BSc. COMPUTING

CSY2038|DATABASES 2

**GROUP 25**

|  |  |
| --- | --- |
| **Names of group members** | **UoN Id** |
| Shreejan Regmi | 18417132 |
| Neetu Kumari Das | 18406487 |
| Saurav Khatiwada | 18413649 |
| Hira Bahadur Rana Kauchha | 18413703 |

**TABLE OF CONTENTS**

[**1.** **INTRODUCTION** 4](#_Toc7606606)

[**1.1.** **DESIGN** 4](#_Toc7606607)

[**1.1.1.** **ENTITY REALTIONSHIP DIAGRAM** 4](#_Toc7606608)

[**1.1.2.** **CHOSEN ENTITY RELATIONSHIP DIAGRAM** 5](#_Toc7606609)

[**1.1.3.** **SCHEMA** 6](#_Toc7606610)

[**1.1.4.** **TABLE SPECIFICATION** 16](#_Toc7606611)

[**2.** **AUTOMATION STRATEGY** 19](#_Toc7606612)

[**3.** **TESTING** 23](#_Toc7606613)

[**3.1.** **TEST PLAN** 23](#_Toc7606614)

[**3.2.** **TEST CASES** 24](#_Toc7606615)

**TABLE OF FIGURES**

[Fig 1. Provided Entity Relationship Diagram 4](#_Toc7606497)

[Fig 2. Selected ERD 5](#_Toc7606498)

[Fig 3. Proposed Schema 6](#_Toc7606499)

# **INTRODUCTION**

The assessment is to build an object relational database by designing, creating, testing and document methods with the useful data abstraction and automated processes using PL/SQL. Those includes tables, user-defined datatypes(UDTs), collection types, procedures, functions, triggers and cursors.

## **DESIGN**

### **ENTITY REALTIONSHIP DIAGRAM**

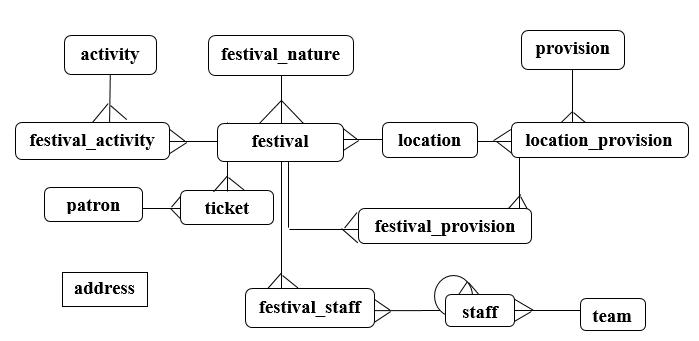


Fig 1. Provided Entity Relationship Diagram

The group had to select five entities from the above ERD, propose schema and table specification to build an object relational database using complex data abstraction and automated processes using PL/SQL.

### **CHOSEN ENTITY RELATIONSHIP DIAGRAM**

After analyzing the provided ERD, the group selected the five entities and they are festival\_nature, location, festival, staff and festival\_staff as shown in the above ERD.

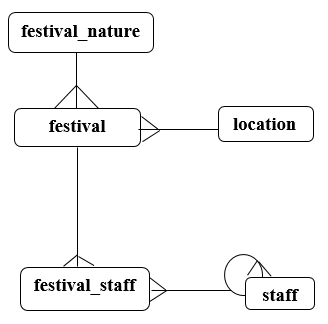


Fig 2. Selected ERD

### **SCHEMA**

After the analysis of selected ERD and group discussion, following schema was proposed.

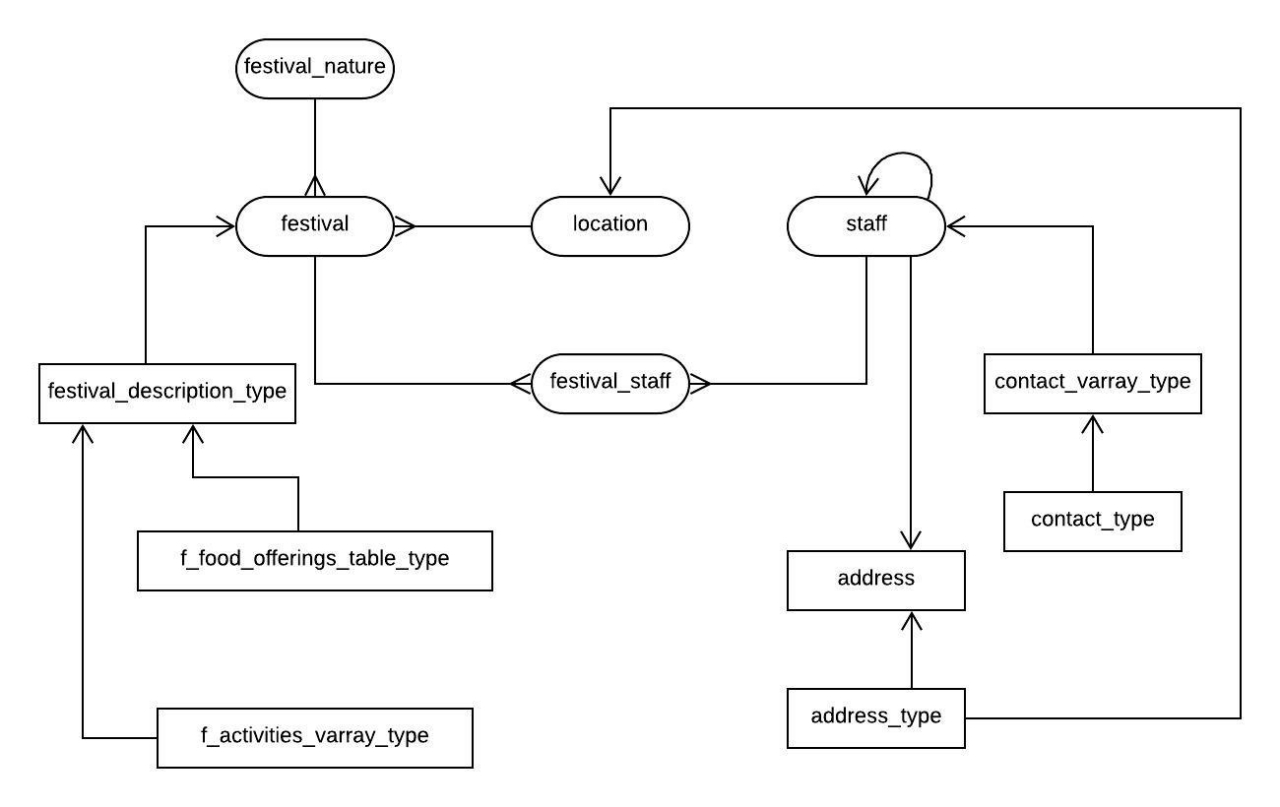


Fig 3. Proposed Schema

The details of the schema are as follows:

Number of Tables: 5

Number of Primary Keys: 3

Number of Compound Primary Keys: 2

Number of Foreign Keys: 4

Number of Compound Foreign Keys: 1

Number of Sequences: 3

Number of Unique Constraints: 1

Number of Check Constraints: 11

Number of Procedures: 8

Number of Procedures for functions: 6

Number of Functions: 7

Number of Triggers: 6

Number of Procedures for Cursors: 3

Number of Packages: 1

Number of Views: 1

#### **Entity Description**

|  |  |
| --- | --- |
| **ENTITY** | **DESCRIPTION** |
| festival\_nature | Natures of festivals e.g. Art, Music |
| Location | Details of the locations where festivals are organized, such as address |
| Festival | Details of festivals and their activities carried out |
| Staff | Details of staff |
| festival\_staff | Details of staff related to organizing the respective festivals |

#### **User-defined Types(UDTs) Description**

|  |  |
| --- | --- |
| **OBJECT TYPE/ OBJECT TABLE** | **DESCRIPTION** |
| address\_type | Contains street, city and country |
| contact\_type | Contains phone and email |
| contact\_varray\_type | Is of contact\_type |
| f\_food\_offerings\_table\_type | Stores the details of food catered in the festivals |
| f\_activities\_varray\_type | Stores the activities of the festivals |
| festival\_description\_type | Stores the details of the festivals which includes running time, f\_food\_offerings\_table\_type and f\_activities\_varray\_type |
| Addresses | Stores the street, city and country of address\_type |

#### **Creation Order**

Order of creation of types

* + - * address\_type
      * contact\_type
      * contact\_varray\_type
      * f\_food\_offerings\_table\_type
      * f\_activities\_varray\_type
      * festival\_description\_type

`

Order of creation of tables

* addresses
* festival\_natures
* locations
* festivals
* staff
* festival\_staff

Order of creation of sequences

|  |  |
| --- | --- |
| **SEQUENCES** | **TABLES** |
| seq\_f\_nature\_id | festival\_natures |
| seq\_location\_id | locations |
| seq\_staff\_id | Staff |

Order of creation of primary keys

|  |  |
| --- | --- |
| **PRIMARY KEYS** | **TABLES** |
| pk\_festival\_natures | festival\_natures |
| pk\_locations | locations |
| pk\_festivals | festivals |
| pk\_staff | staff |
| pk\_festival\_staff | festival\_staff |

Order of creation of foreign keys

|  |  |
| --- | --- |
| **FOREIGN KEYS** | **TABLES** |
| fk\_f\_festival\_natures | festivals |
| fk\_f\_locations | festivals |
| fk\_fs\_staff | staff |
| fk\_fs\_festivals | festival\_staff |
| fk\_s\_staff | festival\_staff |

Order of creation of unique constraints

uc\_festival\_nature\_name

Order of creation of check constraints

|  |  |
| --- | --- |
| **CHECK CONSTRAINTS** | **TABLES** |
| ck\_f\_nature\_name | festival\_natures |
| ck\_festival\_name | festivals |
| ck\_location\_name\_street | locations |
| ck\_location\_name\_city | locations |
| ck\_location\_name\_country | locations |
| ck\_staff\_fname | staff |
| ck\_staff\_lname | staff |
| ck\_s\_gender | staff |
| ck\_address\_street | addresses |
| ck\_address\_city | addresses |
| ck\_address\_country | addresses |

Order of creation of procedures

|  |  |
| --- | --- |
| **PROCEDURES** | **TABLES** |
| proc\_insert\_f\_natures | festival\_natures |
| proc\_insert\_locations | locations |
| proc\_insert\_staff | staff |
| proc\_update\_staff\_salary | staff |
| proc\_delete\_staff | staff |
| proc\_out\_param | staff |
| proc\_out\_in\_param | festivals |
| proc\_insert\_history | history |

Order of creation of functions

|  |  |
| --- | --- |
| **FUNCTIONS** | **TABLES** |
| func\_count\_festivals | festivals |
| func\_make\_username | staff |
| func\_staff\_name | staff |
| func\_staff\_salary | staff |
| func\_highest\_ticket\_price | festivals |
| func\_calculate\_staff\_age | staff |
| func\_calc\_retiring | staff |

Order of creation of procedures for functions

|  |  |
| --- | --- |
| **PROCEDURES** | **TABLES** |
| proc\_count\_festivals | festivals |
| proc\_output\_username | staff |
| proc\_output\_staff\_data | staff |
| proc\_output\_ticket\_price | festivals |
| proc\_test\_calc\_age | staff |
| proc\_calc\_retiring | staff |

Order of creation of triggers

|  |  |
| --- | --- |
| **TRIGGERS** | **TABLES** |
| trig\_staff\_dob\_ck | staff |
| trig\_locations\_message | locations |
| trig\_combined | staff |
| trig\_schema\_level | schema |
| trig\_database\_login | database |
| trig\_database\_log\_off | database |

Order of creation of procedures for cursors

|  |  |
| --- | --- |
| **PROCEDURES** | **TABLES** |
| proc\_imp\_del\_staff | staff |
| proc\_exp\_count\_festivals | festivals |
| proc\_exp\_show\_staff | staff |

Order of creation of views

view\_staff\_age\_group

Order of creation of package

pkg\_leader\_details

#### **Order of Drops**

Foreign Keys Drops

* fk\_f\_festival\_natures
* fk\_f\_locations
* fk\_fs\_staff
* fk\_fs\_festivals
* fk\_s\_staff

Unique Constraints Drops

* uc\_festival\_nature\_name

Check Constraint Drops

* ck\_f\_nature\_name
* ck\_festival\_name
* ck\_location\_name\_street
* ck\_location\_name\_city
* ck\_location\_name\_country
* ck\_s\_fname
* ck\_s\_lname
* ck\_s\_gender
* ck\_address\_street
* ck\_address\_city
* ck\_address\_country

Primary Keys Drops

* pk\_festival\_natures
* pk\_locations
* pk\_festivals
* pk\_staff
* pk\_festival\_staff

Sequence Drops

* seq\_f\_nature\_id
* seq\_location\_id
* seq\_staff\_id

Triggers drop

* trig\_staff\_dob\_ck
* trig\_locations\_message
* trig\_combined
* trig\_schema\_level
* trig\_database\_login
* trig\_database\_logoff

Tables Drops

* festival\_staff
* addresses
* festivals
* staff
* festival\_natures
* locations
* history

