**ASSIGNMENT COVER SHEET**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course: BSc Computing (SE)** | | | | **Year: 2** | | **CSY2038** | |
| **Group Project** | | **Title:** Design, create and test methods for building an object relational database with useful data abstraction and automatizing useful processes in Pl/SQL | | | | | |
| Date due out: | Date due in: 13th Nov, 2022 | | Extension date: | | | | Extension agreed by: |
| **Team: 19**  **Group Members:**   1. Miraj Thapa 2. Rupak Upreti 3. Sakshyam Aryal 4. Suyog Kadariya | | | | | | **Tutor: Ankit Thapa** | |
| Student comment, specific request for feedback etc. | | | | | Marker’s General View of the work | | |

Video Demo Link:

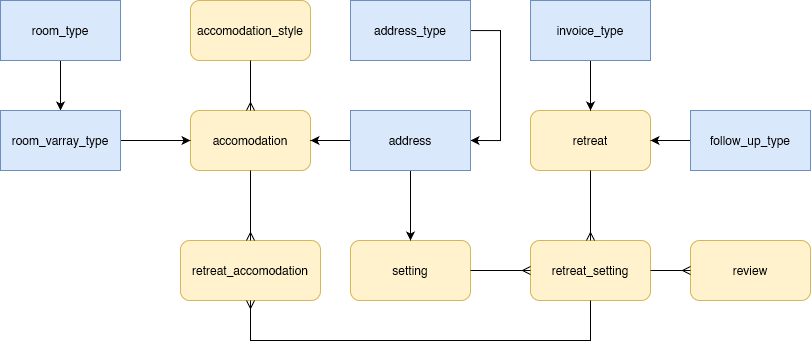
**Contents**

**Abstract**

In any system where data management is required, the implementation of a database is absolutely essential. In order to establish an object relational database with usable data abstraction and automate helpful procedures in PL/SQL, we designed and created methods for doing so. We, the team members, talked on how to organize files securely and appropriately as a collaborative project. First, we generated a skeleton table in accordance with the schema we had defined. All the tables, or UDTs, are included in our skeleton table. For this project, we have also created functions, triggers, and cursors as well as developed queries and procedures. By assigning responsibilities to each member independently, the four of us worked on distinct parts of the project. As part of our collaboration, we discussed each question and came up with answers to each issue. With the completion of this project, we hope to design, develop, and test techniques for creating object relational databases.

1. **Database Schema Design**

Schema design is the initial step in the database design process. The database schema refers to the overall layout of the database. It stands for the logical overview of the entire database architecture. The database schema that we designed is provided below.



*Fig: Database Schema Design*

1. **Skeleton Table Design**

To comprehend the database structure at its most fundamental level, a skeleton table is required. Based on the schema design, the skeleton table that follows shows all the specifics of the database tables, including their characteristics, keys, datatypes, constraints and default values.

| **Tables** | **attribute** | | **Key** | **Datatype** | **Constraints \**  **Defaults** |
| --- | --- | --- | --- | --- | --- |
| **Retreats** | retreat\_id | | *pk* | NUMBER(6) |  |
|  | retreat\_name | |  | VARCHAR2(30) | NOT NULL |
|  | invoice | |  | invoice\_type |  |
|  | follow\_up | |  | follow\_up\_type |  |
| **Settings** | setting\_id | | *pk* | NUMBER(6) |  |
|  | setting\_name | |  | VARCHAR2(30) | NOT NULL |
|  | address | |  | address\_type |  |
| **retreat\_settings** | | retreat\_setting\_id | *pk* | NUMBER(6) |  |
|  | setting\_id | | *FK* | NUMBER(6) | NOT NULL |
|  | retreat\_id | | *FK* | NUMBER(6) | NOT NULL |
| **accomodation\_styles** | accomodation\_style\_id | | *pk* | NUMBER(6) |  |
|  | accomodation\_style\_name | | *pk fk* | NUMBER(6) | NOT NULL |
| **accomodations** | accommodation\_id | | *pk* | NUMBER(6) |  |
|  | accommodation\_name | |  | VARCHAR2(50) | NOT NULL |
|  | room | |  | room\_varray\_type |  |
|  | no\_of\_room | |  | VARCHAR2(12) |  |
|  | address | |  | address\_type |  |
|  | description | |  | VARCHAR2(100) |  |
|  | accommodation\_style\_id | | *FK* | NUMBER(6) | NOT NULL |
| **retreat\_accommodations** | retreat\_setting\_id | | *pk*  *FK* | NUMBER(6) | NOT NULL |
|  | accommodation\_id | | *pk fk* | NUMBER(6) | NOT NULL |

