP SEQUENCE REGION\_REG\_ID\_SEQ;

CREATE SEQUENCE REGION\_REG\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE NEIGHBORHOOD\_NBHD\_ID\_SEQ;

CREATE SEQUENCE NEIGHBORHOOD\_NBHD\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 98;

DROP SEQUENCE PAYMENT\_PMNT\_ID\_SEQ;

CREATE SEQUENCE PAYMENT\_PMNT\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE ROOM\_RATE\_RATE\_QUERY\_ID\_SEQ;

CREATE SEQUENCE ROOM\_RATE\_RATE\_QUERY\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE GENDER\_GENDER\_ID\_SEQ;

CREATE SEQUENCE GENDER\_GENDER\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE AVAILABLE\_ROOM\_AVAIL\_ID\_SEQ;

CREATE SEQUENCE AVAILABLE\_ROOM\_AVAIL\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

DROP SEQUENCE REVENUE\_PER\_CITY\_REVPER\_ID\_SEQ;

CREATE SEQUENCE REVENUE\_PER\_CITY\_REVPER\_ID\_SEQ INCREMENT BY 1 START WITH 1 MAXVALUE 99999998;

## /\*Sequence Trigger Creation Script\*/

CREATE OR REPLACE TRIGGER TRIG\_EMP\_SEQ BEFORE

INSERT ON JABBERWOCKY.EMPLOYEE FOR EACH ROW DECLARE TEMP\_EMP\_NO JABBERWOCKY.EMPLOYEE.EMP\_ID%TYPE ;

BEGIN

SELECT 'EMP'

|| LPAD( TO\_CHAR(EMPLOYEE\_EMP\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_EMP\_NO

FROM DUAL;

:NEW.EMP\_ID := TEMP\_EMP\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_REW\_SEQ BEFORE

INSERT ON JABBERWOCKY.REWARD\_SYSTEM FOR EACH ROW DECLARE TEMP\_REW\_NO JABBERWOCKY.REWARD\_SYSTEM.REWARD\_ID%TYPE ;

BEGIN

SELECT 'REW'

|| LPAD( TO\_CHAR(REWARD\_SYSTEM\_REWARD\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_REW\_NO

FROM DUAL;

:NEW.REWARD\_ID := TEMP\_REW\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_DEP\_SEQ BEFORE

INSERT ON JABBERWOCKY.DEPARTMENT FOR EACH ROW DECLARE TEMP\_DEP\_NO JABBERWOCKY.DEPARTMENT.DEPT\_ID%TYPE ;

BEGIN

SELECT 'DEP'

|| LPAD( TO\_CHAR(DEPARTMENT\_DEPT\_ID\_SEQ.NEXTVAL ),6,'0')

INTO TEMP\_DEP\_NO

FROM DUAL;

:NEW.DEPT\_ID := TEMP\_DEP\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_QUE\_SEQ BEFORE

INSERT ON JABBERWOCKY.SURVEY\_QUESTIONS FOR EACH ROW DECLARE TEMP\_QUE\_NO JABBERWOCKY.SURVEY\_QUESTIONS.QUESTION\_ID%TYPE ;

BEGIN

SELECT 'QUE'

|| LPAD( TO\_CHAR(SURVEY\_QUESTIONS\_QUE\_SEQ.NEXTVAL ),7,'0')

INTO TEMP\_QUE\_NO

FROM DUAL;

:NEW.QUESTION\_ID := TEMP\_QUE\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_RES\_SEQ BEFORE

INSERT ON JABBERWOCKY.RESERVATION FOR EACH ROW DECLARE TEMP\_RES\_NO JABBERWOCKY.RESERVATION.RESERV\_ID%TYPE ;

BEGIN

SELECT 'RES'

|| LPAD( TO\_CHAR(RESERVATION\_RESERV\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_RES\_NO

FROM DUAL;

:NEW.RESERV\_ID := TEMP\_RES\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_CAN\_SEQ BEFORE

INSERT ON JABBERWOCKY.CANCELLATION FOR EACH ROW DECLARE TEMP\_CAN\_NO JABBERWOCKY.CANCELLATION.CANC\_ID%TYPE ;

BEGIN

SELECT 'CAN'

|| LPAD( TO\_CHAR(CANCELLATION\_CANC\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_CAN\_NO

FROM DUAL;

:NEW.CANC\_ID := TEMP\_CAN\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_ROO\_SEQ BEFORE

INSERT ON JABBERWOCKY.ROOM FOR EACH ROW DECLARE TEMP\_ROO\_NO JABBERWOCKY.ROOM.ROOM\_ID%TYPE ;

BEGIN

SELECT 'ROO'

|| LPAD( TO\_CHAR(ROOM\_ROOM\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_ROO\_NO

FROM DUAL;

:NEW.ROOM\_ID := TEMP\_ROO\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_ADD\_SEQ BEFORE

INSERT ON JABBERWOCKY.ADDRESS FOR EACH ROW DECLARE TEMP\_ADD\_NO JABBERWOCKY.ADDRESS.ADDRS\_ID%TYPE ;

BEGIN

SELECT 'ADD'

|| LPAD( TO\_CHAR(ADDRESS\_ADDRS\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_ADD\_NO

FROM DUAL;

:NEW.ADDRS\_ID := TEMP\_ADD\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_SUR\_SEQ BEFORE

INSERT ON JABBERWOCKY.SURVEYS FOR EACH ROW DECLARE TEMP\_SUR\_NO JABBERWOCKY.SURVEYS.SURVEY\_ID%TYPE ;

BEGIN

SELECT 'SUR'

|| LPAD( TO\_CHAR(SURVEYS\_SURVEY\_ID\_SEQ.NEXTVAL ),7,'0')

INTO TEMP\_SUR\_NO

FROM DUAL;

:NEW.SURVEY\_ID := TEMP\_SUR\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_CO\_\_SEQ BEFORE

INSERT ON JABBERWOCKY.COMPANY FOR EACH ROW DECLARE TEMP\_CO\_\_NO JABBERWOCKY.COMPANY.CO\_ID%TYPE ;

BEGIN

SELECT 'CO\_'

|| LPAD( TO\_CHAR(COMPANY\_CO\_ID\_SEQ.NEXTVAL ),5,'0')

INTO TEMP\_CO\_\_NO

FROM DUAL;

:NEW.CO\_ID := TEMP\_CO\_\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_CON\_SEQ BEFORE

INSERT ON JABBERWOCKY.CO\_CONTACT FOR EACH ROW DECLARE TEMP\_CON\_NO JABBERWOCKY.CO\_CONTACT.CONT\_ID%TYPE ;

BEGIN

SELECT 'CON'

|| LPAD( TO\_CHAR(CO\_CONTACT\_CONT\_ID\_SEQ.NEXTVAL ),5,'0')

INTO TEMP\_CON\_NO

FROM DUAL;

:NEW.CONT\_ID := TEMP\_CON\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_OFF\_SEQ BEFORE

INSERT ON JABBERWOCKY.OFFICE FOR EACH ROW DECLARE TEMP\_OFF\_NO JABBERWOCKY.OFFICE.OFF\_ID%TYPE ;

BEGIN

SELECT 'OFF'

|| LPAD( TO\_CHAR(OFFICE\_OFF\_ID\_SEQ.NEXTVAL ),5,'0')

INTO TEMP\_OFF\_NO

FROM DUAL;

:NEW.OFF\_ID := TEMP\_OFF\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_REG\_SEQ BEFORE

INSERT ON JABBERWOCKY.REGION FOR EACH ROW DECLARE TEMP\_REG\_NO JABBERWOCKY.REGION.REG\_ID%TYPE ;

BEGIN

SELECT 'REG'

|| LPAD( TO\_CHAR(REGION\_REG\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_REG\_NO

FROM DUAL;

:NEW.REG\_ID := TEMP\_REG\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_NBH\_SEQ BEFORE

INSERT ON JABBERWOCKY.NEIGHBORHOOD FOR EACH ROW DECLARE TEMP\_NBH\_NO JABBERWOCKY.NEIGHBORHOOD.NBHD\_ID%TYPE ;

BEGIN

SELECT 'NBH'

|| LPAD( TO\_CHAR(NEIGHBORHOOD\_NBHD\_ID\_SEQ.NEXTVAL ),3,'0')

INTO TEMP\_NBH\_NO

FROM DUAL;

:NEW.NBHD\_ID := TEMP\_NBH\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_PMN\_SEQ BEFORE

INSERT ON JABBERWOCKY.PAYMENT FOR EACH ROW DECLARE TEMP\_PMN\_NO JABBERWOCKY.PAYMENT.PMNT\_ID%TYPE ;

BEGIN

SELECT 'PMN'

|| LPAD( TO\_CHAR(PAYMENT\_PMNT\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_PMN\_NO

FROM DUAL;

:NEW.PMNT\_ID := TEMP\_PMN\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_RAT\_SEQ BEFORE

INSERT ON JABBERWOCKY.ROOM\_RATE FOR EACH ROW DECLARE TEMP\_RAT\_NO JABBERWOCKY.ROOM\_RATE.RATE\_QUERY\_ID%TYPE ;

BEGIN

SELECT 'RAT'

|| LPAD( TO\_CHAR(ROOM\_RATE\_RATE\_QUERY\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_RAT\_NO

FROM DUAL;

:NEW.RATE\_QUERY\_ID := TEMP\_RAT\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_GEN\_SEQ BEFORE

INSERT ON JABBERWOCKY.GENDER FOR EACH ROW DECLARE TEMP\_GEN\_NO JABBERWOCKY.GENDER.GENDER\_ID%TYPE ;

BEGIN

SELECT 'GEN'

|| LPAD( TO\_CHAR(GENDER\_GENDER\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_GEN\_NO

FROM DUAL;

:NEW.GENDER\_ID := TEMP\_GEN\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_AVA\_SEQ BEFORE

INSERT ON JABBERWOCKY.AVAILABLE\_ROOM FOR EACH ROW DECLARE TEMP\_AVA\_NO JABBERWOCKY.AVAILABLE\_ROOM.AVAIL\_ID%TYPE ;

BEGIN

SELECT 'AVA'

|| LPAD( TO\_CHAR(AVAILABLE\_ROOM\_AVAIL\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_AVA\_NO

FROM DUAL;

:NEW.AVAIL\_ID := TEMP\_AVA\_NO ;

END ;

/

CREATE OR REPLACE TRIGGER TRIG\_REV\_SEQ BEFORE

INSERT ON JABBERWOCKY.REVENUE\_PER\_CITY FOR EACH ROW DECLARE TEMP\_REV\_NO JABBERWOCKY.REVENUE\_PER\_CITY.REVPER\_ID%TYPE ;

BEGIN

SELECT 'REV'

|| LPAD( TO\_CHAR(REVENUE\_PER\_CITY\_REVPER\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_REV\_NO

FROM DUAL;

:NEW.REVPER\_ID := TEMP\_REV\_NO ;

END ;