Types Drops

* festival\_description\_type
* f\_activities\_varray\_type
* f\_food\_offerings\_table\_type
* contact\_varray\_type
* contact\_type
* address\_type

Functions drop

* func\_count\_festivals
* func\_make\_username
* func\_staff\_name
* func\_staff\_salary
* func\_highest\_ticket\_price
* func\_calculate\_staff\_age
* func\_calc\_retiring

Procedures Drops

* proc\_insert\_f\_natures
* proc\_insert\_locations
* proc\_insert\_staff
* proc\_update\_staff\_salary
* proc\_delete\_staff
* proc\_out\_param
* proc\_out\_in\_param
* proc\_insert\_history

Procedures drops used for functions

* proc\_count\_festivals
* proc\_output\_username
* proc\_output\_staff\_data
* proc\_output\_ticket\_price
* proc\_test\_calc\_age
* proc\_calc\_retiring

Procedures drops used for cursors

* proc\_imp\_del\_staff
* proc\_exp\_count\_festivals
* proc\_exp\_show\_staff

Views drop

* view\_staff\_age\_group

Packages drop

* pkg\_leader\_details

### **TABLE SPECIFICATION**

|  |  |  |
| --- | --- | --- |
| **TYPES** | **ATTRIBUTES** | **DATATYPE** |
| **address\_type** | street | VARCHAR2(20) |
|  | city | VARCHAR2(20) |
|  | country | VARCHAR2(20) |
|  |  |  |
| **contact\_type** | phone | VARCHAR2(15) |
|  | email | VARCHAR2(20) |
|  |  |  |
| **contact\_varray\_type** |  | **contact\_type** |
|  |  |  |
| **f\_food\_offerings\_table\_type** |  | VARCHAR2(20) |
|  |  |  |
| **f\_activities\_varray\_type** |  | VARCHAR2(20) |
|  |  |  |
| **festival\_description\_type** | activities | **f\_activities\_varray\_type** |
|  | running time | VARCHAR2(10) |
|  | Food | **f\_food\_offerings\_table\_type** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TABLES** | **ATTRIBUTES** | **DATATYPE** | **CONSTRAINT/ DEFAULT** | **KEY** | **SEQUENCE** |
| **addresses** | **address\_type:** |  |  |  |  |
|  | street | VARCHAR2(20) | UPPER |  |  |
|  | city | VARCHAR2(20) | UPPER |  |  |
|  | country | VARCHAR2(20) | UPPER |  |  |
|  | | | | | |
| **festival\_natures** | f\_nature\_id | NUMBER(3) |  | *PK* | seq\_f\_nature\_id |
|  | f\_nature\_name | VARCHAR2(15) | UNIQUE, UPPER |  |  |
|  | | | | | |
| L**ocations** | location\_id | NUMBER(3) |  | *PK* | seq\_location\_id |
|  | location\_name( | **address\_type:** |  |  |  |
|  | street | VARCHAR2(20) | UPPER |  |  |
|  | city | VARCHAR2(20) | UPPER |  |  |
|  | country) | VARCHAR2(20) | UPPER |  |  |
|  | | | | | |
| **Festivals** | f\_nature\_id\* | NUMBER(3) | NOT NULL | *PK, FK* |  |
|  | location\_id \* | NUMBER(3) | NOT NULL | *PK, FK* |  |
|  | festival\_name | VARCHAR2(25) | UPPER |  |  |
|  | festival\_date | DATE |  |  |  |
|  | ticket\_price | NUMBER(6,2) |  |  |  |
|  | festival\_details | **festival\_description\_type** |  |  |  |
|  | | | | | |
| **Staff** | staff\_id | NUMBER(3) |  | *PK* | seq\_staff\_id |
|  | staff\_fname | VARCHAR2(15) | UPPER |  |  |
|  | staff\_lname | VARCHAR2(15) | UPPER |  |  |
|  | leader\_id\* | NUMBER(3) | NOT NULL | *FK* |  |
|  | staff\_salary | NUMBER(7,2) |  |  |  |
|  | gender | CHAR | CHECK IN (‘F’, ‘M’, ‘O’)  DEFAULT ‘F’ |  |  |
|  | staff\_dob | DATE |  |  |  |
|  | staff\_image | BLOB |  |  |  |
|  | staff\_contact | **contact\_varray\_type** |  |  |  |
|  | staff\_address | **REF of address\_type** |  |  |  |
|  | | | | | |
| **festival\_staff** | staff\_id \* | NUMBER(3) | NOT NULL | *PK, FK* |  |
|  | f\_nature\_id\*\* | NUMBER(3) | NOT NULL | *PK, FK* |  |
|  | location\_id \*\* | NUMBER(3) | NOT NULL | *PK, FK* |  |

# **AUTOMATION STRATEGY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Procedural Element type | Name | Purpose | Parameter (with types) | Return items |
| Procedure | * proc\_insert\_f\_natures | * to insert festival\_nature\_name into festival\_natures table | * in\_f\_nature\_name |  |
| Procedure | * proc\_insert\_locations | * to insert location into locations table. | * no parameter |  |
| Procedure | * proc\_insert\_staff | * to insert staff into staff table. | * in\_fname, in\_lname |  |
| Procedure | * proc\_update\_staff\_salary | * to update staff’s salary. | * in\_staff\_id, in\_staff\_salary |  |
| Procedure | * proc\_delete\_staff | * to delete staff | * in\_staff\_id |  |
| Procedure | * proc\_out\_param | * to display set full name of staff identified by staff id provided as argument | * in\_staff\_id, out\_staff\_name |  |
| Procedure | * proc\_out\_in\_param\_\_\_ | * to guess the highest ticket price of festivals | * in\_out\_max\_price |  |
| Function | * func\_count\_festivals | * to count the number of festivals in festival table |  |  |
| Procedure | * proc\_count\_festivals | * to use function   func\_count\_festivals |  |  |
| Function | * func\_make\_username | * to create usernames using staff's first and lastname | * in\_staff\_id |  |
| Procedure | * proc\_output\_username | * to output the username generated by function func\_make\_username | * in\_staff\_id |  |
| Function | * func\_staff\_name | * to return full staff name that takes staff id as parameter | * in\_staff\_id |  |
| Function | * func\_staff\_salary | * to return staff salary that takes staff id as parameter | * in\_staff\_id |  |
| Procedure | * proc\_output\_staff\_data | * to output name and salary of the staff identified by staff id from parameter | * in\_staff\_id |  |
| Function | * func\_highest\_ticket\_price | * to return highest ticket price festival |  |  |
| Procedure | * proc\_output\_ticket\_price | * to output the highest ticket price festival using function func\_highest\_ticket\_price |  |  |
| Trigger | * trig\_staff\_dob\_ck | * to check birth date of staff |  |  |
| Trigger | * trig\_locations\_message | * to output message of insertion successful after insert on locations table |  |  |
| Function | * func\_calculate\_staff\_age | * to calculate the age of staff with DOB as parameter | * in\_staff\_dob |  |
| Procedure | * proc\_test\_calc\_age | * to test function func\_calculate\_staff\_age | * in\_staff\_dob, in\_staff\_firstname |  |
| Function | * func\_calc\_retiring | * to calculate days to retire | * in\_staff\_dob |  |
| Procedure | * proc\_calc\_retiring | * to test function func\_calc\_retiring | * in\_staff\_dob |  |
| Trigger | * trig\_combined |  |  |  |