UDT’s used are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **address\_type** | street |  | VARCHAR2(50) |  |
|  | city |  | VARCHAR2(50) |  |
|  | country |  | VARCHAR2(50) |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **follow\_up\_type** | start\_time |  | VARCHAR2(12) |  |
|  | duration |  | VARCHAR2(12) |  |
|  | date |  | DATE |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **invoice\_type** | invoice\_holder |  | NUMBER(6) |  |
|  | amount |  | VARCHAR2(10,2) |  |
|  | release\_date |  | DATE |  |
|  | due\_date |  | DATE |  |

Varray\_type used are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **room\_type** | room\_id |  | NUMBER(6) |  |
|  | capacity |  | VARCHAR2(10,2) |  |
|  | description |  | VARCHAR2(100) |  |

1. **Database tables**

To make clear of all the tables, separate tables have been made for good understandings. Attributes, keys, datatypes, constraints and default values are also included.

**Table no 1)** retreat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| retreats | retreat\_id | *pk* | NUMBER(6) | NOT NULL |
|  | retreat\_name |  | VARCHAR2(30) | NOT NULL |
|  | invoice |  | invoice\_type |  |
|  | follow\_up |  | follow\_up\_type |  |

**Table no 2)** setting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| settings | setting\_id | *pk* | NUMBER(6) | NOT NULL |
|  | setting\_name |  | VARCHAR2(30) | NOT NULL |
|  | address |  | address\_type |  |

**Table no 3)** retreat\_setting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| retreat\_settings | retreat\_setting\_id | *pk* | NUMBER(6) | NOT NULL |
|  | setting\_id | *fk* | NUMBER(6) | NOT NULL |
|  | retreat\_id | *f*k | NUMBER(6) | NOT NULL |

**Table no 4)** accommodation\_style

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| accommodation\_styles | accommodation\_style\_id | *pk* | NUMBER(6) | NOT NULL |
|  | accommodation\_style\_name | *pk, fk* | NUMBER(6) | NOT NULL |

**Table no 5)** accommodation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| accommodations | accommodation\_id | *pk* | NUMBER(6) | NOT NULL |
|  | accommodation\_name |  | VARCHAR2(50) |  |
|  | room |  | room\_varray\_type |  |
|  | no\_of\_rooms |  | VARCHAR2(12) |  |
|  | address |  | address\_type |  |
|  | description |  | VARCHAR2(100) |  |
|  | accomodation\_style\_id | *fk* | NUMBER(6) | NOT NULL |

**Table no 6)** retreat\_accommodation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE** | **ATTRIBUTE** | **KEY** | **DATATYPE** | **CONSTRAINTS/DEFAULTS** |
| retreat\_accommodations | retreat\_setting \_id | *pk, fk* | NUMBER(6) | NOT NULL |
|  | accommodation\_id | *pk, fk* | NUMBER(6) | NOT NULL |

1. **Functions, Sequences, Procedures, Triggers, Cursors and Queries**

Table below consists of all the functions, procedures, triggers and queries created. Table is divided into Name and description of the names.