/

## /\*Table Insert Script\*/

/\*Table Insert Script\*/

INSERT ALL

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000005',

'OFF00002',

'REW000000011',

'GEN000000008',

'N',

'Platinum',

'7882.511',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000001',

'Platinum',

'8',

'Jordan',

'Jones',

'jjones@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000013',

'OFF00001',

'REW000000006',

'GEN000000024',

'N',

'Platinum',

'6887.0272',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000002',

'Platinum',

'21',

'Richard',

'Fox',

'rfox@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000013',

'OFF00003',

'REW000000006',

'GEN000000024',

'N',

'Silver',

'5878.7432',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000003',

'Silver',

'15',

'Juan',

'Garcia',

'jgarcia@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000007',

'OFF00004',

'REW000000009',

'GEN000000024',

'Y',

'Gold',

'4869.863',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000004',

'Gold',

'2',

'Lawrence',

'Medina',

'test@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000010',

'OFF00003',

'REW000000001',

'GEN000000024',

'N',

'Platinum',

'6262.5753',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000005',

'Platinum',

'9',

'Thomas',

'Rodriguez',

'trodriguez@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000010',

'OFF00002',

'REW000000004',

'GEN000000024',

'N',

'Platinum',

'2502.8567',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000006',

'Platinum',

'33',

'Aaron',

'Jones',

'ajones@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000004',

'OFF00001',

'REW000000002',

'GEN000000024',

'N',

'Gold',

'4509.3544',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000007',

'Gold',

'11',

'Alan',

'Lee',

'alee@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000012',

'OFF00005',

'REW000000005',

'GEN000000045',

'N',

'Platinum',

'3424.2898',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000008',

'Platinum',

'26',

'Janet',

'Grant',

'jgrant@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000010',

'OFF00004',

'REW000000002',

'GEN000000024',

'N',

'Gold',

'6777.2697',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000009',

'Platinum',

'34',

'Raymond',

'Myers',

'rmyers@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000008',

'OFF00002',

'REW000000004',

'GEN000000024',

'N',

'Gold',

'7023.7763',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000010',

'Gold',

'21',

'Daniel',

'Ellis',

'dellis@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000010',

'OFF00001',

'REW000000004',

'GEN000000024',

'N',

'Platinum',

'8923.929',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000011',

'Platinum',

'5',

'Nicholas',

'Smith',

'nsmith@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000008',

'OFF00004',

'REW000000002',

'GEN000000024',

'Y',

'Gold',

'6339.1869',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000012',

'Gold',

'34',

'Andrew',

'Williams',

'awilliams@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000014',

'OFF00001',

'REW000000003',

'GEN000000003',

'N',

'Platinum',

'8265.0272',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000013',

'Platinum',

'40',

'Debra',

'Holmes',

'dholmes@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000013',

'OFF00001',

'REW000000005',

'GEN000000024',

'N',

'Platinum',

'2628.3818',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000014',

'Platinum',

'21',

'Christopher',

'Cunningham',

'ccunningham@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000009',

'OFF00002',

'REW000000003',

'GEN000000047',

'N',

'Silver',

'7977.7828',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000015',

'Silver',

'11',

'James',

'Walker',

'jwalker@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000001',

'OFF00001',

'REW000000003',

'GEN000000024',

'N',

'Gold',

'3424.433',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000016',

'Gold',

'15',

'Brandon',

'Shaw',

'bshaw@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000002',

'OFF00006',

'REW000000002',

'GEN000000015',

'N',

'Silver',

'3133.3491',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000017',

'Silver',

'8',

'Jean',

'White',

'jwhite@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000010',

'OFF00006',

'REW000000003',

'GEN000000007',

'N',

'Platinum',

'2906.0811',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000018',

'Platinum',

'37',

'Tammy',

'Garcia',

'tgarcia@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000007',

'OFF00004',

'REW000000003',

'GEN000000025',

'N',

'Platinum',

'5819.5329',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000019',

'Platinum',

'19',

'Scott',

'Anderson',

'sanderson@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000003',

'OFF00005',

'REW000000008',

'GEN000000024',

'Y',

'Gold',

'9619.6852',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000020',

'Gold',

'30',

'Ethan',

'Mitchell',

'emitchell@intuit.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000016',

'OFF00012',

'REW000000004',

'GEN000000015',

'N',

'Gold',

'8873.2232',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000021',

'Gold',

'19',

'Nicole',

'Peters',

'npeters@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000019',

'OFF00011',

'REW000000011',

'GEN000000024',

'N',

'Platinum',

'8590.1743',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000022',

'Platinum',

'7',

'Paul',

'Young',

'pyoung@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000020',

'OFF00007',

'REW000000010',

'GEN000000024',

'N',

'Gold',

'9227.9176',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000023',

'Platinum',

'3',

'Frank',

'Bennett',

'fbennett@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000020',

'OFF00010',

'REW000000001',

'GEN000000024',

'N',

'Gold',

'6247.5452',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000024',

'Platinum',

'13',

'Joshua',

'Chavez',

'jchavez@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000016',

'OFF00009',

'REW000000002',

'GEN000000024',

'N',

'Platinum',

'7014.4249',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000025',

'Platinum',

'32',

'Andrew',

'Peterson',

'apeterson@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000016',

'OFF00008',

'REW000000008',

'GEN000000024',

'N',

'Silver',

'2912.7679',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000026',

'Silver',

'29',

'Samuel',

'Johnson',

'sjohnson@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000019',

'OFF00008',

'REW000000003',

'GEN000000024',

'N',

'Gold',

'4930.378',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000027',

'Platinum',

'29',

'Anthony',

'Harrison',

'aharrison@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000020',

'OFF00009',

'REW000000001',

'GEN000000014',

'N',

'Silver',

'192.165',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000028',

'Silver',

'13',

'Tiffany',

'Perry',

'tperry@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000016',

'OFF00007',

'REW000000010',

'GEN000000015',

'N',

'Platinum',

'4162.5964',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000029',

'Platinum',

'37',

'Emma',

'Sanchez',

'esanchez@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000019',

'OFF00009',

'REW000000004',

'GEN000000024',

'N',

'Platinum',

'6704.3813',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000030',

'Platinum',

'27',

'Joseph',

'Coleman',

'jcoleman@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000020',

'OFF00007',

'REW000000011',

'GEN000000036',

'N',

'Platinum',

'8241.7497',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000031',

'Platinum',

'17',

'Ronald',

'Castro',

'rcastro@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000016',

'OFF00009',

'REW000000008',

'GEN000000015',

'N',

'Gold',

'391.1959',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000032',

'Gold',

'11',

'Andrea',

'Morgan',

'amorgan@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000019',

'OFF00011',

'REW000000001',

'GEN000000013',

'N',

'Platinum',

'4179.5349',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000033',

'Platinum',

'2',

'Eugene',

'Bailey',

'ebailey@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000020',

'OFF00011',

'REW000000003',

'GEN000000015',

'N',

'Platinum',

'6089.9167',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000034',

'Platinum',

'36',

'Nancy',

'Taylor',

'ntaylor@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000020',

'OFF00011',

'REW000000001',

'GEN000000007',

'N',

'Silver',

'8348.1046',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000035',

'Silver',

'39',

'Michelle',

'Daniels',

'mdaniels@sap.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000018',

'OFF00014',

'REW000000002',

'GEN000000024',

'N',

'Silver',

'6992.5406',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000036',

'Silver',

'4',

'Steven',

'Mills',

'smills@microsoft.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000015',

'OFF00013',

'REW000000002',

'GEN000000024',

'Y',

'Platinum',

'6619.724',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000037',

'Platinum',

'16',

'John',

'Young',

'jyoung@microsoft.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000017',

'OFF00013',

'REW000000004',

'GEN000000024',

'N',

'Silver',

'5461.2108',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000038',

'Silver',

'20',

'Vincent',

'Martin',

'vmartin@microsoft.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000017',

'OFF00014',

'REW000000004',

'GEN000000024',

'N',

'Platinum',

'9221.7688',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000039',

'Platinum',

'37',

'David',

'Cook',

'dcook@microsoft.com'

)

INTO JABBERWOCKY.EMPLOYEE

(

DEPT\_ID,

OFF\_ID,

REWARD\_ID,

GENDER\_ID,

BANNED,

CURRENT\_REW\_LVL,

EMP\_POINTS,

EMP\_DOB,

EMP\_PHOTO,

HIGEST\_REW\_LV,

POINTS\_TO\_NXT\_RWD,

EMP\_FIRST,

EMP\_LAST,

EMAIL

)

VALUES

(

'DEP000018',

'OFF00015',

'REW000000002',

'GEN000000015',

'N',

'Platinum',

'9388.0918',

SYSDATE-27375+0,

'C:\EMPLOYEE\_PHOTOS\IMAGE000040',

'Platinum',

'9',

'Lisa',

'Nelson',

'lnelson@microsoft.com'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000001',

SYSDATE-168,

'%+Gdkevmixaz'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000002',

SYSDATE-114,

'%+]gcr~|zcwd'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000003',

SYSDATE-157,

'&+Bzf}pboijo'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000004',

SYSDATE-143,

'cohost'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000005',

SYSDATE-135,

'&\*Mhnnhnactk'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000006',

SYSDATE-96,

'$)9ai|d{qw}o'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000007',

SYSDATE-136,

'#+Lngwkxvvrv'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000008',

SYSDATE-113,

'%(Mzgafstrz{'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000009',

SYSDATE-178,

'#)Gvii{ciphp'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000010',

SYSDATE-157,

'#(\akzhcj|dx'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000011',

SYSDATE-118,

'#(J}qswpovjf'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000012',

SYSDATE-111,

'#(X{nmvvkvol'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000013',

SYSDATE-155,

'$)Gggklbgm~s'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000014',

SYSDATE-100,

'#)Dl~xltlotx'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000015',

SYSDATE-138,

'%(Czn}swqlfz'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000016',

SYSDATE-116,

'%+8{|idrl~oz'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000017',

SYSDATE-103,

'$(Czideb}nbu'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000018',

SYSDATE-103,

'%\*Rck~nlm|lw'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000019',

SYSDATE-158,

'&+Qtvfwjgbff'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000020',

SYSDATE-141,

'&)Zbu{htzyud'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000021',

SYSDATE-132,

'&\*Ebjuse{spm'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000022',

SYSDATE-115,

'$+Otntqvejvk'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000023',

SYSDATE-101,

'#(Wtbyqlk|up'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000024',

SYSDATE-170,

'&\*3tdiu|jupp'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000025',

SYSDATE-175,

'%([l}r~ggtqp'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000026',

SYSDATE-156,

'%+Xxuejtqrmk'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000027',

SYSDATE-117,

'$\*Cwy~cciqwg'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000028',

SYSDATE-137,

'$(\_f~rbu~a~{'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000029',

SYSDATE-175,

'#\*^miv}q~xh~'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000030',

SYSDATE-94,

'$\*Xwq|f}j|pt'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000031',

SYSDATE-125,

'&\*9qf|r~oam{'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000032',

SYSDATE-136,

'#(5moeslkdu~'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000033',

SYSDATE-151,

'%)2xstw}d{{f'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000034',

SYSDATE-100,

'#)Vpn{fyywro'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000035',

SYSDATE-158,

'&+[rxzxfvhm~'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000036',

SYSDATE-106,

'$)Vsznu~yodw'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000037',

SYSDATE-128,

'#+Busofvlzpi'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000038',

SYSDATE-149,

'&(Uhogpegg~n'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000039',

SYSDATE-111,

'%(?htvpyuheh'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000001',

'EMP000000040',

SYSDATE-119,

'%(Onvfgzor{{'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000004',

SYSDATE - 69,

'cohost'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000008',

SYSDATE - 46,

'$)4onbfvrq}m'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000009',

SYSDATE - 74,

'$)Nc~uyvtbod'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000014',

SYSDATE - 70,

'#)>~hvycgfyu'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000019',

SYSDATE - 62,

'$+9ww}jgmqid'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000020',

SYSDATE - 86,

'&\*R~zzlvcd{y'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000025',

SYSDATE - 71,

'%(Dwtzrhwdas'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000028',

SYSDATE - 78,

'&\*;zapa{kdwl'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000033',

SYSDATE - 73,

'&)<gofix{k{|'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000002',

'EMP000000038',

SYSDATE - 81,

'&+5dimgsxvuk'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000003',

'EMP000000004',

SYSDATE - 40,

'cohost'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000003',

'EMP000000008',

SYSDATE - 13,

'#)/go~jr~qlv'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000003',

'EMP000000019',

SYSDATE - 5,

'&(Fpceehphhw'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000003',

'EMP000000020',

SYSDATE - 12,

'#(Immkbujbzl'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000003',

'EMP000000028',

SYSDATE - 14,

'&)9eiogdqoef'

)

INTO JABBERWOCKY.PASSWORD

(

SEQUENCE\_ID,

EMP\_ID,

PASSWORD\_DATE,

USER\_PASSWORD

)

VALUES

(

'SEQ0000003',

'EMP000000038',

SYSDATE - 39,

'#\*Wmqo|mtonj'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Points',

'Marriott Rewards'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Points',

'Wyndham Rewards'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Points',

'Best Western Rewards'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Points',

'Club Carlson'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Points',

'IHG Rewards Club'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Miles',

'Alaska Airlines Mileage PLan'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Miles',

'American Airlines AAdvantage'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Miles',

'Southwest Rapid Rewards'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Miles',

'JetBlue TrueBlue'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Miles',

'HawaiianMiles'

)

INTO JABBERWOCKY.REWARD\_SYSTEM

(

REWARD\_TYPE,

REWARD\_NAME

)

VALUES

(

'Cash',

'Cash'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Administration',

900

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Sales',

673.2

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Finance and Operations',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Coporate Strategy and Development',

900

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Human Resources',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Information Technology',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Legal',

1054.8

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Marketing',

1054.8

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Product Management',

1530

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Customer Service and Support',

1306.8

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Process Excellence',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Design and User Experienece',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Project and Program Management',

673.2

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00001',

'Software Engineering',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00004',

'Sales',

502.2

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00002',

'Human Resources',

2160

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00004',

'Marketing',

0

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00004',

'Process Excellence',

2019.6

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00002',

'Project and Program Management',

2088

)

INTO JABBERWOCKY.DEPARTMENT

(

CO\_ID,

DEPT\_NAME,

TOT\_DEPT\_SPEND

)

VALUES

(

'CO\_00002',

'Software Engineering',

2426.4

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000007',

'PMN000000001',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000012',

'PMN000000001',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000038',

'PMN000000002',

SYSDATE,

'300'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000040',

'PMN000000002',

SYSDATE,

'900'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000025',

'PMN000000004',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000026',

'PMN000000004',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000009',

'PMN000000005',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000017',

'PMN000000005',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000015',

'PMN000000006',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000017',

'PMN000000006',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000027',

'PMN000000007',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000028',

'PMN000000007',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000040',

'PMN000000008',

SYSDATE,

'600'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000038',

'PMN000000008',

SYSDATE,

'200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000012',

'PMN000000010',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000010',

'PMN000000010',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000037',

'PMN000000011',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000040',

'PMN000000011',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000026',

'PMN000000012',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000022',

'PMN000000012',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000008',

'PMN000000014',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000012',

'PMN000000014',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000012',

'PMN000000015',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000007',

'PMN000000015',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000028',

'PMN000000016',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000026',

'PMN000000016',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000016',

'PMN000000017',

SYSDATE,

'600'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000017',

'PMN000000017',

SYSDATE,

'300'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000021',

'PMN000000018',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000029',

'PMN000000018',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000011',

'PMN000000020',

SYSDATE,

'900'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000009',

'PMN000000020',

SYSDATE,

'600'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000030',

'PMN000000021',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000023',

'PMN000000021',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000035',

'PMN000000022',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000023',

'PMN000000022',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000014',

'PMN000000023',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000019',

'PMN000000023',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000020',

'PMN000000024',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000017',

'PMN000000024',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000027',

'PMN000000025',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000022',

'PMN000000025',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000036',

'PMN000000026',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000038',

'PMN000000026',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000004',

'PMN000000027',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000017',

'PMN000000027',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000011',

'PMN000000028',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000029',

'PMN000000028',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000034',

'PMN000000029',

SYSDATE,

'1200'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000026',

'PMN000000029',

SYSDATE,

'400'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000004',

'PMN000000030',

SYSDATE,

'800'

)

INTO JABBERWOCKY.REWARD\_LOG

(

EMP\_ID,

PMNT\_ID,

REWARD\_DATE,

REWARD\_AMT

)

VALUES

(

'EMP000000002',

'PMN000000030',

SYSDATE,

'1200'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000004',

'57'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000007',

'65'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000008',

'27'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000009',

'98'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000011',

'22'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000012',

'83'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000014',

'48'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000015',

'94'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000016',

'47'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000020',

'89'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000021',

'85'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000025',

'28'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000026',

'47'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000027',

'52'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000028',

'67'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000030',

'21'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000032',

'36'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000034',

'67'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000035',

'3'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000036',

'26'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000037',

'78'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000038',

'94'

)

INTO JABBERWOCKY.TRAVELER

(

TRAVELER\_ID,

TRAVELER\_RATING

)

VALUES

(

'EMP000000040',

'23'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000002',

'76'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000004',

'81'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000005',

'13'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000007',

'91'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000009',

'55'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000010',

'10'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000011',

'79'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000012',

'5'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000016',

'15'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000017',

'49'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000019',

'68'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000022',

'81'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000023',

'9'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000026',

'10'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000028',

'29'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000029',

'63'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000038',

'84'

)

INTO JABBERWOCKY.HOST

(

HOST\_ID,

HOST\_RATING

)