# **TESTING**

## **TEST PLAN**

## **TEST CASES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N. | Description | Expected  Output | Screenshots of Actual Output | Results |
| Procedure 1 | | | | |
| 1. | Creating 'proc\_insert\_f\_natures' with parameters 'in\_f\_nature\_name' to insert into festival\_natures table   * CREATE OR REPLACE PROCEDURE proc\_insert\_f\_natures(in\_f\_nature\_name festival\_natures.f\_nature\_name%TYPE) IS   BEGIN  INSERT INTO festival\_natures (f\_nature\_id, f\_nature\_name)  VALUES (seq\_f\_nature\_id.NEXTVAL, in\_f\_nature\_name);  DBMS\_OUTPUT.PUT\_LINE('INSERTION SUCCESSFUL');  END proc\_insert\_f\_natures;  /  SHOW ERRORS; | Procedure created.  No errors. |  | Successful |
| Testing Procedure | | | | |
|  | Querying the festival\_natures table   * SELECT f\_nature\_id, f\_nature\_name FROM festival\_natures; | 5 rows  selected. |  |  |
|  | Executing procedure proc\_param\_insert\_f\_natures   * EXECUTE proc\_insert\_f\_natures('ART'); | Insertion Successful |  |  |
|  | Querying the festival\_natures table to check if procedure is executed properly   * SELECT f\_nature\_id, f\_nature\_name FROM festival\_natures; | 6 rows selected. |  |  |
| Procedure 2 | | | | |
| 2. | Creating procedure proc\_insert\_locations to insert into locations table   * CREATE OR REPLACE PROCEDURE proc\_insert\_locations IS   vc\_street VARCHAR2(20):='4128 WHITEMAN';  vc\_city locations.location\_name.city%TYPE := 'PEMBERTON';  vc\_country locations.location\_name.country%TYPE:='NEW ZEALAND';  BEGIN  INSERT INTO locations(location\_id, location\_name)  VALUES (seq\_location\_id.NEXTVAL, address\_type(vc\_street, vc\_city, vc\_country));  DBMS\_OUTPUT.PUT\_LINE('INSERTION SUCCESSFUL');  END proc\_insert\_locations;  /  SHOW ERRORS; | Procedure created.  No errors. |  | Successful |
| Testing Procedure | | | | |
|  | Querying locations table before executing   * SELECT l.location\_id, l.location\_name.street Street, l.location\_name.city City, l.location\_name.country Country   FROM locations l; | 9 rows selected. |  |  |
|  | Executing procedure proc\_insert\_locations   * EXECUTE proc\_insert\_locations; | Insertion Successful |  |  |
|  | Querying locations table after executing   * SELECT l.location\_id, l.location\_name.street Street, l.location\_name.city City, l.location\_name.country Country   FROM locations l; | 10 rows selected. |  |  |
| Procedure 3 | | | | |
| 3. | Creating prodedure 'proc\_insert\_staff' to insert into staff table   * CREATE OR REPLACE PROCEDURE proc\_insert\_staff(in\_fname staff.staff\_fname%TYPE, in\_lname staff.staff\_lname%TYPE, in\_l\_id staff.leader\_id%TYPE, in\_salary staff.staff\_salary%TYPE ,in\_g staff.gender%TYPE, in\_address\_street addresses.street%TYPE) IS   vc\_contact contact\_varray\_type := contact\_varray\_type(contact\_type('+9779877622432', 'LUPIN@GMAIL.COM'), contact\_type('+9778977654320', 'LUPIN@YAHOO.COM'));  BEGIN  INSERT INTO staff(staff\_id, staff\_fname, staff\_lname, leader\_id, staff\_salary ,gender, staff\_contact, staff\_address)  SELECT seq\_staff\_id.NEXTVAL, in\_fname , in\_lname, in\_l\_id, in\_salary ,in\_g , vc\_contact, REF(a)  FROM addresses a  WHERE a.street= in\_address\_street; DBMS\_OUTPUT.PUT\_LINE('INSERTION SUCCESSFUL');  END proc\_insert\_staff;  /  SHOW ERRORS; | Procedure created.  No errors. |  | Successful |
| Testing Procedure | | | | |
|  | Querying staff table before executing procedure proc\_insert\_staff   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff; | 5 rows selected. |  |  |
|  | Executing procedure proc\_insert\_staff   * EXECUTE proc\_insert\_staff('LUPIN', 'DAHAL', 1, 1200.5 ,'M', '12 SAINT ROAD'); | Insertion Succcessful |  |  |
|  | Querying staff table after executing procedure proc\_insert\_staff   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff; | 6 rows selected. |  |  |
| Procedure 4 | | | | |
| 4. | Creating procedure 'proc\_update\_staff\_salary' with user\_defined exception to update staff's salary   * CREATE OR REPLACE PROCEDURE proc\_update\_staff\_salary(in\_gender staff.gender%TYPE) IS   vn\_count NUMBER;  update\_salary EXCEPTION;  no\_update EXCEPTION;  BEGIN  SELECT COUNT(staff\_id)  INTO vn\_count  FROM staff  WHERE gender = in\_gender;  IF vn\_count < 4 THEN  RAISE update\_salary;  ELSE  RAISE no\_update;  END IF;  EXCEPTION  WHEN update\_salary THEN  UPDATE staff  SET staff\_salary = staff\_salary \* 1.1  WHERE gender = in\_gender;  DBMS\_OUTPUT.PUT\_LINE('UPDATE SUCCESSFUL');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('NO UPDATES EXECUTED');  END;  /  SHOW ERRORS; | Procedure created.  No errors. |  | Successful |
| Testing Procedure | | | | |
|  | Querying staff details before executing procedure   * SELECT s.staff\_id, s.staff\_fname, s.staff\_salary FROM staff s WHERE s.gender = 'F'; | 2 rows selected |  |  |
|  | Executing procedure proc\_update\_staff\_salary   * EXECUTE proc\_update\_staff\_salary('M'); | Update Successful |  |  |
|  | Querying staff details after executing procedure   * SELECT s.staff\_id, s.staff\_fname, s.staff\_salary FROM staff s WHERE s.gender = 'F'; | 2 rows selected |  |  |
|  | Querying staff details before executing procedure   * SELECT s.staff\_id, s.staff\_fname, s.staff\_salary FROM staff s WHERE s.gender = 'M'; | 4 rows selected |  |  |