|  |  |
| --- | --- |
| **NAME** | **DESCRIPTION** |
| Functions |  |
| *func\_avg\_rating* | *to get the average rating from the accomodations review* |
| *func\_total\_amount* | *to get the total amount of the accomodation* |
| *func\_number\_ho(p\_string IN VARCHAR2)* | *To get the string to the number* |
| Sequence |  |
| seq\_retreats | *starts with 10000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_addresses | *starts with 20000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_settings | *starts with 30000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_retreat\_settings | *starts with 40000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_reviews | *starts with 50000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_accomodation\_styles | *starts with 60000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_accomodations | *starts with 70000 and increments by 1 upto*  *999999999999999999999999999* |
| seq\_retreat\_accomodations | *starts with 80000 and increments by 1 upto*  *999999999999999999999999999* |
| **NAME** | **DESCRIPTION** |
| Procedures |  |
| *proc\_retreat\_name* | *procedure to list down all the retreat name with for loop* |
| *proc\_add\_retreat* | *procedure to insert into the retreats* |
| Triggers |  |
| *trig\_setting\_name\_ck* | *Setting name should not contain number* |
| *trig\_del\_retreat\_name* | *Fires message while delete retreat* |
| *trig\_duplicate\_no\_insert* | *No duplicate insert in accomodations* |
| Cursors |  |
| *proc\_cur\_del\_address* | *The cursor deletes an address from the address\_type where SQL%FOUND returns true or falseand is used for DML in Pl/SQL.* |
| *cur\_retreats* | *Shows the name and id of the retreats, helpful for directly inserting in procedure and fetchs all the values in loops.* |
| *check\_rating* | *Checks the Rating inputted in the parameter. If the rating do exists or not* |
| Queries | |
| *e.g., highest booking cost using functions* | |
|  | |
|  | |
|  | |
|  | |
|  | |

1. **Proposed Automation Strategy**
2. **Evidence of Additional Research**

In addition to the assignment briefs and project files we received, we had to conduct extensive study on database construction techniques before beginning the job. We conducted team research on many topics. We read a few research papers and forum posts from Oracle developers before having a team discussion. The "**Oracle Database Online Documentation 12c Release 1(12.1)**" and a few other articles listed in the reference section of this document were the most helpful ones we could find.

1. **Description along with screenshots**

Below contains all the function, triggers, procedures functionalities that describes all the functionalities along with the name.

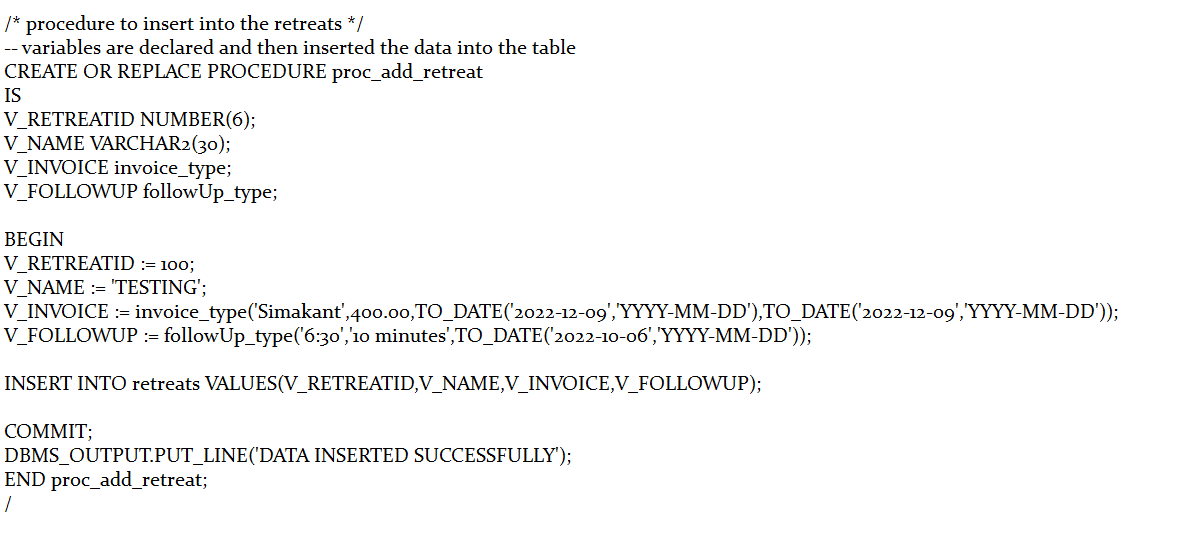
* 1. **Procedures**

Similar to database functions, procedures comprise SQL statements that are only used to carry out certain tasks. Procedures are mostly used for data retrieval, data retrieval, access control, and multiple parameter queries. Below table consists procedure name and description.