VALUES

(

'EMP000000040',

'42'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000007',

'ROO000000010',

'OFF00004',

'PMN000000001',

'DEP000008',

'08-JUN-15',

'12-JUN-15',

'550.8',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000037',

'ROO000000022',

'OFF00015',

'PMN000000002',

'DEP000018',

'09-JUN-15',

'12-JUN-15',

'504.9',

'Y'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000038',

'ROO000000022',

'OFF00015',

'PMN000000003',

'DEP000018',

'09-JUN-15',

'12-JUN-15',

'504.9',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000025',

'ROO000000015',

'OFF00008',

'PMN000000004',

'DEP000016',

'15-JUN-15',

'19-JUN-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000009',

'ROO000000013',

'OFF00006',

'PMN000000005',

'DEP000002',

'15-JUN-15',

'19-JUN-15',

'673.2',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000015',

'ROO000000023',

'OFF00004',

'PMN000000006',

'DEP000007',

'06-JUL-15',

'10-JUL-15',

'550.8',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000027',

'ROO000000018',

'OFF00009',

'PMN000000007',

'DEP000020',

'06-JUL-15',

'10-JUL-15',

'673.2',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000040',

'ROO000000021',

'OFF00013',

'PMN000000008',

'DEP000015',

'15-JUL-15',

'17-JUL-15',

'167.4',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000032',

'ROO000000015',

'OFF00008',

'PMN000000009',

'DEP000016',

'13-JUL-15',

'17-JUL-15',

'540',

'Y'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000012',

'ROO000000005',

'OFF00001',

'PMN000000010',

'DEP000004',

'13-JUL-15',

'17-JUL-15',

'900',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000037',

'ROO000000022',

'OFF00015',

'PMN000000011',

'DEP000018',

'20-JUL-15',

'24-JUL-15',

'673.2',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000026',

'ROO000000020',

'OFF00011',

'PMN000000012',

'DEP000019',

'20-JUL-15',

'24-JUL-15',

'504',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000038',

'ROO000000022',

'OFF00015',

'PMN000000013',

'DEP000018',

'27-JUL-15',

'29-JUL-15',

'336.6',

'Y'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000008',

'ROO000000011',

'OFF00004',

'PMN000000014',

'DEP000010',

'27-JUL-15',

'31-JUL-15',

'550.8',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000012',

'ROO000000003',

'OFF00001',

'PMN000000015',

'DEP000001',

'27-JUL-15',

'31-JUL-15',

'900',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000028',

'ROO000000015',

'OFF00008',

'PMN000000016',

'DEP000019',

'10-AUG-15',

'14-AUG-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000016',

'ROO000000023',

'OFF00004',

'PMN000000017',

'DEP000010',

'11-AUG-15',

'14-AUG-15',

'378',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000021',

'ROO000000017',

'OFF00007',

'PMN000000018',

'DEP000016',

'10-AUG-15',

'14-AUG-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000027',

'ROO000000019',

'OFF00009',

'PMN000000019',

'DEP000020',

'10-AUG-15',

'14-AUG-15',

'673.2',

'Y'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000011',

'ROO000000009',

'OFF00004',

'PMN000000020',

'DEP000010',

'18-AUG-15',

'21-AUG-15',

'378',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000030',

'ROO000000016',

'OFF00007',

'PMN000000021',

'DEP000020',

'24-AUG-15',

'28-AUG-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000035',

'ROO000000016',

'OFF00007',

'PMN000000022',

'DEP000020',

'14-SEP-15',

'18-SEP-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000014',

'ROO000000012',

'OFF00004',

'PMN000000023',

'DEP000008',

'21-SEP-15',

'25-SEP-15',

'504',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000020',

'ROO000000023',

'OFF00004',

'PMN000000024',

'DEP000007',

'21-SEP-15',

'25-SEP-15',

'504',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000027',

'ROO000000020',

'OFF00011',

'PMN000000025',

'DEP000019',

'19-OCT-15',

'23-OCT-15',

'504',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000036',

'ROO000000021',

'OFF00013',

'PMN000000026',

'DEP000015',

'26-OCT-15',

'30-OCT-15',

'334.8',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000004',

'ROO000000013',

'OFF00003',

'PMN000000027',

'DEP000013',

'02-NOV-15',

'06-NOV-15',

'673.2',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000025',

'ROO000000017',

'OFF00007',

'PMN000000028',

'DEP000016',

'16-NOV-15',

'20-NOV-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000034',

'ROO000000015',

'OFF00008',

'PMN000000029',

'DEP000019',

'07-DEC-15',

'11-DEC-15',

'540',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000004',

'ROO000000001',

'OFF00002',

'PMN000000030',

'DEP000009',

'07-DEC-15',

'11-DEC-15',

'900',

'N'

)

INTO JABBERWOCKY.RESERVATION

(

TRAVELER\_ID,

ROOM\_ID,

OFF\_ID,

PMNT\_ID,

DEPT\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ISCANC

)

VALUES

(

'EMP000000007',

'ROO000000008',

'OFF00002',

'PMN000000031',

'DEP000009',

'15-DEC-15',

'20-DEC-15',

'630',

'N'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.CANCELLATION

(

CANC\_ID,

RESERV\_ID,

CANC\_LIMIT,

CANC\_REASON,

CANC\_DATE,

REF\_AMT

)

VALUES

(

'CAN000000001',

'RES000000002',

'08-JUN-15',

'Other',

'06-JUN-15',

'504.9'

)

INTO JABBERWOCKY.CANCELLATION

(

CANC\_ID,

RESERV\_ID,

CANC\_LIMIT,

CANC\_REASON,

CANC\_DATE,

REF\_AMT

)

VALUES

(

'CAN000000002',

'RES000000009',

'12-JUL-15',

'Other',

'10-JUL-15',

'540'

)

INTO JABBERWOCKY.CANCELLATION

(

CANC\_ID,

RESERV\_ID,

CANC\_LIMIT,

CANC\_REASON,

CANC\_DATE,

REF\_AMT

)

VALUES

(

'CAN000000003',

'RES000000013',

'26-JUL-15',

'Other',

'26-JUL-15',

'336.6'

)

INTO JABBERWOCKY.CANCELLATION

(

CANC\_ID,

RESERV\_ID,

CANC\_LIMIT,

CANC\_REASON,

CANC\_DATE,

REF\_AMT

)

VALUES

(

'CAN000000004',

'RES000000019',

'09-AUG-15',

'Other',

'08-AUG-15',

'673.2'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000001',

'90',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000001',

'90',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000002',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000002',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000003',

'120',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000004',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000005',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000006',

'120',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000007',

'140',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000008',

'140',

'2',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000008',

'168',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000009',

'120',

'2',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000010',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000011',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000012',

'140',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000013',

'168',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000014',

'192',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000015',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000015',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000016',

'120',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000017',

'192',

'2',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000018',

'100',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000019',

'225',

'1',

'Y',

'-'

)

INTO JABBERWOCKY.ROOM

(

ADDRS\_ID,

SQR\_FEET,

BED\_COUNT,

PRIVATE\_BTHRM,

ROOM\_COMMENTS

)

VALUES

(

'ADD000000019',

'140',

'1',

'Y',

'-'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000001',

'EMP000000002',

'NBH001',

'523 Natoma St',

'San Francisco',

'San Francisco',

'CA',

'94103',

'USA',

'37.779747',

'-122.408323',

'-',

'N',

'Y',

'N',

'N',

'N',

'Y',

'Y',

'House',

'N',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000002',

'EMP000000007',

'NBH002',

'904 Steiner St',

'San Francisco',

'San Francisco',

'CA',

'94117',

'USA',

'37.777812',

'-122.432973',

'-',

'Y',

'N',

'N',

'N',

'Y',

'Y',

'Y',

'House',

'Y',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000003',

'EMP000000010',

'NBH001',

'70 S Park St',

'San Francisco',

'San Francisco',

'CA',

'94107',

'USA',

'37.782027',

'-122.393959',

'-',

'Y',

'N',

'N',

'Y',

'N',

'Y',

'Y',

'House',

'Y',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000004',

'EMP000000011',

'NBH003',

'235 Andover St',

'San Francisco',

'San Francisco',

'CA',

'94110',

'USA',

'37.739959',

'-122.41627',

'-',

'N',

'N',

'N',

'Y',

'N',

'Y',

'Y',

'House',

'Y',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000005',

'EMP000000016',

'NBH003',

'3763 Folsom St',

'San Francisco',

'San Francisco',

'CA',

'94110',

'USA',

'37.73952',

'-122.413197',

'-',

'Y',

'Y',

'Y',

'N',

'N',

'Y',

'N',

'House',

'N',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000006',

'EMP000000004',

'NBH004',

'11468 Vela Dr',

'San Diego',

'San Diego',

'CA',

'92126',

'USA',

'32.933534',

'-117.137635',

'-',

'N',

'N',

'N',

'Y',

'Y',

'N',

'Y',

'House',

'N',

'Y',

'Y',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000007',

'EMP000000009',

'NBH004',

'7521 Flower Meadow DR',

'San Diego',

'San Diego',

'CA',

'92126',

'USA',

'32.92734',

'-117.160658',

'-',

'N',

'N',

'Y',

'N',

'Y',

'N',

'N',

'House',

'N',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000008',

'EMP000000012',

'NBH005',

'13191 Carolee Ave',

'San Diego',

'San Diego',

'CA',

'92129',

'USA',

'32.959835',

'-117.128803',

'-',

'N',

'N',

'Y',

'Y',

'Y',

'Y',

'Y',

'House',

'N',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000009',

'EMP000000019',

'NBH005',

'7665 Via Cristal',

'San Diego',

'San Diego',

'CA',

'92129',

'USA',

'32.963274',

'-117.158167',

'-',

'N',

'N',

'Y',

'Y',

'Y',

'Y',

'N',

'Condo',

'Y',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000010',

'EMP000000017',

'NBH006',

'2302 Webster St',

'Palo Alto',

'Santa Clara',

'CA',

'94301',

'USA',

'37.435113',

'-122.135937',

'-',

'Y',

'Y',

'Y',

'N',

'Y',

'Y',

'Y',

'House',

'Y',

'Y',

'Y',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000011',

'EMP000000005',

'NBH007',

'2077 Valparaiso Ave',

'Menlo Park',

'San Mateo',

'CA',

'94025',

'USA',

'37.434314',

'-122.205344',

'-',

'N',

'Y',

'N',

'N',

'N',

'Y',

'Y',

'House',

'N',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000012',

'EMP000000026',

'NBH008',

'11982 Mayfield Ave',

'Los Angeles',

'Los Angeles',

'CA',

'90049',

'USA',

'34.047401',

'-118.468346',

'-',

'N',

'Y',

'N',

'N',

'N',

'N',

'Y',

'House',

'N',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000013',

'EMP000000023',

'NBH009',

'3082 Molokai Pl',

'Costa Mesa',

'Orange',

'CA',

'92626',

'USA',

'33.682575',

'-117.926591',

'-',

'N',

'N',

'Y',

'N',

'N',

'Y',

'Y',

'House',

'Y',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000014',

'EMP000000029',

'NBH010',

'431 Curie Ave',

'Santa Ana',

'Orange',

'CA',

'92707',

'USA',

'33.702121',

'-117.873078',

'-',

'N',

'N',

'N',

'Y',

'Y',

'Y',

'Y',

'House',

'Y',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000015',

'EMP000000028',

'NBH006',

'2200 Byron St',

'Palo Alto',

'Santa Clara',

'CA',

'94301',

'USA',

'37.436393',

'-122.136076',

'-',

'N',

'N',

'N',

'N',

'Y',

'N',

'N',

'House',

'Y',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000016',

'EMP000000022',

'NBH011',

'2163 Alexander Way',

'Pleasanton',

'Alameda',

'CA',

'94588',

'USA',

'37.686648',

'-121.87452',

'-',

'Y',

'Y',

'N',

'N',

'Y',

'Y',

'N',

'House',

'Y',

'N',

'Y',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000017',

'EMP000000038',

'NBH012',

'4460 Cavitt Stallman Rd',

'Granite Bay',

'Placer',

'CA',

'95746',

'USA',

'38.758547',

'-121.22011',

'-',

'N',

'Y',

'N',

'Y',

'Y',

'N',

'Y',

'House',

'N',

'N',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000018',

'EMP000000040',

'NBH013',

'2452 Elka Ave',

'Mountain View',

'Santa Clara',

'CA',

'94043',

'USA',

'37.410328',

'-122.100277',

'-',

'N',

'Y',

'Y',

'Y',

'Y',

'N',

'Y',

'House',

'Y',

'Y',

'N',

'-'

)

INTO JABBERWOCKY.ADDRESS

(

ADDRS\_ID,

HOST\_ID,

NBHD\_ID,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_COUNTY,

ADDRS\_STATE,

ADDRS\_ZIP,

ADDRS\_COUNTRY,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_RMS,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER

)

VALUES

(

'ADD000000019',

'EMP000000017',

'NBH014',

'377 13th St',

'Del Mar',

'San Diego',

'CA',

'92014',

'USA',

'32.958045',

'-117.263052',

'-',

'N',

'N',

'N',

'N',

'Y',

'N',

'Y',

'House',

'N',

'Y',

'Y',

'-'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Intuit',

8092.8

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'SAP',

6674.4

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Mattress Firm',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Microsoft ',

2521.8

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Oracle ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Symantec ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Apple Inc ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Hewlett-Packard ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Dell ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Intel ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Cisco Systems ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'NCR ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'IBM ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Xerox ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'CSC ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Amazon.com ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Google ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'eBay ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Yahoo! ',

0

)

INTO JABBERWOCKY.COMPANY

(

CO\_NAME,

TOTAL\_CO\_SPEND

)

VALUES

(

'Facebook',

0

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00001',

'Mark',

'Wright',

SYSDATE - 2456,

(SYSDATE - 2456) + 515,

'mwright@intuit.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00002',

'Jonathan',

'Barnes',

SYSDATE - 2908,

NULL,

'jbarnes@sap.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00003',

'Aaron',

'Gonzalez',

SYSDATE - 4550,

NULL,

'agonzalez@mattressfirm.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00004',