|  | Executing procedure proc\_update\_staff\_salary   * EXECUTE proc\_update\_staff\_salary('M'); | No updates executed |  |  |
|  | Querying staff details after executing procedure   * SELECT s.staff\_id, s.staff\_fname, s.staff\_salary FROM staff s WHERE s.gender = 'M'; | 4 rows selected |  |  |
| Procedure 5 | | | | |
| 5. | Creating procedure 'proc\_delete\_staff' to delete staff   * CREATE OR REPLACE PROCEDURE proc\_delete\_staff(in\_staff\_id staff.staff\_id%TYPE) IS   BEGIN  DELETE FROM staff  WHERE staff\_id=in\_staff\_id;  DBMS\_OUTPUT.PUT\_LINE('DELETION SUCCESSFUL');  END proc\_delete\_staff;  /  SHOW ERRORS; | Procedure created.  No errors |  | Successful |
| Testing Procedure | | | | |
|  | Querying staff details before executing procedure   * SELECT s.staff\_id, s.staff\_fname, s.staff\_salary FROM staff s WHERE s.staff\_id=6;   --OUTPUT:  -- STAFF\_ID STAFF\_FNAME STAFF\_SALARY  -- ---------- --------------- ------------  -- 6 LUPIN 1200.5 | 1 row selected |  |  |
|  | Executing procedure proc\_delete\_staff   * EXECUTE proc\_delete\_staff(6); | Deletion Successful |  |  |
|  | Querying staff details after executing procedure   * SELECT s.staff\_id, s.staff\_fname, s.staff\_salary FROM staff s WHERE s.staff\_id=6; | No rows selected |  |  |
| Procedure 6 | | | | |
| 6. | Creating procedure with ‘out’ parameter and predefined exception supported by TimesTen to display set full name of staff identified by staff id provided as argument   * CREATE OR REPLACE PROCEDURE proc\_out\_param(in\_staff\_id staff.staff\_id%TYPE, out\_staff\_name OUT VARCHAR2) AS   BEGIN  SELECT staff\_fname||' '||staff\_lname  INTO out\_staff\_name  FROM staff  WHERE staff\_id=in\_staff\_id;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  out\_staff\_name := '(NO ROWS FOUND)';  END proc\_out\_param;  /  SHOW ERRORS; | Procedure created.  No errors. |  | Successful |
| Testing Procedure | | | | |
|  | Checking the creation of procedure proc\_out\_param   * SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_NAME= 'PROC\_OUT\_PARAM'; | 1 row selected |  |  |
|  | Creating anonymous block to test procedure proc\_out\_param   * DECLARE   vc\_staff\_name VARCHAR2(30);    BEGIN  proc\_out\_param(1, vc\_staff\_name);  DBMS\_OUTPUT.PUT\_LINE(vc\_staff\_name);  END;  / | Aayush Moktan |  |  |
|  | Testing exception handling of proc\_out\_param   * DECLARE   vc\_staff\_name VARCHAR2(30);    BEGIN  proc\_out\_param(15, vc\_staff\_name);  DBMS\_OUTPUT.PUT\_LINE(vc\_staff\_name);  END;  / | No rows found |  |  |
| Procedure 7 | | | | |
| 7. | Create procedure with a parameter that acts as both in and out procedure to guess the highest ticket price of festivals   * CREATE OR REPLACE PROCEDURE proc\_out\_in\_param(in\_out\_max\_price IN OUT festivals.ticket\_price%TYPE) AS   BEGIN  SELECT MAX(ticket\_price)  INTO in\_out\_max\_price  FROM festivals;  END proc\_out\_in\_param;  /  SHOW ERRORS; | Procedure created.  No errors. |  | Successful |
| Testing Procedure | | | | |
|  | Checking the creation of procedure proc\_out\_in\_param   * SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_NAME= 'PROC\_OUT\_IN\_PARAM'; | 1 row selected |  |  |
|  | Creating anonymous block to test procedure proc\_out\_in\_param   * DECLARE   vn\_ticket\_price festivals.ticket\_price%TYPE;    BEGIN  DBMS\_OUTPUT.PUT\_LINE('GAME-GUESS THE HIGHEST TICKET PRICE');  vn\_ticket\_price:=&price;  DBMS\_OUTPUT.PUT\_LINE('YOUR INPUT PRICE IS '||vn\_ticket\_price);  proc\_out\_in\_param(vn\_ticket\_price);  DBMS\_OUTPUT.PUT\_LINE('THE HIGHEST TICKET PRICE IS '||vn\_ticket\_price);  END;  / |  |  |  |
| Functions | | | | |
| 1. | Creating function func\_count\_festivals to count the number of festivals in festivals table   * CREATE OR REPLACE FUNCTION func\_count\_festivals RETURN NUMBER IS   vn\_ct\_festivals NUMBER(3);    BEGIN  SELECT COUNT(\*)  INTO vn\_ct\_festivals  FROM festivals;    RETURN vn\_ct\_festivals;  END func\_count\_festivals;  /  SHOW ERRORS; | Function created.  No errors |  | Successful |
| 2. | Creating procedure proc\_count\_festivals to use function func\_count\_festivals |  |  |  |
|  | * CREATE OR REPLACE PROCEDURE proc\_count\_festivals IS   vn\_ct\_festivals NUMBER(3);    BEGIN  vn\_ct\_festivals:=func\_count\_festivals;  DBMS\_OUTPUT.PUT\_LINE('The number of festivals in festivals table is '||vn\_ct\_festivals);  END proc\_count\_festivals;  /  SHOW ERRORS; | Procedure created.  No errors |  | Successful |
| 2.i. | Testing procedure   * proc\_count\_festivals | The number of festivals in festival table is 9. |  | Successful |
| 3. | Creating FUNCTION func\_make\_username   * CREATE OR REPLACE FUNCTION func\_make\_username   (in\_staff\_id NUMBER) RETURN VARCHAR2 IS  vc\_username VARCHAR2(10);  BEGIN  SELECT CONCAT(SUBSTR(s.staff\_fname,1,3),SUBSTR(s.staff\_lname,1,4))  INTO vc\_username  FROM staff s  WHERE s.staff\_id=in\_staff\_id ;    RETURN vc\_username;  END func\_make\_username;  /  SHOW ERRORS; | Function created.  No errors |  | Successful |
| 4. | Creating procedure to output the username generated by function func\_make\_username   * CREATE OR REPLACE PROCEDURE proc\_output\_username(in\_staff\_id NUMBER) IS   vc\_username VARCHAR2(10);  BEGIN  vc\_username:=func\_make\_username(in\_staff\_id);  DBMS\_OUTPUT.PUT\_LINE('The username for staff ID '||in\_staff\_id||' is '||vc\_username);  END proc\_output\_username;  /  SHOW ERRORS; | Procedure created.  No errors |  | Successful |