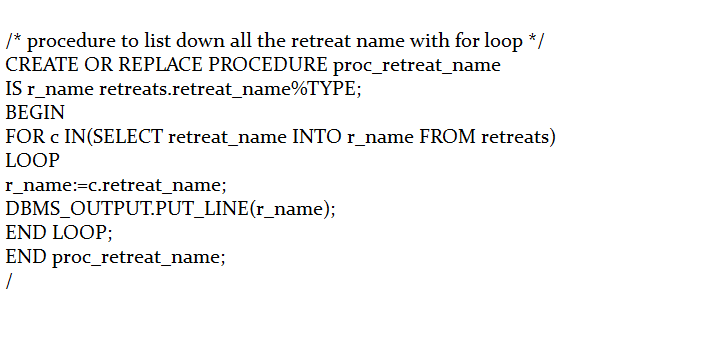
|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *proc\_retreat\_name* | *Gets the names of the retreats that are present in the programme. For getting the name we have selected the retreats table and read the data of their name using for loop. Similarly, we have printed the name when each loop executes.* |
| 2 | *proc\_add\_retreat* | *It inserts the given data in the retreat table. Here we have declared the attributes variable and then we initialized it. After initializing we had used insert query to add the data to the table.* |

* + 1. **Procedure functionalities**

1. *proc\_add\_retreat*



1. *proc\_retreat\_name*



* 1. **Functions**

A function is a collection of SQL statements that carry out a particular task. To make operations simpler and quicker, we have created three distinct functionalities. The table below lists the functions and their descriptions.

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *func\_avg\_rating* | *It gives the average all the accommodations currently present in the programme. Here we accessed all the rating attributes from the reviews table and gets the average from inbuilt function.* |
| 2 | *func\_total\_amount* | *It gives the total amount of money generated from the programme. Here we had accessed the invoice type and gets the amount and by using sum function we have got our total invoice amount.* |
| 3 | *func\_number\_ho(p\_string IN VARCHAR2)* | *It changes the string to number and returns number variable. This function is used in the trigger named*  *trig\_setting\_name\_ck to get the value of string to number.* |

* + 1. **Function functionalities**

1. *func\_avg\_rating*

Text

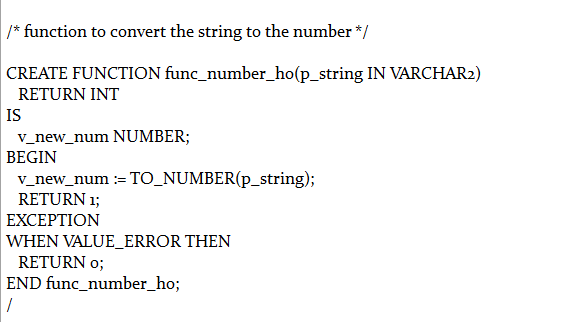
Description automatically generated

1. func\_total\_amount

Text

Description automatically generated with medium confidence

1. func\_number\_ho(p\_string IN VARCHAR2)



1. function\_one\_name

Add screenshot of the function.

* 1. **Cursors**

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *proc\_cur\_del\_address* | *The cursor deletes an address from the address\_type where SQL%FOUND returns true or false and is used for DML in Pl/SQL* |
| 2 | *cur\_retreats* | *Shows the name and id of the retreats, helpful for directly inserting in procedure and fetchs all the values in loops. This cursor is used to select all the and reurn the multiple rows.* |
| 3 | *check\_rating* | *.Checks the Rating inputted in the parameter. If the rating do exists or not. This cursor checks if the raing is given or not.* |

**7.4** **Triggers**

A trigger in a database is a piece of procedural code that is automatically performed out in response to specific events on a specific table or view in a database. We created 3 different triggers for specific events that is described in the table below.