'Willie',

'James',

SYSDATE - 4787,

NULL,

'wjames@microsoft.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00005',

'Zachary',

'Long',

SYSDATE - 1863,

NULL,

'zlong@oracle.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00006',

'Nicholas',

'Jackson',

SYSDATE - 4425,

(SYSDATE - 4425) + 725,

'njackson@symantec.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00007',

'Jack',

'Holmes',

SYSDATE - 3531,

NULL,

'jholmes@apple.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00008',

'Eugene',

'Ramirez',

SYSDATE - 4161,

NULL,

'eramirez@hp.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00009',

'Kelly',

'Porter',

SYSDATE - 2130,

(SYSDATE - 2130) + 36,

'kporter@dell.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00010',

'Ralph',

'Thompson',

SYSDATE - 3378,

(SYSDATE - 3378) + 168,

'rthompson@intel.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00011',

'Joseph',

'Payne',

SYSDATE - 4508,

NULL,

'jpayne@cisco.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00012',

'Michael',

'Mitchell',

SYSDATE - 3210,

NULL,

'mmitchell@ncr.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00013',

'Olivia',

'Stevens',

SYSDATE - 2205,

NULL,

'ostevens@ibm.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00014',

'Debra',

'Tucker',

SYSDATE - 4290,

(SYSDATE - 4290) + 742,

'dtucker@xerox.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00015',

'Jean',

'Tucker',

SYSDATE - 2414,

NULL,

'jtucker@csc.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00016',

'Doris',

'Kelly',

SYSDATE - 4690,

NULL,

'dkelly@amazon.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00017',

'Christina',

'Carter',

SYSDATE - 3249,

NULL,

'ccarter@google.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00018',

'Beverly',

'Hicks',

SYSDATE - 2064,

(SYSDATE - 2064) + 434,

'bhicks@ebay.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00019',

'Matthew',

'Ford',

SYSDATE - 1571,

NULL,

'mford@yahoo.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00020',

'Wayne',

'Phillips',

SYSDATE - 4504,

NULL,

'wphillips@facebook.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00003',

'Rachel',

'Sullivan',

((SYSDATE - 2456) + 655) + 1,

NULL,

'rsullivan@mattressfirm.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00006',

'Terry',

'Gordon',

((SYSDATE - 4787) + 522) + 1,

NULL,

'tgordon@symantec.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00008',

'Timothy',

'Jenkins',

((SYSDATE - 3531) + 708) + 1,

(((SYSDATE - 3531) + 708) + 1) + 119,

'tjenkins@hp.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00011',

'Martha',

'Schmidt',

((SYSDATE - 3378) + 97) + 1,

NULL,

'mschmidt@cisco.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00015',

'Joan',

'Patel',

((SYSDATE - 4290) + 220) + 1,

(((SYSDATE - 4290) + 220) + 1) + 253,

'jpatel@csc.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00019',

'Jose',

'Griffin',

((SYSDATE - 4504) + 713) + 1,

NULL,

'jgriffin@yahoo.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00006',

'Jacqueline',

'Simmons',

((((SYSDATE - 4787) + 522) + 1) + 443) + 1,

NULL,

'jsimmons@symantec.com'

)

INTO JABBERWOCKY.CO\_CONTACT

(

CO\_ID,

CONT\_FIRST,

CONT\_LAST,

CONT\_BEGIN\_DATE,

CONT\_END\_DATE,

CONT\_EMAIL

)

VALUES

(

'CO\_00015',

'Patricia',

'Elliott',

((((SYSDATE - 4290) + 220) + 1) + 216) + 1,

NULL,

'pelliott@csc.com'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00001',

'REG000000027',

'71 Stevenson St, Suite 900',

'San Francisco',

'CA',

'94105',

'USA',

'37.789383',

'-122.399754'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00001',

'REG000000027',

'22 4th Street, 12th Floor',

'San Francisco',

'CA',

'94103',

'USA',

'37.784907',

'-122.405555'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00001',

'REG000000030',

'1800 Jefferson Dr',

'Menlo Park',

'CA',

'94025',

'USA',

'37.481349',

'-122.170637'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00001',

'REG000000026',

'7535 Torrey Santa Fe Rd',

'San Diego',

'CA',

'92129',

'USA',

'32.959522',

'-117.158163'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00001',

'REG000000012',

'21215 Burbank Blvd, Ste. 100',

'Woodland Hills',

'CA',

'91367',

'USA',

'34.173832',

'-118.593084'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00001',

'REG000000032',

'2632 Marine Way',

'Mountain View',

'CA',

'94043',

'USA',

'37.428289',

'-122.097631'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00002',

'REG000000021',

'18101 Von Karman Ave, Suite 900',

'Irvine',

'CA',

'92612',

'USA',

'33.679957',

'-117.853416'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00002',

'REG000000012',

'1901 Avenue of the Stars, Suite 1455',

'Los Angeles',

'CA',

'90067',

'USA',

'34.05952',

'-118.417793'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00002',

'REG000000032',

'3410 Hillview Ave',

'Palo Alto',

'CA',

'94304',

'USA',

'37.398893',

'-122.146425'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00002',

'REG000000030',

'1 Tower Place, Centennial Towers, Suite 1100',

'South San Francisco',

'CA',

'94080',

'USA',

'37.666392',

'-122.397917'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00002',

'REG000000001',

'1 And 3 Sybase Dr, Bldg A and B',

'Dublin',

'CA',

'94568',

'USA',

'37.70675',

'-121.890043'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00002',

'REG000000032',

'910 Hermosa Court',

'Sunnyvale',

'CA',

'94085',

'USA',

'37.386984',

'-122.04074'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00004',

'REG000000024',

'1415 L Street, Suite 200',

'Sacramento',

'CA',

'95814',

'USA',

'38.576767',

'-121.488272'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00004',

'REG000000027',

'555 California, Suite 200',

'San Francisco',

'CA',

'94104',

'USA',

'37.791875',

'-122.403811'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00004',

'REG000000032',

'1065 La Avenida',

'Mountain View',

'CA',

'94043',

'USA',

'37.411706',

'-122.07136'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000012',

'440 E Huntington Dr, Suite 400',

'Arcadia',

'CA',

'91006',

'USA',

'34.13912',

'-118.022066'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000012',

'5750 Hannum Ave, Suite 200',

'Culver City',

'CA',

'90230',

'USA',

'33.986582',

'-118.383807'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000012',

'200 N Sepulveda Blvd, Suite 400',

'El Segundo',

'CA',

'90245',

'USA',

'33.917769',

'-118.395219'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000012',

'15760 Ventura Blvd, Suite 1400',

'Encino',

'CA',

'91436',

'USA',

'34.154764',

'-118.476754'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000022',

'1001 Sunset Blvd',

'Rocklin',

'CA',

'95765',

'USA',

'38.816959',

'-121.286074'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000030',

'1100 Grundy Ln, Suite 300',

'San Bruno',

'CA',

'94006',

'USA',

'37.629999',

'-122.422721'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000026',

'Eastgate Technology Park, 9515 Towne Center Dr',

'San Diego',

'CA',

'92121',

'USA',

'32.880065',

'-117.207086'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000026',

'Eastgate Technology Park, 9525 Towne Center Dr',

'San Diego',

'CA',

'92121',

'USA',

'32.880912',

'-117.207205'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000030',

'500 Oracle Parkway',

'Redwood Shores',

'CA',

'94065',

'USA',

'37.530613',

'-122.262446'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000021',

'17901 Von Karman Ave, Suite 800',

'Irvine',

'CA',

'92614',

'USA',

'33.683179',

'-117.849688'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000035',

'775 Baywood Dr, Suite 390',

'Petaluma',

'CA',

'94954',

'USA',

'38.23154',

'-122.61454'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000027',

'475 Sansome St, 15th Floor',

'San Francisco',

'CA',

'94111',

'USA',

'37.794658',

'-122.401892'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000027',

'717 Market Street, 7th Floor',

'San Francisco',

'CA',

'94103',

'USA',

'37.786836',

'-122.403707'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000032',

'488 Almaden Blvd',

'San Jose',

'CA',

'95110',

'USA',

'37.327121',

'-121.889251'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000032',

'4230 Leonard Stocking Dr',

'Santa Clara',

'CA',

'95054',

'USA',

'37.392877',

'-121.952732'

)

INTO JABBERWOCKY.OFFICE

(

CO\_ID,

REG\_ID,

OFF\_STREET,

OFF\_CITY,

OFF\_STATE,

OFF\_ZIP,

OFF\_COUNTRY,

OFF\_LATITUDE,

OFF\_LONGITUDE

)

VALUES

(

'CO\_00005',

'REG000000001',

'5805 Owens Dr',

'Pleasanton',

'CA',

'94588',

'USA',

'37.699644',

'-121.891357'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Standard Rate ',

'Applies for all locations without specified rates ',

'89',

'89',

'89',

'89',

'89',

'89',

'89',

'89',

'89',

'89',

'89',

'89'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Antioch / Brentwood / Concord ',

'Contra Costa ',

'132',

'132',

'132',

'132',

'132',

'132',

'132',

'132',

'132',

'132',

'132',

'132'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Bakersfield / Ridgecrest ',

'Kern ',

'95',

'95',

'95',

'95',

'95',

'95',

'95',

'95',

'95',

'95',

'95',

'95'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Barstow / Ontario / Victorville ',

'San Bernardino ',

'96',

'96',

'96',

'96',

'96',

'96',

'96',

'96',

'96',

'96',

'96',

'96'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Death Valley ',

'Inyo ',

'102',

'102',

'102',

'102',

'102',

'102',

'102',

'102',

'102',

'102',

'102',

'102'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Eureka / Arcata / McKinleyville ',

'Humboldt ',

'94',

'94',

'94',

'94',

'94',

'94',

'94',

'94',

'112',

'112',

'112',

'94'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Fresno ',

'Fresno ',

'92',

'92',

'92',

'92',

'92',

'92',

'92',

'92',

'92',

'92',

'92',

'92'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Los Angeles ',

'Los Angeles / Orange / Ventura / Edwards AFB less the city of Santa Monica ',

'150',

'150',

'150',

'157',

'157',

'157',

'150',

'150',

'150',

'150',

'150',

'150'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Mammoth Lakes ',

'Mono ',

'116',

'116',

'159',

'159',

'159',

'116',

'116',

'116',

'116',

'116',

'116',

'116'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Mill Valley / San Rafael / Novato ',

'Marin ',

'146',

'130',

'130',

'146',

'146',

'146',

'146',

'146',

'146',

'146',

'146',

'146'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Monterey ',

'Monterey ',

'134',

'134',

'134',

'134',

'134',

'134',

'134',

'134',

'134',

'175',

'175',

'134'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Napa ',

'Napa ',

'191',

'149',

'149',

'149',

'149',

'149',

'149',

'191',

'191',

'191',

'191',

'191'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Oakhurst ',

'Madera ',

'91',

'91',

'91',

'91',

'91',

'91',

'91',

'91',

'115',

'115',

'115',

'91'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Oakland ',

'Alameda ',

'140',

'140',

'140',

'140',

'140',

'140',

'140',

'140',

'140',

'140',

'140',

'140'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Palm Springs ',

'Riverside ',

'123',

'123',

'123',

'123',

'123',

'123',

'123',

'123',

'92',

'92',

'92',

'123'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Point Arena / Gualala ',

'Mendocino ',

'106',

'106',

'106',

'106',

'106',

'106',

'106',

'106',

'106',

'106',

'106',

'106'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Redding ',

'Shasta ',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Sacramento ',

'Sacramento ',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'112'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'San Diego ',

'San Diego ',

'140',

'140',

'140',

'153',

'153',

'153',

'153',

'153',

'153',

'153',

'140',

'140'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'San Francisco ',

'San Francisco ',

'250',

'250',

'250',

'250',

'250',

'250',

'250',

'250',

'250',

'250',

'250',

'250'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'San Luis Obispo ',

'San Luis Obispo ',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'112',

'133',

'133',

'133',

'112'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'San Mateo / Foster City / Belmont ',

'San Mateo ',

'178',

'178',

'178',

'184',

'184',

'184',

'178',

'178',

'178',

'178',

'178',

'178'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Santa Barbara ',

'Santa Barbara ',

'161',

'161',

'161',

'161',

'161',

'161',

'161',

'161',

'161',

'211',

'211',

'161'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Santa Cruz ',

'Santa Cruz ',

'129',

'129',

'129',

'129',

'129',

'129',

'129',

'129',

'164',

'164',

'164',

'129'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Santa Monica ',

'City limits of Santa Monica ',

'217',

'217',

'217',

'217',

'217',

'217',

'217',

'217',

'252',

'252',

'252',

'217'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Santa Rosa ',

'Sonoma ',

'134',

'118',

'118',

'118',

'118',

'134',

'134',

'134',

'134',

'134',

'134',

'134'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'South Lake Tahoe ',

'El Dorado ',

'111',

'111',

'136',

'136',

'136',

'136',

'136',

'136',

'136',

'162',

'162',

'111'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Stockton ',

'San Joaquin ',

'97',

'97',

'97',

'97',

'97',

'97',

'97',

'97',

'97',

'97',

'97',

'97'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Sunnyvale / Palo Alto / San Jose ',

'Santa Clara ',

'187',

'187',

'187',

'187',

'187',

'187',

'175',

'175',

'187',

'187',

'187',

'187'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Tahoe City ',

'Placer ',

'93',

'93',

'93',

'93',

'93',

'93',

'93',

'93',

'93',

'93',

'93',

'93'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Truckee ',

'Nevada ',

'120',

'120',

'120',

'120',

'120',

'120',

'120',

'120',

'120',

'120',

'120',

'120'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Visalia / Lemoore ',

'Tulare / Kings ',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90',

'90'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'West Sacramento / Davis ',

'Yolo ',

'113',

'113',

'113',

'113',

'113',

'113',

'113',

'113',

'113',

'113',

'113',

'113'

)

INTO JABBERWOCKY.REGION\_RAW

(

PRIMARY\_DEST,

COUNTY,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

)

VALUES

(

'Yosemite National Park ',

'Mariposa ',

'118',

'118',

'118',

'107',

'107',

'107',

'107',

'107',

'132',

'132',

'132',

'118'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT INTO REGION

(COUNTY

)

SELECT COUNTY

FROM

(SELECT PRIMARY\_DEST,

RTRIM(LTRIM(regexp\_substr(REPLACE(COUNTY,'/',','),'[^,]+', 1, level))) AS COUNTY

FROM REGION\_RAW

CONNECT BY RTRIM(LTRIM(regexp\_substr(REPLACE(COUNTY,'/',','),'[^,]+', 1, level))) IS NOT NULL

)

GROUP BY COUNTY

ORDER BY COUNTY;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'South of Market, San Francisco',

'Urban',

'Hot spot for tech companies, several museums and other cultural centers, close to the water'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Western Addition, San Francisco',

'Urban',

'Close to parks and dining, beautiful city views'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Bernal Heights, San Francisco',

'Suburban',

'Lots of parks, mostly residential, family friendly'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Mira Mesa, San Diego',