| 4.i. | Testing procedure proc\_output\_username   * EXECUTE proc\_output\_username(1); | The username for staff ID is AAYMOKT |  | Successful |
| 5. | Creating function to return full staff name that takes staff id as parameter   * CREATE OR REPLACE FUNCTION func\_staff\_name(in\_staff\_id staff.staff\_id%TYPE) RETURN VARCHAR2 IS   vc\_full\_name VARCHAR2(30);    BEGIN  SELECT CONCAT(CONCAT(s.staff\_fname, ' '), s.staff\_lname)  INTO vc\_full\_name  FROM staff s  WHERE staff\_id=in\_staff\_id;    RETURN vc\_full\_name;  END func\_staff\_name;  /  SHOW ERRORS; | Function created.  No errors |  | Successful |
| 6. | Creating function to return staff salary that takes staff id as parameter   * CREATE OR REPLACE FUNCTION func\_staff\_salary(in\_staff\_id staff.staff\_id%TYPE) RETURN NUMBER IS   vn\_salary staff.staff\_salary%TYPE;    BEGIN  SELECT staff\_salary  INTO vn\_salary  FROM staff s  WHERE staff\_id=in\_staff\_id;    RETURN vn\_salary;  END func\_staff\_salary;  /  SHOW ERRORS; | Function created.  NO errors |  | Successful |
|  |  |  |  |  |
| 7. | Creating procedure to output name and salary of the staff identified by staff id from parameter   * CREATE OR REPLACE PROCEDURE proc\_output\_staff\_data(in\_staff\_id NUMBER) IS   vc\_fullname VARCHAR2(30);  vn\_salary staff.staff\_salary%TYPE;    BEGIN  vc\_fullname:= func\_staff\_name(in\_staff\_id);  vn\_salary := func\_staff\_salary(in\_staff\_id);  DBMS\_OUTPUT.PUT\_LINE('The staff '||vc\_fullname ||' has salary '||vn\_salary);  END proc\_output\_staff\_data;  /  SHOW ERRORS; | Procedure created.  No errors |  | Successful |
| 7.i. | Testing procedure  proc\_output\_staff\_data   * EXECUTE proc\_output\_staff\_data(2); | The staff HIRA KAUCHHA has salary 2200.2 |  | Successful |
| 8. | Creating function to return highest ticket price festival   * CREATE OR REPLACE FUNCTION func\_highest\_ticket\_price RETURN NUMBER IS   vn\_price festivals.ticket\_price%TYPE;    BEGIN  SELECT MAX(ticket\_price)  INTO vn\_price  FROM festivals f;    RETURN vn\_price;  END func\_highest\_ticket\_price;  /  SHOW ERRORS; | Functions created.  No errors |  | Successful |
| 9. | Creating procedure to output the highest ticket price festival   * CREATE OR REPLACE PROCEDURE proc\_output\_ticket\_price IS   vn\_price festivals.ticket\_price%TYPE;    BEGIN  vn\_price := func\_highest\_ticket\_price;  DBMS\_OUTPUT.PUT\_LINE('The highest ticket price among registered festival is '||vn\_price);  END proc\_output\_ticket\_price;  /  SHOW ERRORS; | Procedure created.  No errors |  | Successful |
| 9.i. | Testing procedure  proc\_output\_ticket\_price   * EXECUTE proc\_output\_ticket\_price |  |  | Successful |
| 10. | Creating trigger trig\_staff\_dob\_ck to check birth date of staff (not let below 18 years)   * CREATE OR REPLACE TRIGGER trig\_staff\_dob\_ck   BEFORE INSERT OR UPDATE  ON staff  FOR EACH ROW  WHEN (NEW.staff\_dob IS NOT NULL)    DECLARE  vd\_today DATE;    BEGIN  SELECT SYSDATE  INTO vd\_today  FROM DUAL;    IF :NEW.staff\_dob>(vd\_today-(365\*18)) THEN  RAISE\_APPLICATION\_ERROR(-20001, 'STAFF SHOULD BE AN ADULT');  END IF;  END trig\_staff\_dob\_ck;  /  SHOW ERRORS; | Trigger created.  No errors. |  | Successful |
|  | Testing trigger trig\_staff\_dob\_ck   * SELECT trigger\_name FROM user\_triggers WHERE trigger\_name= 'TRIG\_STAFF\_DOB\_CK' | 1 row returned. |  | Successful |
|  | Testing trigger trig\_staff\_dob\_ck  by inserting keeping age less than 18   * INSERT INTO staff   SELECT seq\_staff\_id.NEXTVAL, 'GEORGE', 'MICHAEL', 1, 4444.56, 'M', '2-JUN-2009' , NULL ,contact\_varray\_type(contact\_type('+9779866544377', 'GEORGE@GMAIL.COM'),  contact\_type('+9771234567998', 'GEORGE@YAHOO.COM')), REF(a)  FROM addresses a  WHERE a.street='562 HOCKEY STREET'; | Error Thrown: STAFF SHOULD BE AN ADULT |  | Successful |
|  | * SELECT staff\_id, staff\_fname, staff\_lname FROM staff WHERE staff\_fname LIKE '%G%'; | 0 row returned.  No row selected. |  | Successful |
|  | Testing the trigger trig\_staff\_dob\_ck by inserting keeping age more than 18   * INSERT INTO staff   SELECT seq\_staff\_id.NEXTVAL, 'GEORGE', 'MICHAEL', 1, 4444.56, 'M', '2-SEP-1998' ,NULL,contact\_varray\_type(contact\_type('+9779866544377', 'GEORGE@GMAIL.COM'),  contact\_type('+9771234567998', 'GEORGE@YAHOO.COM')), REF(a)  FROM addresses a  WHERE a.street='562 HOCKEY STREET'; | 1 row created. |  | Successful |
|  | Testing the trigger trig\_staff\_dob\_ck by inserting keeping age equal to 18   * INSERT INTO staff   SELECT seq\_staff\_id.NEXTVAL, 'SHAWN', 'MICHAEL', 1, 4444.56, 'M', '8-APR-2001' , NULL ,contact\_varray\_type(contact\_type('+9779866544377', 'GEORGE@GMAIL.COM'),  contact\_type('+9771234567998', 'GEORGE@YAHOO.COM')), REF(a)  FROM addresses a  WHERE a.street='562 HOCKEY STREET'; | 1 row created. |  | Successful |
|  | Checking if the row is inserted   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff WHERE staff\_fname LIKE 'SHAWN'; | 1 row returned. |  | Successful |