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **DESCRIPTION** |
| 1 | *trig\_setting\_name\_ck* | *To check if the name of setting contains number and to throw error if the name contains number. To do so we inserted the value with number and it fired the message*  *20001,'ERROR! INVALID SETTING NAME! NAME CANNOT CONTAIN NUMBERS'* |
| 2 | *trig\_del\_retreat\_name* | *To throw the message that the retreat is deleted after the retreat is deleted. We added by inserting one retreat name and later delete it then it fires the trigger with message*  *YOU DELETED THE RETREAT* |
| 3 | *trig\_duplicate\_no\_insert* | *To throw message that no duplicate accommodation id, name can be created. We insert the accommodation value which was inserted earlier to fire trigger:*  *-20000, 'cannot create duplicate version of accomodation'* |

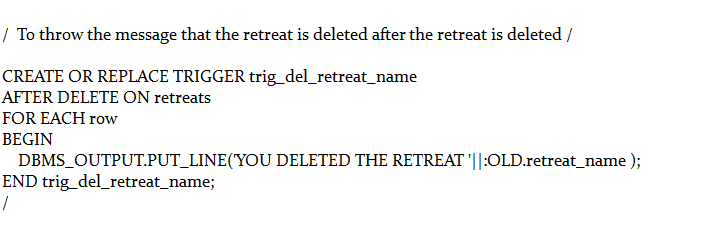
* + 1. **Trigger functionalities**

1. trig\_setting\_name\_ck

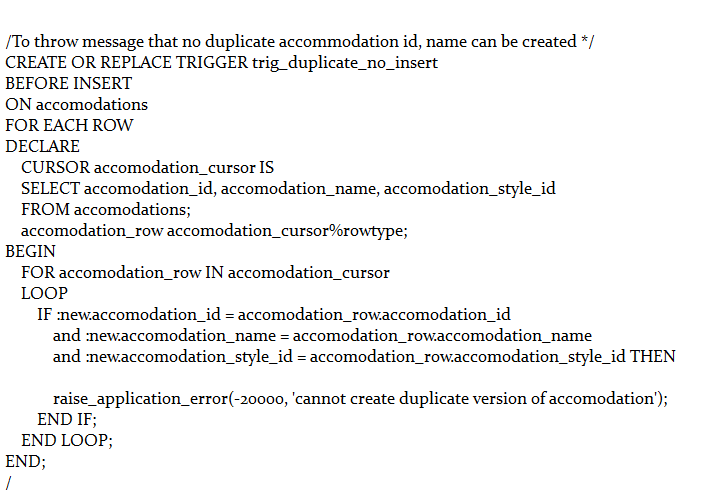
Graphical user interface, text, application, email