'Suburban',

'Relatively young and ethnically diverse neighborhood, plenty of shopping and recreational options, large variety of food options'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Rancho Penasquitos, San Diego',

'Suburban',

'Plenty of open space, close to hiking, biking, and equestrian trails'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Old Palo Alto, Palo Alto',

'Suburban',

'Close to Stanford, tree-lined streets, lively nightlife'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'West Menllo Park',

'Suburban',

'Quiet, tree-lined streets, close to golf course and other outdoor activities'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Sawtelle, Los Angeles',

'Urban',

'Close to UCLA, lively nightlife, close to Beverly Hills'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Mesa Verde, Costa Mesa',

'Suburban',

'Close to golf course, close to Newport Beach'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Sandpointe, Santa Ana',

'Suburban',

'Close to shopping, easy freeway access'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Asco, Pleasanton',

'Suburban',

'Close to outlets and shopping, close to sports park'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Granite Bay',

'Rural',

'Quiet country-style living, wide open spaces but still close to civilization, close to Folsom Lake'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Monta Loma',

'Suburban',

'Close to the Caltrain and shopping'

)

INTO JABBERWOCKY.NEIGHBORHOOD

(

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT

)

VALUES

(

'Del Mar',

'Suburban',

'Close to the beach, upscale dining, close to hiking trails'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000008',

'05-JUN-15',

'550.8'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000018',

'05-JUN-15',

'504.9'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000018',

'06-JUN-15',

'504.9'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000016',

'10-JUN-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000002',

'12-JUN-15',

'673.2'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000007',

'01-JUL-15',

'550.8'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000020',

'02-JUL-15',

'673.2'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000015',

'02-JUL-15',

'167.4'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000016',

'06-JUL-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000004',

'07-JUL-15',

'900'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000018',

'20-JUL-15',

'673.2'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000019',

'20-JUL-15',

'504'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000018',

'22-JUL-15',

'336.6'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000010',

'23-JUL-15',

'550.8'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000001',

'23-JUL-15',

'900'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000019',

'27-JUL-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000010',

'03-AUG-15',

'378'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000016',

'03-AUG-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000020',

'03-AUG-15',

'673.2'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000010',

'07-AUG-15',

'378'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000020',

'10-AUG-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000020',

'31-AUG-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000008',

'08-SEP-15',

'504'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000007',

'08-SEP-15',

'504'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000019',

'02-OCT-15',

'504'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000015',

'12-OCT-15',

'334.8'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000013',

'26-OCT-15',

'673.2'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000016',

'02-NOV-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000019',

'20-NOV-15',

'540'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000009',

'20-NOV-15',

'900'

)

INTO JABBERWOCKY.PAYMENT

(

DEPT\_ID,

PMNT\_DATE,

PMNT\_PRICE

)

VALUES

(

'DEP000009',

'11-DEC-15',

'630'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000001',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000002',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000003',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000004',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000005',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000006',

'Full'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000007',

'Full'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000008',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000009',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000010',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B2',

'ROO000000010',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000011',

'King'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000012',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B2',

'ROO000000012',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000013',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000014',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000015',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000016',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000017',

'King'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000018',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000019',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000020',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000021',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B2',

'ROO000000021',

'Queen'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000022',

'Twin'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000023',

'Cal King'

)

INTO JABBERWOCKY.ROOM\_BEDS

(

BED\_NO,

ROOM\_ID,

BED\_SIZE

)

VALUES

(

'B1',

'ROO000000024',

'Queen'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000001',

'ROO000000001'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000002',

'ROO000000003'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000003',

'ROO000000005'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000004',

'ROO000000009'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000005',

'ROO000000010'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000006',

'ROO000000011'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000007',

'ROO000000012'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000008',

'ROO000000013'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000009',

'ROO000000015'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000010',

'ROO000000016'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000011',

'ROO000000017'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000012',

'ROO000000018'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000013',

'ROO000000019'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000014',

'ROO000000020'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000015',

'ROO000000021'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000016',

'ROO000000022'

)

INTO JABBERWOCKY.ROOM\_PICTURES

(

ROOM\_PICTURE,

ROOM\_ID

)

VALUES

(

'C:\ROOM\_PHOTOS\IMAGE000000000017',

'ROO000000023'

)

SELECT \* FROM DUAL;

COMMIT;

INSERT INTO REGION\_RATE

(RATE\_BEGIN\_DATE, REG\_ID, REGION\_RATE

)

SELECT AA.RATE\_BEGIN\_DATE,

BB.REG\_ID,

AA.REGION\_RATE

FROM

(SELECT \*

FROM

(SELECT TRIM(

CASE

WHEN SUBSTR(COUNTY,1,(INSTR(COUNTY,'/',1,1)) - 1) IS NULL

THEN COUNTY

ELSE SUBSTR(COUNTY,1,(INSTR(COUNTY,'/',1,1)) - 1)

END) AS COUNTY ,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

FROM region\_raw

UNION

SELECT TRIM(SUBSTR(COUNTY,(INSTR(COUNTY,'/',1,1) + 1),((INSTR(COUNTY,'/',1,2)-1) - (INSTR(COUNTY,'/',1,1) + 1)))) AS COUNTY ,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

FROM region\_raw

WHERE TRIM(SUBSTR(COUNTY,(INSTR(COUNTY,'/',1,1) + 1),((INSTR(COUNTY,'/',1,2)-1) - (INSTR(COUNTY,'/',1,1) + 1)))) IS NOT NULL

UNION

SELECT TRIM(SUBSTR(COUNTY,(INSTR(COUNTY,'/',1,2) + 1),((INSTR(COUNTY,'/',1,3)-1) - (INSTR(COUNTY,'/',1,2) + 1)))) AS COUNTY ,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

FROM region\_raw

WHERE TRIM(SUBSTR(COUNTY,(INSTR(COUNTY,'/',1,2) + 1),((INSTR(COUNTY,'/',1,3)-1) - (INSTR(COUNTY,'/',1,2) + 1)))) IS NOT NULL

UNION

SELECT

CASE

WHEN INSTR(COUNTY,'/',1,3) = 0

THEN NULL

ELSE TRIM(SUBSTR(COUNTY,(INSTR(COUNTY,'/',1,3) + 1),(LENGTH(COUNTY) - INSTR(COUNTY,'/',1,3) + 1)))

END AS COUNTY ,

OCT15,

NOV15,

DEC15,

JAN16,

FEB16,

MAR16,

APR16,

MAY16,

JUN16,

JUL16,

AUG16,

SEP16

FROM region\_raw

WHERE

CASE

WHEN INSTR(COUNTY,'/',1,3) = 0

THEN NULL

ELSE TRIM(SUBSTR(COUNTY,(INSTR(COUNTY,'/',1,3) + 1),(LENGTH(COUNTY) - INSTR(COUNTY,'/',1,3) + 1)))

END IS NOT NULL

) UNPIVOT (REGION\_RATE FOR RATE\_BEGIN\_DATE IN (OCT15 AS '01-OCT-15',NOV15 AS '01-NOV-15',DEC15 AS '01-DEC-15',JAN16 AS '01-JAN-16',FEB16 AS '01-FEB-16',MAR16 AS '01-MAR-16',APR16 AS '01-APR-16',MAY16 AS '01-MAY-16',JUN16 AS '01-JUN-16',JUL16 AS '01-JUL-16',AUG16 AS '01-AUG-16',SEP16 AS '01-SEP-16'))

) AA

INNER JOIN REGION BB

ON AA.COUNTY = BB.COUNTY

ORDER BY BB.REG\_ID,

AA.RATE\_BEGIN\_DATE;

COMMIT;

INSERT ALL

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Agender'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Androgyne'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Androgynous'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Bigender'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cis'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cisgender'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cis Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cis Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cis Man'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cis Woman'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cisgender Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cisgender Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cisgender Man'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Cisgender Woman'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Female to Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'FTM'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Gender Fluid'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Gender Nonconforming'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Gender Questioning'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Gender Variant'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Genderqueer'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Intersex'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Male to Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'MTF'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Neither'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Neutrois'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Non-binary'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Other'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Pangender'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans\*'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans\* Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans\* Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans Man'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans\* Man'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans Person'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans\* Person'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans Woman'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Trans\* Woman'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transfeminine'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transgender'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transgender Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transgender Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transgender Man'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transgender Person'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transgender Woman'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transmasculine'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transsexual'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transsexual Female'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transsexual Male'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transsexual Man'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transsexual Person'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Transsexual Woman'

)

INTO JABBERWOCKY.GENDER

(

GENDER\_DESCR

)

VALUES

(

'Two-Spirit'

)

SELECT \* FROM DUAL;

COMMIT;

## /\*Alter Table Statements\*/

ALTER TABLE JABBERWOCKY.EMPLOYEE ADD ( CONSTRAINT EMPLOYEE\_DEPTID\_FK FOREIGN KEY(DEPT\_ID) REFERENCES JABBERWOCKY.DEPARTMENT(DEPT\_ID), CONSTRAINT EMPLOYEE\_OFFID\_FK FOREIGN KEY(OFF\_ID) REFERENCES JABBERWOCKY.OFFICE(OFF\_ID), CONSTRAINT EMPLOYEE\_REWARDID\_FK FOREIGN KEY(REWARD\_ID) REFERENCES JABBERWOCKY.REWARD\_SYSTEM(REWARD\_ID), CONSTRAINT EMPLOYEE\_GENDERID\_FK FOREIGN KEY(GENDER\_ID) REFERENCES JABBERWOCKY.GENDER(GENDER\_ID));

ALTER TABLE JABBERWOCKY.PASSWORD ADD ( CONSTRAINT PASSWORD\_SQNC\_MPD\_PK PRIMARY KEY (SEQUENCE\_ID,EMP\_ID), CONSTRAINT PASSWORD\_EMPID\_FK FOREIGN KEY(EMP\_ID) REFERENCES JABBERWOCKY.EMPLOYEE(EMP\_ID));

ALTER TABLE JABBERWOCKY.REWARD\_LOG ADD ( CONSTRAINT REWARD\_LOG\_MPD\_PMNT\_PK PRIMARY KEY (EMP\_ID,PMNT\_ID), CONSTRAINT REWARD\_LOG\_EMPID\_FK FOREIGN KEY (EMP\_ID) REFERENCES JABBERWOCKY.EMPLOYEE (EMP\_ID), CONSTRAINT REWARD\_LOG\_PMNTID\_FK FOREIGN KEY(PMNT\_ID) REFERENCES JABBERWOCKY.PAYMENT(PMNT\_ID));

ALTER TABLE JABBERWOCKY.TRAVELER ADD ( CONSTRAINT TRAVELER\_TRAVELERID\_FK FOREIGN KEY(TRAVELER\_ID) REFERENCES JABBERWOCKY.EMPLOYEE(EMP\_ID));

ALTER TABLE JABBERWOCKY.HOST ADD ( CONSTRAINT HOST\_HOSTID\_FK FOREIGN KEY(HOST\_ID) REFERENCES JABBERWOCKY.EMPLOYEE(EMP\_ID));

ALTER TABLE JABBERWOCKY.RESERVATION ADD ( CONSTRAINT RESERVATION\_TRAVELERID\_FK FOREIGN KEY(TRAVELER\_ID) REFERENCES JABBERWOCKY.TRAVELER(TRAVELER\_ID), CONSTRAINT RESERVATION\_ROOMID\_FK FOREIGN KEY(ROOM\_ID) REFERENCES JABBERWOCKY.ROOM(ROOM\_ID), CONSTRAINT RESERVATION\_OFFID\_FK FOREIGN KEY(OFF\_ID) REFERENCES JABBERWOCKY.OFFICE(OFF\_ID), CONSTRAINT RESERVATION\_PMNTID\_FK FOREIGN KEY(PMNT\_ID) REFERENCES JABBERWOCKY.PAYMENT(PMNT\_ID), CONSTRAINT RESERVATION\_DEPTID\_FK FOREIGN KEY(DEPT\_ID) REFERENCES JABBERWOCKY.DEPARTMENT(DEPT\_ID));

ALTER TABLE JABBERWOCKY.CANCELLATION ADD ( CONSTRAINT CANCELLATION\_RESERVID\_FK FOREIGN KEY(RESERV\_ID) REFERENCES JABBERWOCKY.RESERVATION(RESERV\_ID));

ALTER TABLE JABBERWOCKY.ROOM ADD ( CONSTRAINT ROOM\_ADDRSID\_FK FOREIGN KEY(ADDRS\_ID) REFERENCES JABBERWOCKY.ADDRESS(ADDRS\_ID));

ALTER TABLE JABBERWOCKY.ADDRESS ADD ( CONSTRAINT ADDRESS\_HOSTID\_FK FOREIGN KEY(HOST\_ID) REFERENCES JABBERWOCKY.HOST(HOST\_ID), CONSTRAINT ADDRESS\_NBHDID\_FK FOREIGN KEY(NBHD\_ID) REFERENCES JABBERWOCKY.NEIGHBORHOOD(NBHD\_ID));

ALTER TABLE JABBERWOCKY.SURVEYS ADD ( CONSTRAINT SURVEYS\_RESERVID\_FK FOREIGN KEY(RESERV\_ID) REFERENCES JABBERWOCKY.RESERVATION(RESERV\_ID), CONSTRAINT SURVEYS\_EMPID\_FK FOREIGN KEY(EMP\_ID) REFERENCES JABBERWOCKY.EMPLOYEE(EMP\_ID));

ALTER TABLE JABBERWOCKY.ANSWERS ADD ( CONSTRAINT ANSWERS\_QSTN\_SRVY\_PK PRIMARY KEY (QUESTION\_ID,SURVEY\_ID), CONSTRAINT ANSWERS\_QUESTIONID\_FK FOREIGN KEY (QUESTION\_ID) REFERENCES JABBERWOCKY.SURVEY\_QUESTIONS (QUESTION\_ID), CONSTRAINT ANSWERS\_SURVEYID\_FK FOREIGN KEY(SURVEY\_ID) REFERENCES JABBERWOCKY.SURVEYS(SURVEY\_ID));

ALTER TABLE JABBERWOCKY.CO\_CONTACT ADD ( CONSTRAINT CO\_CONTACT\_COID\_FK FOREIGN KEY(CO\_ID) REFERENCES JABBERWOCKY.COMPANY(CO\_ID));