|  | Testing the trigger trig\_staff\_dob\_ck by inserting keeping staff\_dob NULL   * INSERT INTO staff   SELECT seq\_staff\_id.NEXTVAL, 'PAUL', 'MCCARTNEY', 1, 4444.56, 'M', NULL , NULL ,contact\_varray\_type(contact\_type('+9779866544377', 'PAUL@GMAIL.COM'),  contact\_type('+9771234567998', 'PAUL@YAHOO.COM')), REF(a)  FROM addresses a  WHERE a.street='562 HOCKEY STREET'; | 1 row created. |  | Successful |
|  | Checking if the row is inserted   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff WHERE staff\_fname = 'PAUL'; | 1 row returned. |  | Successful |
|  | Testing the trigger trig\_staff\_dob\_ck by updating keeping age NULL   * UPDATE staff SET staff\_dob = NULL   WHERE staff\_id = 9; | 1 row updated |  | Successful |  |
|  | Checking if the row is updated   * SELECT staff\_id, staff\_fname, staff\_lname, staff\_dob FROM staff WHERE staff\_fname = 'PAUL'; | 1 row returned. |  | Successful |
|  | Testing the trigger trig\_staff\_dob\_ck by updating keeping age less than 18   * UPDATE staff SET staff\_dob = '14-MAR-2010'   WHERE staff\_id = 5; | Error thrown. |  | Successful |
|  | Checking update of column staff\_dob for staff id 5   * SELECT staff\_id, staff\_fname, staff\_lname, staff\_dob FROM staff WHERE staff\_id=5; | Not updated. |  | Successful |
|  | Testing the trigger trig\_staff\_dob\_ck by updating keeping age equal to 18   * UPDATE staff SET staff\_dob = '8-April-2001'   WHERE staff\_id = 5; | 1 row updated. |  | Successful |
|  | Checking update of column staff\_dob for staff id 5   * SELECT staff\_id, staff\_fname, staff\_lname, staff\_dob FROM staff WHERE staff\_id=5; | Row updated. |  | Successful |
|  | --Creating a trigger to output message of insertion successful after insert on locations table   * CREATE OR REPLACE TRIGGER trig\_locations\_message   AFTER INSERT OR UPDATE  ON locations    BEGIN  IF INSERTING THEN  DBMS\_OUTPUT.PUT\_LINE('INSERTION SUCCESSFUL');  ELSIF UPDATING THEN  DBMS\_OUTPUT.PUT\_LINE('UPDATE SUCCESSFUL');  END IF;  END trig\_locations\_message;  /  SHOW ERRORS; | Triggers created.  No errors |  | Successful |
|  | Checking the creation of trigger trig\_locations\_message   * SELECT trigger\_name FROM user\_triggers WHERE trigger\_name= 'TRIG\_LOCATIONS\_MESSAGE' | 1 row returned. |  | Successful |
|  | Testing the trigger trig\_locations\_message by inserting a row in locations table   * INSERT INTO locations   VALUES (seq\_location\_id.NEXTVAL,address\_type('TAMGHAS 6', 'GULMI', 'NEPAL')); | 1 row created. |  | Successful |
|  | Checking if the row is inserted   * SELECT l.location\_id, l.location\_name FROM locations l WHERE l.location\_name.street='TAMGHAS 6' | 1 row returned |  | Successful |
|  | Testing the trigger trig\_locations\_message by updating an existing column in locations table   * UPDATE locations l   SET l.location\_name.street='TAMGHAS 9'  WHERE l.location\_id=10 | 1 row updated. |  | Successful |
|  | Checking if the row is updated   * SELECT l.location\_id, l.location\_name FROM locations l WHERE l.location\_name.street='TAMGHAS 9' | 1 row returned. |  | Successful |
|  | Creating a function func\_calculate\_staff\_age to calculate the age of staff with DOB as parameter   * CREATE OR REPLACE FUNCTION func\_calculate\_staff\_age(in\_staff\_dob DATE) RETURN NUMBER IS   vn\_staff\_age NUMBER(3);    BEGIN  vn\_staff\_age:=(SYSDATE - in\_staff\_dob)/365;  RETURN vn\_staff\_age;  END func\_calculate\_staff\_age;  /  SHOW ERRORS | Function created.  No errors |  | Successful |
|  | Creating procedure proc\_test\_calc\_age to test function func\_calculate\_staff\_age   * CREATE OR REPLACE PROCEDURE proc\_test\_calc\_age(in\_staff\_dob DATE, in\_staff\_firstname VARCHAR2) IS   vn\_staff\_age NUMBER(3);    BEGIN  vn\_staff\_age:=func\_calculate\_staff\_age(in\_staff\_dob);  DBMS\_OUTPUT.PUT\_LINE('You have added '||in\_staff\_firstname||' and their age is ' ||vn\_staff\_age);  END proc\_test\_calc\_age;  /  SHOW ERRORS | Procedure created.  No errors |  | Successful |
|  | Creating function func\_calc\_retiring to calculate days to retire   * CREATE OR REPLACE FUNCTION func\_calc\_retiring(in\_staff\_dob DATE) RETURN NUMBER IS   vn\_retiring\_days NUMBER(6);  BEGIN  vn\_retiring\_days:=(65-func\_calculate\_staff\_age(in\_staff\_dob))\*365;  RETURN vn\_retiring\_days;  END func\_calc\_retiring;  /  SHOW ERRORS | Function created.  No errors |  | Successful |
|  | Creating procedure proc\_calc\_retiring to test function func\_calc\_retiring   * CREATE OR REPLACE PROCEDURE proc\_calc\_retiring (in\_staff\_dob DATE) IS   vn\_retiring\_days NUMBER(6);  vn\_retiring\_years NUMBER(3);  BEGIN  vn\_retiring\_days:=func\_calc\_retiring(in\_staff\_dob);  vn\_retiring\_years:=vn\_retiring\_days/365;  DBMS\_OUTPUT.PUT\_LINE('Days before retirement: ' || vn\_retiring\_days);  DBMS\_OUTPUT.PUT\_LINE('Years before retirement: '||vn\_retiring\_years);  END proc\_calc\_retiring;  /  SHOW ERRORS | Procedure created.  No errors |  | Successful |
|  | Creating a combined trigger with switch cases for table staff   * CREATE OR REPLACE TRIGGER trig\_combined   AFTER INSERT OR UPDATE OR DELETE  ON staff  REFERENCING OLD AS existing  FOR EACH ROW  BEGIN  IF (INSERTING AND :NEW.staff\_dob IS NOT NULL) THEN  proc\_test\_calc\_age(:NEW.staff\_dob, :NEW.staff\_fname);  ELSIF (UPDATING AND :NEW.staff\_dob IS NOT NULL) THEN  proc\_calc\_retiring(:NEW.staff\_dob);  ELSIF DELETING THEN  DBMS\_OUTPUT.PUT\_LINE('You are deleting '||:existing.staff\_fname||' '||:existing.staff\_lname);  END IF;  END trig\_combined;  /  SHOW ERRORS | Trigger created.  No errors |  | Successful |