Description automatically generated

1. trig\_del\_retreat\_name



1. trig\_duplicate\_no\_insert



1. **Test Plan**

The testing phase of a project is one of the most fundamental stages. In the testing phase, we put our attention on investigation and discovery. All of the functions, procedures, sequences, cursors and triggers we developed were tested to make sure they all produced the desired outcome. We created a list of all the test procedures to be followed, serially numbered the IDs, and then ran the test. The table below displays the findings of our tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **TEST DESCRIPTION** | **EXPECTED RESULT** | **ACTUAL RESULT** | **ACTION** |
| 1 | *COLUMN review\_id HEADING ‘REVIEW’*  *COLUMN ROUND(rating) HEADING 'ROUND'*  *COLUMN FLOOR(rating) HEADING 'FLOOR'*  *COLUMN CEIL(rating) HEADING 'CEIL'*  *SELECT review\_id, ROUND(rating), FLOOR(rating),CEIL(rating) , rating*  *FROM reviews*  *WHERE rating = (*  *SELECT MAX(rating)*  *FROM reviews*  *);* | *REVIEW 5*  *ROUND 5*  *FLOOR 5*  *CEIL 5*  *RATING 5* | AS EXPECTED  (PASS) |  |
| 2 | COLUMN settting\_id HEADING ‘SettingId’  COLUMN address.street HEADING 'Street' FORMAT A30  COLUMN address.city HEADING 'City' FORMAT A20  COLUMN address.country HEADING 'Country' FORMAT A15  SELECT s.setting\_id,s.address.street, s.address.city, s.address.country  FROM settings s  WHERE s.address.city IN (  SELECT a.address.city  FROM accomodations a  WHERE a.address.country = ’NEPAL’  ); | SETTING\_ID 47  STREET 93 DEVKOTA STREET  CITY BIRATNAGAR  COUNTRY NEPAL | AS EXPECTED  (PASS) | - |
| 3 | COLUMN accomodation\_style\_id HEADING ‘AccomodationStyleID'  COLUMN accomodation\_style\_name HEADING ‘AccomodationStyleName’  COLUMN accomodation\_description HEADING ‘AccomodationDescription’  SELECT a.accomodation\_style\_id,a.accomodation\_style\_name,r.description  FROM accomodation\_styles a  JOIN accomodations r  ON a.accomodation\_style\_id = r.accomodation\_style\_id  WHERE a.accomodation\_style\_id=11; | ‘AccomodationStyleID' ‘AccomodationStyleName’  --------------------- --------------------  DESCRIPTION  --------------------------------------------------------------------------------  11 TREEHOUSE  ONE AND ONLY TREE ACCOMODATION IN BIHAR  11 TREEHOUSE  ONE AND ONLY TREE ACCOMODATION IN BIHAR  11 TREEHOUSE  GET GOOD SERVICE HERE | AS EXPECTED  (PASS) |  |
| 4 | COLUMN accomodation\_style\_id HEADING ‘AccomodationStyleID'  COLUMN accomodation\_style\_name HEADING ‘AccomodationStyleName’  COLUMN accomodation\_description HEADING ‘AccomodationDescription’  SELECT a.accomodation\_style\_id,a.accomodation\_style\_name,r.description  FROM accomodation\_styles a  JOIN accomodations r  ON a.accomodation\_style\_id = r.accomodation\_style\_id; | ‘AccomodationStyleID' ‘AccomodationStyleName’  --------------------- --------------------  DESCRIPTION  --------------------------------------------------------------------------------  10 LUXUARY  LUXUARY HOTEL OF KANPUR  11 TREEHOUSE  ONE AND ONLY TREE ACCOMODATION IN BIHAR  11 TREEHOUSE  GET GOOD SERVICE HERE  AccomodationStyleID' AccomodationStyleNa  --------------------- --------------------  DESCRIPTION  --------------------------------------------------------------------------------  11 TREEHOUSE  ONE AND ONLY TREE ACCOMODATION IN BIHAR | AS EXPECTED  (PASS) |  |
| 5 | COLUMN setting\_name HEADING ‘SettingName’  COLUMN DEREF(address) HEADING 'AddressTYPE’  SELECT setting\_name, DEREF(address)  FROM settings; | ‘SettingName'  ------------------------------  AddressTYPE  --------------------------------------------------------------------------------  WINTER WONDERLAND  ADDRESS\_TYPE('BANEPA ROAD', 'BANEPA', 'NEPAL')  PLANETARY LEY LINES  ADDRESS\_TYPE('PANAUTI ROAD', 'PANAUTI', 'NEPAL')  COSTAL  ADDRESS\_TYPE('165 PRAKSHPUR', 'JHAPA', 'NEPAL')  SettingName'  ------------------------------  AddressTYPE  --------------------------------------------------------------------------------  LAKESIDE  ADDRESS\_TYPE('93 DEVKOTA STREET', 'BIRATNAGAR', 'NEPAL')  WOODLAND | AS EXPECTED  (PASS) |  |
| 6 | COLUMN accomodation\_id HEADING ‘AccomodationID'  COLUMN accomodation\_name HEADING ‘AccomodationName’  COLUMN description HEADING ‘Description’  COLUMN room\_number HEADING ‘RoomNo’  SELECT a.accomodation\_id, a.accomodation\_name, a.description, r.room\_number, r.description  FROM accomodations a, TABLE(a.room) r  WHERE r.room\_number = 501; | ‘AccomodationID' ‘AccomodationName'  ---------------- --------------------------------------------------  Description'  --------------------------------------------------------------------------------  Roo  ----  `Description'  --------------------------------------------------------------------------------  4 SNG  GET GOOD SERVICE HERE  501  WELL ORGANIZED | AS EXPECTED  (PASS) |  |
| 7 | EXEC proc\_add\_retreat ; | DATA INSERTED SUCCESSFULLY  PL/SQL procedure successfully completed. | AS EXPECTED  (PASS) |  |
| 8 | EXEC proc\_retreat\_name; | squats  push up  Abdominal Crunches  squats  Burpees  PL/SQL procedure successfully completed. | AS EXPECTED  (PASS) |  |
| 9 | EXEC proc\_cur\_del\_address('JHAPA'); | CITY "in\_city" WAS DELETED!  PL/SQL procedure successfully completed. | AS EXPECTED  (PASS) |  |
| 10 | EXEC check\_rating(3); | RATING EXISTS  PL/SQL procedure successfully completed. | AS EXPECTED  (PARTIAL) | If not exist then wont show doesn’t exist |
| 11 | INSERT INTO settings(setting\_id,setting\_name)  VALUES(900,22);  DELETE FROM settings WHERE setting\_id=900; | 20001,'ERROR! INVALID SETTING NAME! NAME CANNOT CONTAIN NUMBERS' | AS EXPECTED  (PASS) | putting setting name as 22 to fire the trigger |
| 12 | INSERT INTO retreats(retreat\_id,retreat\_name)  VALUES (77,'BEND OVER ROW');  DELETE FROM retreats WHERE retreat\_name='BEND OVER ROW'; | you deleted the retreat | NOT RUN | Inserting value to delete in order to fire trigger.Shows one row deleted only |
| 13 | INSERT INTO accomodations(accomodation\_id,accomodation\_name,accomodation\_style\_id)  VALUES(4,'SNG',11); | 20000, 'cannot create duplicate version of accomodation' | AS EXPECTED  (PASS) | Adding same value in order to fire trigger. |

1. **References**
2. Drew, A. and Paapanen, E. (2003) *Using Triggers*, *9 using triggers*. Oracle Corporation. [Available at: https://docs.oracle.com/cd/B13789\_01/appdev.101/b10795/adfns\_tr.htm](Available%20at:%20https:/docs.oracle.com/cd/B13789_01/appdev.101/b10795/adfns_tr.htm) (Accessed: October 15, 2022).
3. Sharan, A. (2018) *Cursors in PL/SQL*, *GeeksforGeeks*. GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/cursors-in-pl-sql/> (Accessed: October 15, 2022).
4. Amazon Web Services, Inc. (1989) *Oracle Procedures and Functions and MySQL Stored Procedures*, *Amazon*. Det Danske Missionsselskab. Available at: <https://docs.aws.amazon.com/dms/latest/oracle-to-aurora-mysql-migration-playbook/chap-oracle-aurora-mysql.sql.stored.html> (Accessed: October 23, 2022).
5. Oracle Tutorial (2020) *Oracle create sequence explained by practical examples*, *Oracle Tutorial*. Available at: <https://www.oracletutorial.com/oracle-sequence/oracle-create-sequence/> (Accessed: November 03, 2022).
6. Pedamkar, P. (2021) *Function in oracle: How do functions in Oracle work with examples*, *EDUCBA*. Available at: [https://www.educba.com/function-in-oracle/](https://www.educba.com/function-in-oracle/%20) (Accessed: November 03, 2022).
7. **Appendix**