ALTER TABLE JABBERWOCKY.OFFICE ADD ( CONSTRAINT OFFICE\_COID\_FK FOREIGN KEY(CO\_ID) REFERENCES JABBERWOCKY.COMPANY(CO\_ID), CONSTRAINT OFFICE\_REGID\_FK FOREIGN KEY(REG\_ID) REFERENCES JABBERWOCKY.REGION(REG\_ID));

ALTER TABLE JABBERWOCKY.CONTACT\_PHONE ADD ( CONSTRAINT CONTACT\_PHONE\_CONTID\_FK FOREIGN KEY(CONT\_ID) REFERENCES JABBERWOCKY.CO\_CONTACT(CONT\_ID));

ALTER TABLE JABBERWOCKY.PAYMENT ADD ( CONSTRAINT PAYMENT\_DEPTID\_FK FOREIGN KEY(DEPT\_ID) REFERENCES JABBERWOCKY.DEPARTMENT(DEPT\_ID));

ALTER TABLE JABBERWOCKY.ROOM\_BEDS ADD ( CONSTRAINT ROOM\_BEDS\_BDN\_RMD\_PK PRIMARY KEY (BED\_NO,ROOM\_ID), CONSTRAINT ROOM\_BEDS\_ROOMID\_FK FOREIGN KEY(ROOM\_ID) REFERENCES JABBERWOCKY.ROOM(ROOM\_ID));

ALTER TABLE JABBERWOCKY.ROOM\_PICTURES ADD ( CONSTRAINT ROOM\_PICTURES\_RMPC\_RMD\_PK PRIMARY KEY (ROOM\_PICTURE,ROOM\_ID), CONSTRAINT ROOM\_PICTURES\_ROOMID\_FK FOREIGN KEY(ROOM\_ID) REFERENCES JABBERWOCKY.ROOM(ROOM\_ID));

ALTER TABLE JABBERWOCKY.REGION\_RATE ADD ( CONSTRAINT REGION\_RATE\_RTBG\_RGD\_PK PRIMARY KEY (RATE\_BEGIN\_DATE,REG\_ID), CONSTRAINT REGION\_RATE\_REGID\_FK FOREIGN KEY(REG\_ID) REFERENCES JABBERWOCKY.REGION(REG\_ID));

ALTER TABLE JABBERWOCKY.ROOM\_RATE ADD ( CONSTRAINT ROOM\_RATE\_ROOMID\_FK FOREIGN KEY(ROOM\_ID) REFERENCES JABBERWOCKY.ROOM(ROOM\_ID), CONSTRAINT ROOM\_RATE\_TRAVELERID\_FK FOREIGN KEY(TRAVELER\_ID) REFERENCES JABBERWOCKY.TRAVELER(TRAVELER\_ID));

ALTER TABLE JABBERWOCKY.AVAILABLE\_ROOM ADD ( CONSTRAINT AVAILABLE\_ROOM\_TRAVELERID\_FK FOREIGN KEY(TRAVELER\_ID) REFERENCES JABBERWOCKY.TRAVELER(TRAVELER\_ID), CONSTRAINT AVAILABLE\_ROOM\_ROOMID\_FK FOREIGN KEY(ROOM\_ID) REFERENCES JABBERWOCKY.ROOM(ROOM\_ID));

# /\*Appendix to Chapter 3: Triggers and Procedures\*/

## /\*Trigger 1 – Cancellation\*/

create or replace TRIGGER TRIG\_CAN\_SEQ BEFORE

INSERT ON CANCELLATION

FOR EACH ROW

DECLARE

TEMP\_CAN\_NO CANCELLATION.CANC\_ID%TYPE ;

TEMP\_CANC\_LIMIT CANCELLATION.CANC\_LIMIT%TYPE;

TEMP\_REF\_AMT CANCELLATION.REF\_AMT%TYPE;

TEMP\_RES\_START RESERVATION.RESERV\_START%TYPE;

TEMP\_IS\_CANC RESERVATION.ISCANC%TYPE;

TEMP\_CANC\_DATE CANCELLATION.CANC\_DATE%TYPE;

temp\_tot\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

temp\_tot\_co\_spend COMPANY.total\_co\_spend%type ;

TEMP\_DEPT\_ID DEPARTMENT.DEPT\_ID%TYPE;

TEMP\_CO\_ID COMPANY.CO\_ID%TYPE;

BEGIN

--Insert the cancellation number

SELECT 'CAN'

|| LPAD( TO\_CHAR(CANCELLATION\_CANC\_ID\_SEQ.NEXTVAL ),9,'0')

INTO TEMP\_CAN\_NO

FROM DUAL;

:new.CANC\_ID := TEMP\_CAN\_NO ;

--Handle duplicate cancellation errors

SELECT ISCANC INTO TEMP\_IS\_CANC FROM RESERVATION

WHERE RESERV\_ID = :new.reserv\_ID;

IF (TEMP\_IS\_CANC = 'Y') THEN

raise\_application\_error(-20404, 'This reservation has already been cancelled.');

ELSE

TEMP\_CANC\_DATE := SYSDATE; --assign the canc\_date

TEMP\_IS\_CANC := 'Y'; --mark the reservation as cancelled

SELECT RESERV\_START INTO TEMP\_RES\_START

FROM RESERVATION WHERE RESERV\_ID = :new.RESERV\_ID;

TEMP\_CANC\_LIMIT := TEMP\_RES\_START - 1; --determine the last refundable canc\_date

IF (SYSDATE <= TEMP\_CANC\_LIMIT) THEN

SELECT PMNT\_PRICE INTO TEMP\_REF\_AMT

FROM RESERVATION r JOIN PAYMENT p ON r.PMNT\_ID = p.PMNT\_ID

WHERE RESERV\_ID = :new.RESERV\_ID; --determine ref amt for timely canc

ELSE

TEMP\_REF\_AMT := 0; --no refund for late cancellation

END IF;

--insert values for canc\_limit, ref\_amt, and canc\_date in the CANCELLATION table

:new.CANC\_LIMIT := TEMP\_CANC\_LIMIT;

:new.REF\_AMT := TEMP\_REF\_AMT;

:new.CANC\_DATE := TEMP\_CANC\_DATE;

UPDATE RESERVATION SET isCanc = TEMP\_IS\_CANC --update cancellation state

WHERE RESERV\_ID = :new.RESERV\_ID; --in RESERVATION table

--update the total department spend to subtract refund

SELECT tot\_dept\_spend,r.DEPT\_ID INTO temp\_tot\_dept\_spend, TEMP\_DEPT\_ID

FROM RESERVATION r JOIN DEPARTMENT d ON r.DEPT\_ID = d.DEPT\_ID

WHERE r.RESERV\_ID = :new.reserv\_ID;

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend - TEMP\_REF\_AMT;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_dept\_spend

Where dept\_ID = TEMP\_DEPT\_ID;

--update total company spend to subtract refund

SELECT total\_co\_spend, c.CO\_ID INTO temp\_tot\_co\_spend, TEMP\_CO\_ID

FROM RESERVATION r JOIN DEPARTMENT d ON r.DEPT\_ID = d.DEPT\_ID

JOIN COMPANY c ON d.CO\_ID = c.CO\_ID

WHERE r.RESERV\_ID = :new.reserv\_ID;

temp\_tot\_co\_spend := temp\_tot\_co\_spend - TEMP\_REF\_AMT;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID ;

END IF;

END ;

/

## /\*Trigger 2 – Department and Company Spend\*/

-------------------------------Trigger -------------------------------

--14—- Automatically Update Totol\_Department\_Spend,Total\_Company\_Spend,

--After Insert or Update of pmnt\_price on PAYMENT

-- Event:

-- Trigger:

Create or Replace Trigger Trig\_Dept\_Company\_Spend Before Insert or

Update Of pmnt\_price,dept\_ID on PAYMENT

For Each Row

Declare

temp\_tot\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

temp\_tot\_co\_spend COMPANY.total\_co\_spend%type ;

temp\_pmnt\_price PAYMENT.pmnt\_price%type;

temp\_co\_ID COMPANY.co\_ID%type;

temp\_tot\_old\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

temp\_tot\_new\_dept\_spend DEPARTMENT.tot\_dept\_spend%type ;

Begin

/\*

If Updating('pmnt\_ID') Then

If Updating ('dept\_ID') Then

Select tot\_dept\_spend into temp\_tot\_old\_dept\_spend

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select tot\_dept\_spend into temp\_tot\_new\_dept\_spend

From DEPARTMENT Where dept\_ID = : new.dept\_ID ;

Select co\_ID into temp\_co\_ID

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select total\_co\_spend into temp\_tot\_co\_spend

From COMPANY Where co\_ID = temp\_co\_ID ;

-- Update total departmental spend

temp\_tot\_old\_dept\_spend := temp\_tot\_old\_dept\_spend + : old.pmnt\_price;

temp\_tot\_new\_dept\_spend := temp\_tot\_new\_dept\_spend - : new.pmnt\_price;

-- Update total company's spend

temp\_tot\_co\_spend := temp\_tot\_co\_spend - : old.pmnt\_price + : new.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_old\_dept\_spend

Where dept\_ID = : old.dept\_ID;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_new\_dept\_spend

Where dept\_ID = : new.dept\_ID;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID;

End If;

\*/

If (Inserting)Then

Select tot\_dept\_spend,co\_ID into temp\_tot\_dept\_spend, temp\_co\_ID

From DEPARTMENT Where dept\_ID = : new.dept\_ID ;

Select total\_co\_spend into temp\_tot\_co\_spend

From COMPANY Where co\_ID = temp\_co\_ID ;

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend + : new.pmnt\_price;

temp\_tot\_co\_spend := temp\_tot\_co\_spend + : new.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_dept\_spend

Where dept\_ID = : new.dept\_ID;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID ;

End IF;

If (Updating ('pmnt\_price')) Then

Select tot\_dept\_spend, co\_ID into temp\_tot\_dept\_spend, temp\_co\_ID

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select total\_co\_spend into temp\_tot\_co\_spend

From COMPANY Where co\_ID = temp\_co\_ID ;

-- Update total departmental spend

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend - : old.pmnt\_price;

temp\_tot\_dept\_spend := temp\_tot\_dept\_spend + : new.pmnt\_price;

-- Update total company's spend

temp\_tot\_co\_spend := temp\_tot\_co\_spend - : old.pmnt\_price;

temp\_tot\_co\_spend := temp\_tot\_co\_spend + : new.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_dept\_spend

Where dept\_ID = : old.dept\_ID;

Update COMPANY Set total\_co\_spend = temp\_tot\_co\_spend

Where co\_ID = temp\_co\_ID;

End If;

If (Updating ('dept\_ID')) Then

Select tot\_dept\_spend into temp\_tot\_old\_dept\_spend

From DEPARTMENT Where dept\_ID = : old.dept\_ID ;

Select tot\_dept\_spend into temp\_tot\_new\_dept\_spend

From DEPARTMENT Where dept\_ID = : new.dept\_ID ;

-- Update total departmental spend

temp\_tot\_old\_dept\_spend := temp\_tot\_old\_dept\_spend - : old.pmnt\_price;

temp\_tot\_new\_dept\_spend := temp\_tot\_new\_dept\_spend + : old.pmnt\_price;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_old\_dept\_spend

Where dept\_ID = : old.dept\_ID;

Update DEPARTMENT Set tot\_dept\_spend = temp\_tot\_new\_dept\_spend

Where dept\_ID = : new.dept\_ID;

End If;

End;

/

commit;

-----------------------------Test The Trigger-------------------------------

/\*

-- Test Isert --

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

Insert into PAYMENT(DEPT\_ID,PMNT\_DATE, PMNT\_PRICE) Values ('DEP000007', '01-JAN-16', 1000);

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

--Update DEPARTMENT set tot\_dept\_spend = 10775 where dept\_ID = 'DEP000009';

-- Test Update Payment Price --

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Update PAYMENT Set PMNT\_PRICE = 2000 Where DEPT\_ID = 'DEP000007' And PMNT\_DATE ='01-JAN-16';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

--Update COMPANY set total\_co\_spend = 15337.8 where co\_ID = 'CO\_00001';

--Update DEPARTMENT set tot\_dept\_spend = 8775 where dept\_ID = 'DEP000009';

Rollback;

-- Test Update Department ID --

Select \* from PAYMENT Where DEPT\_ID = 'DEP000009' ;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000009';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' ;

Update PAYMENT Set DEPT\_ID = 'DEP000009' Where PMNT\_ID ='PMN000000059' ;

Select \* from PAYMENT Where DEPT\_ID = 'DEP000009' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000009';

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000010';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

Rollback;

-- Test Update Department ID & Payment Amount--

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007'or DEPT\_ID = 'DEP000009' or DEPT\_ID= 'DEP000010' order by dept\_ID ;

Update PAYMENT Set DEPT\_ID = 'DEP000010', PMNT\_PRICE = 3000 Where PMNT\_ID ='PMN000000059' ;

Select \* from PAYMENT Where DEPT\_ID = 'DEP000010' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000010';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000009' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000009';

Select \* from PAYMENT Where DEPT\_ID = 'DEP000007' order by pmnt\_ID desc;

Select \* from DEPARTMENT Where DEPT\_ID = 'DEP000007';

Select \* from COMPANY Where CO\_ID = 'CO\_00001';

rollback;

\*/

## /\*Procedure 1 – Make Payment\*/

--15—- Insert the Payment based on the Reservation (after execute the procedure)

-- Procedure

-- Input Parameter: reservation ID, reservation start date, reservatin end date

-- Output Parameter: Table: REVENUE\_PER\_CITY

Create or Replace Procedure Make\_Payment(

reservation\_ID RESERVATION.reserv\_ID%type )

AS

--/\*

temp\_start\_date RESERVATION.reserv\_start%type;

temp\_end\_date RESERVATION.reserv\_end%type;

temp\_current\_date RESERVATION.reserv\_start%type;

temp\_dept\_ID DEPARTMENT.dept\_ID%type ;

temp\_off\_ID OFFICE.off\_ID%type;

temp\_reg\_ID REGION.reg\_ID%type;

temp\_room\_ID ROOM.room\_ID%type;

temp\_pmnt\_date PAYMENT.pmnt\_date%type ;

temp\_pmnt\_ID PAYMENT.pmnt\_ID%type ;

temp\_pmnt\_price PAYMENT.pmnt\_price%type ;

temp\_rate\_begin\_date REGION\_RATE.rate\_begin\_date%type;

temp\_region\_rate REGION\_RATE.region\_rate%type;

temp\_room\_price RESERVATION.room\_price%type;

total\_room\_price RESERVATION.room\_price%type;

Begin

-- Delete old pmnt\_ID, if pmnt\_ID is not null

Select pmnt\_ID Into temp\_pmnt\_ID From RESERVATION Where reserv\_ID = reservation\_ID;

If (temp\_pmnt\_ID is not null) Then

Update RESERVATION Set pmnt\_ID = null Where reserv\_ID = reservation\_ID;

Delete From PAYMENT Where pmnt\_ID = temp\_pmnt\_ID;

End If;

-- Set up Initial Values

Select reserv\_start, reserv\_end ,

dept\_ID, off\_ID,room\_ID

Into temp\_start\_date, temp\_end\_date ,

temp\_dept\_ID, temp\_off\_ID, temp\_room\_ID

From RESERVATION

Where reserv\_ID = reservation\_ID;

-- Query the region ID based onthe off\_ID so room rate can be calculated.

Select reg\_ID into temp\_reg\_ID From OFFICE Where off\_ID = temp\_off\_ID;

-- Payment date is the system when execute this procedure

Select sysdate Into temp\_pmnt\_date From DUAL;

temp\_current\_date := temp\_start\_date;

total\_room\_price := 0;

-- Calcute the payment price based on reservation

WHILE (temp\_current\_date < temp\_end\_date) Loop

SELECT RR.REGION\_RATE AS RATE\_CHARGED INTO temp\_region\_rate FROM

REGION\_RATE RR

INNER JOIN REGION RE

ON RR.REG\_ID = RE.REG\_ID

INNER JOIN ADDRESS AA

ON TRIM(LOWER(AA.ADDRS\_COUNTY)) = TRIM(LOWER(RE.COUNTY))

INNER JOIN ROOM RO

ON AA.ADDRS\_ID = RO.ADDRS\_ID

WHERE room\_ID = temp\_room\_ID

And temp\_current\_date >= rate\_begin\_date

and temp\_current\_date <= (ADD\_MONTHS(rate\_begin\_date ,1)-1);

temp\_room\_price := temp\_region\_rate \* 0.9;

total\_room\_price := temp\_room\_price + total\_room\_price;

temp\_current\_date := temp\_current\_date +1;

End Loop;

temp\_pmnt\_price := total\_room\_price;

--Insert date into Table PAYMENT

INSERT INTO PAYMENT(dept\_ID,pmnt\_date, pmnt\_price)

VALUES (temp\_dept\_ID, temp\_pmnt\_date,temp\_pmnt\_price);

-- Update total room price into room price based on reservation.

Update RESERVATION Set room\_price = total\_room\_price Where reserv\_ID = reservation\_ID;

--Update foreign key value, current payment ID into Table RESERVATION

SELECT 'PMN'

|| LPAD( TO\_CHAR(PAYMENT\_PMNT\_ID\_SEQ.currval ),9,'0') into temp\_pmnt\_ID From Dual;

UPDATE RESERVATION SET pmnt\_ID = temp\_pmnt\_ID WHERE reserv\_ID = reservation\_ID;

COMMIT;

EXCEPTION

WHEN no\_data\_found THEN

raise\_application\_error (-20001, reservation\_ID||'does not exist ');

END;

/

-----------------------------Test The Procedure-------------------------------

--/\*

/\*

--Test 1 , Insert a new date into Table RAESERVATION and check the Table PAYMENT and RESERVATION

Select \* from RESERVATION where traveler\_ID = 'EMP000000004';

Insert Into RESERVATION (traveler\_ID,room\_ID,off\_ID,dept\_ID,reserv\_start,reserv\_end, iscanc)

Values ('EMP000000004','ROO000000001','OFF00002','DEP000009','10-DEC-15', '14-JAN-16', 'N');

Select \* From RESERVATION Where traveler\_ID ='EMP000000004';

Execute Make\_Payment('RES000000041');

Select \* From PAYMENT order by PMNT\_ID desc;

Select \* From RESERVATION Where traveler\_ID ='EMP000000004';

Rollback;

-- Test 2 , Insert a new date into Table RESERVATION and check the Table PAYMENT and RESERVATION

Select \* from RESERVATION;

Insert Into RESERVATION (traveler\_ID,room\_ID,off\_ID,dept\_ID,reserv\_start,reserv\_end, iscanc)

Values ('EMP000000004','ROO000000013','OFF00003','DEP000009','17-OCT-15', '14-DEC-15', 'N');

Select \* From RESERVATION Where traveler\_ID ='EMP000000004';

Execute Make\_Payment('RES000000041');

Select \* From PAYMENT Order by PMNT\_ID desc;

Select \* From RESERVATION Where traveler\_ID ='EMP000000004' Order By reserv\_ID Desc;

\*/

## /\*Procedure 2 – Revenue Per Period\*/

--2--Co.Host revenues per city over given time periods

-----------------------------Create Table-------------------------------

/\*

Drop Table REVENUE\_PER\_CITY;

Create Table REVENUE\_PER\_CITY

(

City varchar (20),

StartDate date,

EndDate date,

TotalRevenue decimal (24,6) default 0

) ;

Select \* From REVENUE\_PER\_CITY;

\*/

-------------------------------Procedure -------------------------------

-- Input Parameter: city, start\_date, end\_date

-- Output: Insert into Table REVENUE\_PER\_CITY

-- Procedure Format: Revenue\_Per\_City\_For\_Period ('San Diego','01-JAN-15', '01-FEB-15')

-- Excute the Procedure: Execute Revenue\_Per\_City\_For\_Period ('San Diego','01-JAN-15', '01-FEB-15') ;

Create or Replace Procedure Revenue\_For\_Period

(

start\_date PAYMENT.pmnt\_date%type,

end\_date PAYMENT.pmnt\_date%type

)

--Return PAYMENT.pmnt\_price%type

IS

-- Declare the Variable

Cursor C1 is Select a.addrs\_city, p.pmnt\_date, p.pmnt\_price, c.ref\_amt

From ADDRESS a JOIN ROOM r ON a.addrs\_ID = r.addrs\_ID

JOIN RESERVATION res ON r.room\_ID = res.room\_ID

JOIN PAYMENT p ON res.pmnt\_ID = p.pmnt\_ID

LEFT OUTER JOIN CANCELLATION c ON res.reserv\_ID = c.reserv\_ID

Where pmnt\_date >= start\_date and pmnt\_date <= end\_date

;

Cursor C2 is Select distinct addrs\_city from ADDRESS;

temp\_city ADDRESS.addrs\_city%type;

temp\_revenue PAYMENT.pmnt\_price%type;

total\_revenue PAYMENT.pmnt\_price%type;

Begin

Delete From REVENUE\_PER\_CITY;

For C2\_row in C2 Loop

temp\_revenue:= 0;

total\_revenue:=0;

temp\_city := C2\_row.addrs\_city;

For C1\_row in C1 Loop

If (C1\_row.addrs\_city = temp\_city) Then

temp\_revenue := C1\_row.pmnt\_price - Coalesce(C1\_row.ref\_amt, 0);

total\_revenue := temp\_revenue + total\_revenue ;

End If;

End Loop;

Insert into REVENUE\_PER\_CITY (City,StartDate, EndDate,TotalRevenue)

values(temp\_city, start\_date, end\_date, total\_revenue);

End Loop;

End;

/

--commit;

-----------------------------Test The Procedure-------------------------------

/\*

Execute Revenue\_For\_Period('01-Jan-15','01-JAN-16') ;

Select \* from REVENUE\_PER\_CITY ;

Execute Revenue\_For\_Period( '01-Jan-15','01-SEP-16') ;

Select \* from REVENUE\_PER\_CITY ;

Execute Revenue\_For\_Period('21-May-15','30-SEP-15') ;

Select \* from REVENUE\_PER\_CITY ;

\*/

## /\*Procedure 3 – Available Rooms\*/

/\*------------------------------------------------------------------------------

---------------------Procedure to find avaliable rooms--------------------------

Purpose: This procedure is designed to find all available rooms within

our database where the active Traveler's company matches the

available Hosts' company, the room is in the same region as the

destination region, and where the Traveler is not also the Host.

To accomplish this we take ROOM and use MINUS to remove all rooms

where PREFFERED\_START date and the PREFFERED\_END date overlap the

dates that have been reserved for this room in the RESERVATION table.

Then we get the host information for each available room by joining

the ADDRESS, HOST, EMPLOYEE, and DEPARTMENTS tables and further limit

the results to only those where the TRAVELER is not also the HOST.

Then we validate that the results are in the preffered county and

preffered state. This is to be sure that we do not join to duplicate

counties across many states.

Finally we join the traveler, employee, and department tables where

the TRAVLER\_ID in the TRAVELER table is the same as the active travler

and limit the results to only those where the traveler and host

are in the same company.

We take these results and load them into a cursor and user the cursor.

The procedure then users the cursor to append the results to the

table AVAILABLE\_ROOM. The results are effective dated and

include the active TRAVELER\_ID to allow the front end to quickly

identify pertinent results by choosing all records where the

TRAVELER\_ID in AVAILABLE\_ROOM matches the active TRAVELER\_ID

and where the AVAIL\_CHECK\_DATE is the maximum from that group.

The AVAILABLE\_ROOM table is intended to hold a history of user

queries until such time as it is truncated so that CoHost can

analyze user preferences in their searches.

There is a toggle written in to the procedure that will allow the

user to activate a delete of the AVAILABLE\_ROOM table each time

the procedure is run but this does not actively support more than

one user at a time.

Parameters:

TRAV\_CHECKING The traveler Id of the person looking for a room.

OFFICE\_COUNTY The county that the traveler is trying to travel to.

OFFICE\_STATE The state that the traveler is trying to travel to.

Necessary to differentiate duplicate county names.

PREFERRED\_START The prefferred start date the traveler desires.

PREFFERED\_END The prefferred end date that the traveler desires.

Exception Handling:

There are 2 exceptions handled by this procedure.

1. If the user has entered a date before today's date they are prompted to

enter a valid date.

2. If any other exception happens the ORA-20005 exception is thrown and the user

is given a brief message explaining that no rooms were found for these dates.

Output: This procedure appends a ROOM\_ID and many interesting attributes

related to that room including all tracked amenities. The output also

includes the TRAVELER\_ID of the active traveler, the dates queried,

the date the procedure was run, and the room rate for each room

returned over the dates queried.

Future improvements:

Exception reporting could be improved to identify further errors and exceptions

that occur at runtime.

------------------------------------------------------------------------------\*/

CREATE OR REPLACE PROCEDURE AVAILABLE\_ROOMS(

TRAV\_CHECKING TRAVELER.TRAVELER\_ID%TYPE,

OFFICE\_COUNTY ADDRESS.ADDRS\_COUNTY%TYPE,

OFFICE\_STATE ADDRESS.ADDRS\_STATE%TYPE,

PREFFERED\_START RESERVATION.RESERV\_START%TYPE,

PREFFERED\_END RESERVATION.RESERV\_END%TYPE)

AS

/\*Variable Declaration\*/

TEMP\_TRAV TRAVELER.TRAVELER\_ID%TYPE;

TEMP\_ROOM\_ID ROOM.ROOM\_ID%TYPE;

TEMP\_STREET ADDRESS.ADDRS\_STREET%TYPE;

TEMP\_CITY ADDRESS.ADDRS\_CITY%TYPE;

TEMP\_STATE ADDRESS.ADDRS\_STATE%TYPE;

TEMP\_NBHD\_NAME NEIGHBORHOOD.NBHD\_NAME%TYPE;

TEMP\_NBHD\_TYPE NEIGHBORHOOD.NBHD\_TYPE%TYPE;

TEMP\_NBHD\_FEAT NEIGHBORHOOD.NBHD\_FEAT%TYPE;

TEMP\_WIFI ADDRESS.WIFI%TYPE;

TEMP\_PETS ADDRESS.PETS%TYPE;

TEMP\_CHILDREN ADDRESS.CHILDREN%TYPE;

TEMP\_KITCHEN ADDRESS.KITCHEN%TYPE;

TEMP\_COMMON\_AREA ADDRESS.COMMON\_AREA%TYPE;

TEMP\_OWNER\_OCC ADDRESS.OWNER\_OCC%TYPE;

TEMP\_TELEVISION ADDRESS.TELEVISION%TYPE;

TEMP\_BLDG\_TYPE ADDRESS.BLDG\_TYPE%TYPE;

TEMP\_SMOKING ADDRESS.SMOKING%TYPE;

TEMP\_WASHER\_DRYER ADDRESS.WASHER\_DRYER%TYPE;

TEMP\_POOL ADDRESS.POOL%TYPE;

TEMP\_OTHER ADDRESS.OTHER%TYPE;

TEMP\_COUNTY ADDRESS.ADDRS\_COUNTY%TYPE;

TEMP\_START RESERVATION.RESERV\_START%TYPE;

TEMP\_END RESERVATION.RESERV\_END%TYPE;

TEMP\_LAT ADDRESS.ADDRS\_LATITUDE%TYPE;

TEMP\_LONG ADDRESS.ADDRS\_LONGITUDE%TYPE;

TEMP\_RATE\_DATE RESERVATION.RESERV\_START%TYPE;

TEMP\_RATE RESERVATION.ROOM\_PRICE%TYPE;

TEMP\_INCREMENT RESERVATION.ROOM\_PRICE%TYPE;

/\*Cursor to hold the resulst to insert into the AVAILABLE\_ROOM table\*/

CURSOR C1

IS

SELECT RO.ROOM\_ID,

AA.ADDRS\_STREET,

AA.ADDRS\_CITY,

NBH.NBHD\_NAME,

NBH.NBHD\_TYPE,

NBH.NBHD\_FEAT,

AA.WIFI,

AA.PETS,

AA.CHILDREN,

AA.KITCHEN,

AA.COMMON\_AREA,

AA.OWNER\_OCC,

AA.TELEVISION,

AA.BLDG\_TYPE,

AA.SMOKING,

AA.WASHER\_DRYER,

AA.POOL,

AA.OTHER,

AA.ADDRS\_LATITUDE,

AA.ADDRS\_LONGITUDE

FROM ROOM RO

INNER JOIN

(SELECT RO.ROOM\_ID FROM ROOM RO

MINUS

(SELECT RE.ROOM\_ID

FROM ROOM RO

INNER JOIN reservation RE

ON RO.ROOM\_ID = RE.ROOM\_ID

INNER JOIN ADDRESS AA

ON RO.ADDRS\_ID = AA.ADDRS\_ID

/\*Find all dates that are not already booked\*/

WHERE ((PREFFERED\_START <= RE.RESERV\_START

AND RE.RESERV\_START <= PREFFERED\_END)

OR(PREFFERED\_START < RE.RESERV\_END

AND RE.RESERV\_END <= PREFFERED\_END))

AND AA.ADDRS\_COUNTY = OFFICE\_COUNTY

AND RE.ISCANC = 'N'

GROUP BY RE.ROOM\_ID

)

) ROOMS ON RO.ROOM\_ID = ROOMS.ROOM\_ID

INNER JOIN ADDRESS AA

ON RO.ADDRS\_ID = AA.ADDRS\_ID

INNER JOIN NEIGHBORHOOD NBH

ON AA.NBHD\_ID = NBH.NBHD\_ID

INNER JOIN (SELECT AA.ADDRS\_ID,DE.CO\_ID,HO.HOST\_ID FROM ADDRESS AA

INNER JOIN HOST HO

ON AA.HOST\_ID = HO.HOST\_ID

INNER JOIN EMPLOYEE EE

ON EE.EMP\_ID = HO.HOST\_ID

INNER JOIN DEPARTMENT DE

ON DE.DEPT\_ID = EE.DEPT\_ID

WHERE HO.HOST\_ID <> TEMP\_TRAV) HO

ON AA.ADDRS\_ID = HO.ADDRS\_ID

/\*Limit the results to just those within the proper region\*/

WHERE AA.ADDRS\_COUNTY = OFFICE\_COUNTY

AND AA.ADDRS\_STATE = OFFICE\_STATE

AND HO.CO\_ID =

(SELECT DE.CO\_ID FROM TRAVELER TR

INNER JOIN EMPLOYEE EE

ON EE.EMP\_ID = TR.TRAVELER\_ID

INNER JOIN DEPARTMENT DE

ON DE.DEPT\_ID = EE.DEPT\_ID

WHERE TR.TRAVELER\_ID = TEMP\_TRAV

GROUP BY DE.CO\_ID)

;

BEGIN

/\*Toggle this command depending on how the AVAIABLE\_ROOM table will be used

if it is used to hold all room availability queries ever toggle it off

if it is used to hold only the current session toggle it on. By default

it is set to off\*/

--delete AVAILABLE\_ROOM;

/\*Assign values to those variables that are not held in the cursor\*/

TEMP\_TRAV := TRAV\_CHECKING;

TEMP\_COUNTY := OFFICE\_COUNTY;

TEMP\_STATE := OFFICE\_STATE;

TEMP\_START := PREFFERED\_START;

TEMP\_END := PREFFERED\_END;

TEMP\_RATE\_DATE := PREFFERED\_START;

TEMP\_RATE := 0;

TEMP\_INCREMENT := 0;

/\*For loop to assign values to all variables inside the cursor\*/

IF TO\_DATE(TEMP\_START) < TO\_DATE(SYSDATE)

THEN

raise\_application\_error

(-20009, 'Please enter a valid date equal to or after today''s date.');

ELSE

FOR C1\_REC IN C1

LOOP

TEMP\_ROOM\_ID := C1\_REC.ROOM\_ID;

TEMP\_STREET := C1\_REC.ADDRS\_STREET;

TEMP\_CITY := C1\_REC.ADDRS\_CITY;

TEMP\_NBHD\_NAME := C1\_REC.NBHD\_NAME;

TEMP\_NBHD\_TYPE := C1\_REC.NBHD\_TYPE;

TEMP\_NBHD\_FEAT := C1\_REC.NBHD\_FEAT;

TEMP\_WIFI := C1\_REC.WIFI;

TEMP\_PETS := C1\_REC.PETS;

TEMP\_CHILDREN := C1\_REC.CHILDREN;

TEMP\_KITCHEN := C1\_REC.KITCHEN;

TEMP\_COMMON\_AREA := C1\_REC.COMMON\_AREA;

TEMP\_OWNER\_OCC := C1\_REC.OWNER\_OCC;

TEMP\_TELEVISION := C1\_REC.TELEVISION;

TEMP\_BLDG\_TYPE := C1\_REC.BLDG\_TYPE;

TEMP\_SMOKING := C1\_REC.SMOKING;

TEMP\_WASHER\_DRYER := C1\_REC.WASHER\_DRYER;

TEMP\_POOL := C1\_REC.POOL;

TEMP\_OTHER := C1\_REC.OTHER;

TEMP\_LAT := C1\_REC.ADDRS\_LATITUDE;

TEMP\_LONG := C1\_REC.ADDRS\_LONGITUDE;

/\* If statement checks to see if date is valid. If date is before todays

date we raise an application error to warn the customer to choose a

date that is actually possible\*/

/\*Assign default values to TEMP\_RATE\_DATE and TEMP\_RATE before the loop

to calculate the total amount of the stay\*/

TEMP\_RATE\_DATE := TEMP\_START;

TEMP\_RATE := 0;

/\*Loop to calculate the total amount that will be charged for each

potential stay.\*/

WHILE TEMP\_RATE\_DATE < TEMP\_END

LOOP

SELECT (RR.REGION\_RATE \* .9) INTO TEMP\_INCREMENT

FROM ROOM RO

INNER JOIN ADDRESS AA

ON RO.ADDRS\_ID = AA.ADDRS\_ID

INNER JOIN REGION RE

ON TRIM(LOWER(AA.ADDRS\_COUNTY)) = TRIM(LOWER(RE.COUNTY))

INNER JOIN REGION\_RATE RR

ON RE.REG\_ID = RR.REG\_ID

WHERE RR.RATE\_BEGIN\_DATE <= TEMP\_RATE\_DATE AND TEMP\_RATE\_DATE <= (ADD\_MONTHS(RATE\_BEGIN\_DATE,1)-1)

AND RO.ROOM\_ID = TEMP\_ROOM\_ID;

/\*Assignment for each value during each loop\*/

TEMP\_RATE := TEMP\_RATE + NVL(TEMP\_INCREMENT,0);

TEMP\_RATE\_DATE := TEMP\_RATE\_DATE + 1;

END LOOP;

/\*Insert each row into a table to capture each value.\*/

INSERT

INTO AVAILABLE\_ROOM

(

TRAVELER\_ID,

ROOM\_ID,

RESERV\_START,

RESERV\_END,

ROOM\_PRICE,

ADDRS\_STREET,

ADDRS\_CITY,

ADDRS\_STATE,

NBHD\_NAME,

NBHD\_TYPE,

NBHD\_FEAT,

WIFI,

PETS,

CHILDREN,

KITCHEN,

COMMON\_AREA,

OWNER\_OCC,

TELEVISION,

BLDG\_TYPE,

SMOKING,

WASHER\_DRYER,

POOL,

OTHER,

ADDRS\_LATITUDE,

ADDRS\_LONGITUDE,

AVAIL\_CHECK\_DATE

)

VALUES

(

TEMP\_TRAV,

TEMP\_ROOM\_ID,

TEMP\_START,

TEMP\_END,

TEMP\_RATE,

TEMP\_STREET,

TEMP\_CITY,

TEMP\_STATE,

TEMP\_NBHD\_NAME,

TEMP\_NBHD\_TYPE,

TEMP\_NBHD\_FEAT,

TEMP\_WIFI,

TEMP\_PETS,

TEMP\_CHILDREN,

TEMP\_KITCHEN,

TEMP\_COMMON\_AREA,

TEMP\_OWNER\_OCC,

TEMP\_TELEVISION,

TEMP\_BLDG\_TYPE,

TEMP\_SMOKING,

TEMP\_WASHER\_DRYER,

TEMP\_POOL,

TEMP\_OTHER,

TEMP\_LAT,

TEMP\_LONG,

SYSDATE

);

END LOOP;

END IF;

/\*Commit all changes after the loop has finished\*/

COMMIT;

/\* Exception handling for the case when there are no rooms available\*/

EXCEPTION

WHEN no\_data\_found THEN

raise\_application\_error (-20005, 'No rooms found for these dates');

END;

/

/\*

EXECUTE AVAILABLE\_ROOMS ('EMP000000037','Santa Clara','CA','10-DEC-15','20-DEC-15');

SELECT \* FROM AVAILABLE\_ROOM;

---query for usefull information from available\_room

SELECT ADDRS\_STREET AS "Address Street",

ADDRS\_CITY AS "Address City",

BLDG\_TYPE AS "Building Type",

RESERV\_START AS "Reservation Start Date",

RESERV\_END AS "Reservation End Date",

ROOM\_PRICE AS "Reservation Amount",

NBHD\_NAME AS "Neighborhood",

NBHD\_TYPE AS "Neighborhood Type",

NBHD\_FEAT AS "Neighborhood Features",

WIFI AS "Wifi included",

PETS AS "Pets",

CHILDREN AS "Children",

KITCHEN AS "Kitchen Available",

COMMON\_AREA AS "Shared Common Area",

OWNER\_OCC AS "Owner Occupied",

TELEVISION AS "Television in Room",

SMOKING AS "Smoking allowed",

WASHER\_DRYER AS "Washer/Dryer Available",

POOL AS "Pool Available",

OTHER AS "Other Features"

FROM AVAILABLE\_ROOM AA

WHERE AA.TRAVELER\_ID = \*ACTIVE\_TRAVELER\*

AND AA.AVAIL\_CHECK\_DATE = (SELECT MAX(AVAIL\_CHECK\_DATE) AS MAX\_CHECK FROM AVAILABLE\_ROOM BB WHERE BB.TRAVELER\_ID = AA.TRAVELER\_ID GROUP BY BB.TRAVELER\_ID);

\*/

## /\*Procedure 4 – Room Price\*/

/\*

------------------Procedure to find the daily rate------------------------------

Purpose: Create a list of rooms and dates with the daily price of each room

on each date based on the ROOM\_ID that is prompted, the dates the

active traveler is interested in, and the active traveler\_id at run

time.

To create this list of rooms and dates we join the REGION\_RATE, REGION,

ADDRESS, and the ROOM table.

We initialize a variable that uses the date the traveler wants to start

their travel and use it as the loop index variable checking all dates

from the date they want to start their stay up until one day before

the traveler wants to end their stay. Then we find rates by look for

where the loop index variable (temp\_date\_start) to make sure it falls

between RATE\_BEGIN\_DATE and one month after the RATE\_BEGIN\_DATE. We

find the one month after date by using the ADD\_MONTHS function and

adding one month.

On each loop we append one row to the ROOM\_RATE and calculate the

daily rate as REGION\_RATE \* .9, CoHost policy is that room rates are

locked to 90% of the rate charged for the region the room is in.

Once finished the result is the detail view of the table that the

procedure to find available rooms builds. This is used to see each

daily price that makes up the room\_price in the reservation and

available\_room table.

Parameters

roomno: This is the ROOM\_ID that the active traveler would like to get more

detail about.

date\_want\_start: This is the date that the active traveler is interested in

starting their trip.

date\_want\_end: This is the date that the active traveler is interested in

ending their trip.

query\_trav: this is the active traveler id that is currently using the

website.

Exception Handling:

There are 2 exceptions handled by this procedure.

1. If the user has entered a date before today's date they are prompted to

enter a valid date.

2. If any other exception happens the ORA-20002 exception is thrown and the user

is given a brief message explaining that the traveler id or the room id does

not exist.

Output: This procedure appends data to the room price table.

\*/

CREATE OR REPLACE PROCEDURE ROOM\_PRICE

(roomno room.room\_id%type,

date\_want\_start reservation.reserv\_start%type,

date\_want\_end reservation.reserv\_end%type,

query\_trav employee.emp\_id%type)

as

temp\_date\_start reservation.reserv\_start%type;

temp\_rate region\_rate.region\_rate%type;

temp\_room room.room\_id%type;

temp\_date\_end reservation.reserv\_end%type;

temp\_trav employee.emp\_id%type;

BEGIN

DELETE ROOM\_RATE;

temp\_date\_start := date\_want\_start;

temp\_date\_end := date\_want\_end ;

IF TO\_DATE(temp\_date\_end) < TO\_DATE(SYSDATE)

THEN

raise\_application\_error

(-20009, 'Please enter a date greater than or equal to today''s date.');

ELSE

/\*While loop to move through all days that a traveler would be staying with a host.\*/

WHILE temp\_date\_end > temp\_date\_start

LOOP

SELECT roomno into temp\_room from dual;

SELECT query\_trav into temp\_trav from dual;

/\*Calculate each rate based on the date and price of that date\*/

SELECT (RR.REGION\_RATE \* .9) AS RATE\_CHARGED INTO temp\_rate FROM

REGION\_RATE RR

INNER JOIN REGION RE

ON RR.REG\_ID = RE.REG\_ID

INNER JOIN ADDRESS AA

ON TRIM(LOWER(AA.ADDRS\_COUNTY)) = TRIM(LOWER(RE.COUNTY))

INNER JOIN

ROOM RO

ON AA.ADDRS\_ID = RO.ADDRS\_ID

WHERE room\_id = temp\_room

and temp\_date\_start >= RR.RATE\_BEGIN\_DATE

and temp\_date\_start <= (ADD\_MONTHS(RR.RATE\_BEGIN\_DATE,1)-1);

/\*Insert the new values into the ROOM\_RATE table so that they can be displayed

in the front end\*/

INSERT INTO ROOM\_RATE (ROOM\_ID, TRAVELER\_ID, ROOM\_RATE, ROOM\_DATE, RUN\_DATE)

VALUES (TEMP\_ROOM, TEMP\_TRAV, TEMP\_RATE, temp\_date\_start, SYSDATE);

/\*iterate the looping vairable\*/

temp\_date\_start := temp\_date\_start + 1;

END LOOP;

END IF;

/\*Handle the no data found exception\*/

COMMIT;

Exception

When no\_data\_found Then

raise\_application\_error

(-20002, TEMP\_TRAV || ' or ' || TEMP\_ROOM ||' does not exist.');

END;

/

/\*Test Values used to verify procedure\*/

/\*

EXECUTE ROOM\_PRICE ('ROO000000010','30-DEC-15','05-JAN-16','EMP000000004');

COMMIT;

SELECT \* FROM ROOM\_RATE AA

WHERE AA.TRAVELER\_ID = \*ACTIVE TRAVELER\*

AND AA.RUN\_DATE = (SELECT MAX(RUN\_DATE) ROOM\_RATE BB WHERE BB.TRAVELER\_ID = AA.TRAVELER\_ID GROUP BY TRAVELER\_ID);

\*/