|  | Checking the creation of trigger trig\_combined   * SELECT trigger\_name FROM user\_triggers WHERE trigger\_name= 'TRIG\_COMBINED' | 1 row returned. |  | Successful |
|  | Checking the row in staff table where staff id = 9 before deleting   * SELECT s.staff\_id, s.staff\_fname, s.staff\_lname FROM staff s WHERE s.staff\_id = 9; | 1 row returned |  | Successful |
|  | Testing the trigger trig\_combined by deleting a row from staff table   * DELETE FROM staff   WHERE staff\_id=9 | Triggered fired.  1 row deleted. |  | Successful |
|  | Checking the row in staff table where staff id = 9 after deleting   * SELECT s.staff\_id, s.staff\_fname, s.staff\_lname FROM staff s WHERE s.staff\_id = 9 | No row selected. |  | Successful |
|  | checking the creation of function func\_calculate\_staff\_age   * SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_NAME ='FUNC\_CALCULATE\_STAFF\_AGE' | 1 row returned. |  | Successful |
|  | Checking the creation of procedure proc\_test\_calc\_age   * SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_NAME ='PROC\_TEST\_CALC\_AGE' | 1 row returned. |  | Successful |
|  | Testing function func\_calculate\_staff\_age using procedure proc\_test\_calc\_age   * EXECUTE proc\_test\_calc\_age('05-MAR-1995', 'BETTY') | You have added BETTY and their age is 24 |  | Successful |
|  | Checking the number of rows on staff before executing trigger   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff; | 7 rows selected. |  | Successful |
|  | Inserting new row into staff to test with non null date of birth   * INSERT INTO staff   SELECT seq\_staff\_id.NEXTVAL, 'JOHN', 'LENNON', 1, 4222.56, 'M', '12-JAN-1992' , NULL ,contact\_varray\_type(contact\_type('+9779854422342', 'JOHN@GMAIL.COM'),  contact\_type('+9779877655432', 'JOHN@YAHOO.COM')), REF(a)  FROM addresses a  WHERE a.street='562 HOCKEY STREET' | Triggered fired.  Output: You have added JOHN and their age is 27 |  | Successful |
|  | Checking the number of rows after executing trigger   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff WHERE staff\_fname = 'JOHN' | 1 row returned. |  | Successful |
|  | Inserting new row into staff to test with null date of birth   * INSERT INTO staff   SELECT seq\_staff\_id.NEXTVAL, 'JOHN', 'BONHAM', 1, 4222.56, 'M', NULL , NULL ,contact\_varray\_type(contact\_type('+9779854422343', 'JOHN@GMAIL.COM'),  contact\_type('+9779877655437', 'JOHN@YAHOO.COM')), REF(a)  FROM addresses a  WHERE a.street='562 HOCKEY STREET'; | 1 row created |  | Successful |
|  | Checking the number of rows after executing trigger   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff WHERE staff\_fname = 'JOHN' | 2 rows selected. |  | Successful |
|  | Checking the creation of function func\_calc\_retiring   * SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_NAME ='FUNC\_CALC\_RETIRING' | 1 row returned |  | Successful |
|  | Checking the creation of procedure proc\_calc\_retiring   * SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_NAME ='PROC\_CALC\_RETIRING' | 1 row returned. |  | Successful |
|  | Testing function func\_calc\_retiring using procedure proc\_calc\_retiring   * EXECUTE proc\_calc\_retiring('03-FEB-1977'); | OUTPUT: Days before retirement: 8395 Years before retirement: 23 |  | Successful |
|  | Checking the details of staff to be updated before updating the details   * SELECT staff\_id, staff\_fname, staff\_lname FROM staff WHERE staff\_id=5 | 1 row returned. |  | Successful |
|  | Updating the date of birth of staff with null DOB on staff id 5   * UPDATE staff   SET staff\_dob=NULL  WHERE staff\_id=5 | Trigger not fired. |  | Successful |
|  | Checking the details of staff 5 after updating the details   * SELECT staff\_id, staff\_fname, staff\_lname, staff\_dob FROM staff WHERE staff\_id=5; | 1 row returned |  | Successful |
|  | Updating the date of birth of staff with valid DOB on staff id 5   * UPDATE staff   SET staff\_dob='21-JAN-1989'  WHERE staff\_id=5 | 1 row updated. |  | Successful |
|  | Checking the details of staff 5 after updating the details   * SELECT staff\_id, staff\_fname, staff\_lname, staff\_dob FROM staff WHERE staff\_id=5 |  |  | Successful |
|  | Creating schema level trigger to notify the action being performed  --Citation: (Docs.oracle.com, 2019)   * CREATE OR REPLACE TRIGGER trig\_schema\_level   AFTER CREATE OR ALTER OR DROP  ON SCHEMA  DECLARE  vc\_event VARCHAR2(50);  vc\_current\_schema VARCHAR2(20);  vc\_event\_obj VARCHAR2(50);  vc\_obj\_type VARCHAR2(25);  BEGIN  SELECT ora\_sysevent  INTO vc\_event  FROM DUAL;    SELECT SYS\_CONTEXT('USERENV', 'CURRENT\_SCHEMA')  INTO vc\_current\_schema  FROM DUAL;    SELECT ora\_dict\_obj\_name  INTO vc\_event\_obj  FROM DUAL;    SELECT ora\_dict\_obj\_type  INTO vc\_obj\_type  FROM DUAL;    DBMS\_OUTPUT.PUT\_LINE('Event- '||vc\_event||' performed on schema '||vc\_current\_schema||' for '||vc\_event\_obj ||' '||vc\_obj\_type);  END trig\_schema\_level;  /  SHOW ERRORS; | Triggrs created.  No errors. |  | Successful |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |