TECHNICAL DOCUMENTATION REPORT

CSY2038 DATABASE

Submitted by:

GROUP 9

MEMBERS:

Swastika Adhikari 18406481

Amrit Thapa 18406540

Anuj Budha Magar 18406541

Pratima Dahal 18413693

Contents

[PROPOSED AUTOMATION STRATEGY 3](#_Toc8042069)

[INITIALLY GIVEN SCHEMA: 3](#_Toc8042070)

[SELECTED SCHEMA AND TABLE SPECIFICATION 3](#_Toc8042071)

[SCHEMA ENTITY RELATIONSHIP DIAGRAM 3](#_Toc8042072)

[Details of Schema 3](#_Toc8042073)

[Entity Description 3](#_Toc8042074)

[UDT’s Description 3](#_Toc8042075)

[Object Types 3](#_Toc8042076)

[Object Table 3](#_Toc8042077)

[Varray 3](#_Toc8042078)

[Nested Table 3](#_Toc8042079)

[Creation Order 3](#_Toc8042080)

[Order of Creation of Object Type 3](#_Toc8042081)

[Order of Creation of Varray Type 3](#_Toc8042082)

[Order of Creation of Nested Table Type 3](#_Toc8042083)

[Order of Creation of Tables 3](#_Toc8042084)

[Order of Creation of Sequences 3](#_Toc8042085)

[Order of Creation of Primary Keys 3](#_Toc8042086)

[Order of Creation of Foreign Keys 3](#_Toc8042087)

[Order of Unique Constraints 3](#_Toc8042088)

[Order of Check Constraints 3](#_Toc8042089)

[Order of Procedure 3](#_Toc8042090)

[Order of Functions 3](#_Toc8042091)

[Order of Trigger 3](#_Toc8042092)

[Order of Package 3](#_Toc8042093)

[Order of Drop 3](#_Toc8042094)

[Table Specification: 3](#_Toc8042095)

[Figure 3 : Table of the object table named addresses 3](#_Toc8042096)

[Figure 4 : Table of the object type named social\_media\_type 3](#_Toc8042097)

[Figure 6 : Table of the varray type named contact\_varray\_type 3](#_Toc8042098)

[*Figure 7 : Table of the nested\_table\_type named social\_media\_table\_type* 3](#_Toc8042099)

[Table specification of relational tables: 3](#_Toc8042100)

[Figure 8 : Table specification of relational tables 3](#_Toc8042101)

[TESTING: 3](#_Toc8042102)

[Test Plan: 3](#_Toc8042103)

[Test Log: 3](#_Toc8042104)

[CONCLUSION 112](#_Toc8042105)

[REFERENCES 112](#_Toc8042106)

[APPENDIX 112](#_Toc8042107)

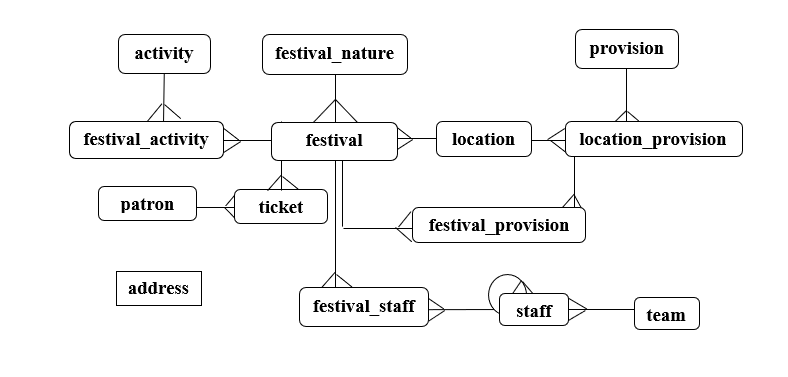
[REFERENCES 282](#_Toc8042108)

# PROPOSED AUTOMATION STRATEGY

Automation strategy is the discussion of different procedural elements used in the project. In this project, function and procedure are most used elements. Their significance in the project is as discussed below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Procedural Element Type** | **Element Name** | **Parameters** | **Purpose** | **Return Type** |
| Function | func\_festivals |  | Get the number of festivals in database | NUMBER |
| Procedure | proc\_festivals |  | Display the number of fesivals using function func\_festivals |  |
| Function | func\_get\_avg\_age\_of\_staff |  | Get the average age of staff | NUMBER |
| Procedure | proc\_avg\_age\_staff |  | Display the average age of staff using function func\_get\_avg\_age\_of\_staff |  |
| Function | func\_staff | in\_staff\_id (NUMBER) | Create the username of staff by using staff id | VARCHAR2 |
| Procedure | proc\_staff | in\_staff\_id (staff.staff\_id%TYPE) | Display the created username of staff |  |
| Function | func\_staff\_check\_gender | in\_staff\_id (NUMBER) | Test whether the staff is male or female | CHAR |
| Procedure | proc\_staff\_gender | in\_staff\_id (staff.staff\_id%TYPE) | Display the relative message by checking gender |  |
| Function | func\_get\_staff\_age | in\_staff\_dob (DATE) | Calculate the age of staff by providing the date of birth (implemented on trigger ) |  |
| Procedure | proc\_staff\_age | in\_staff\_id (NUMBER) | Display the age of staff of given staff id using function func\_get\_staff\_age |  |
| Function | func\_total\_salary |  | Return the total salary of staff per month | NUMBER |
| Procedure | proc\_salary\_func |  | Display the relative message relative to function func\_total\_salary |  |
| Procedure | <unnamed> |  | Display first name, surname and title of staff using different loop |  |
| Procedure | <unnamed> |  | Display the name of staff with particular id handling exceptions |  |
| Procedure | proc\_create\_team |  | Create a relational table team using procedure |  |
| Procedure | proc\_primary\_team |  | Set the primary key for relational table team |  |
| Procedure | proc\_foreign\_team |  | Set the foreign key for relational table team |  |
| Procedure | Proc\_staff |  | Inserting into relational table using procedure |  |
| Procedure | proc\_update\_staff |  | Updating relational table using procedure |  |
| Procedure | proc\_delete\_staff |  | Deleting data of relational table using procedure |  |
| Procedure | proc\_get\_user\_salary |  | Display the salary of a staff using functions like CEIL, FLOOR, TRUNC, CONCAT,ROUND, and SUBSTR for name |  |
| Procedure | <unnamed> |  | Using loop to display the information in different styles |  |
| Procedure | proc\_display\_staff | in\_id (staff.staff\_id%TYPE) | Display the information of staff by entering staff id |  |
| Procedure | proc\_insert\_staff | in\_staff\_firstname (VARCHAR2), in\_staff\_surname (VARCHAR2), in\_staff\_hire\_date (DATE), in\_staff\_dob (DATE) | Insert new data into staff table using procedure |  |
| Package | pkg\_search\_staff | In\_email (VARCHAR2)  (--NOTE : The package uses parameter for procedure and function inside the package) | Search the staff info using their email address |  |
| Trigger | trig\_staff\_age |  | Validate the age of staff on insert and update |  |
| Trigger | trig\_login\_hr |  | Record the session when user is logged in |  |
| Trigger | trig\_operation\_staff |  | Display a message on data manipulation of staff |  |
| Trigger | trig\_audit\_tr |  | Display a message on changes in data definition |  |
| Trigger | trig\_message |  | Display a message on startup / log in of the database user |  |

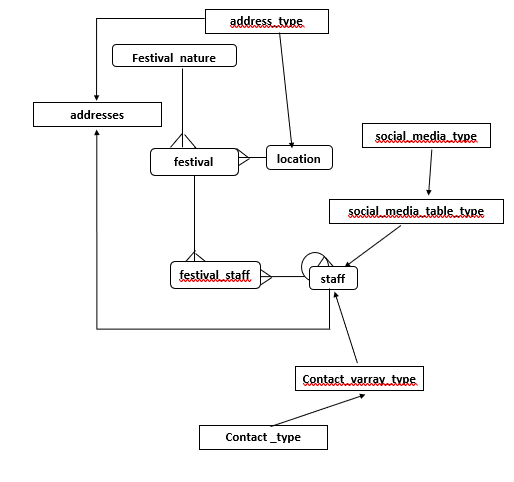
## INITIALLY GIVEN SCHEMA:



The given above schema was initially provided by the module tutor to the group and from there the group had to select the five tables by analyzing it and should propose a table specification including the normal datatypes along with the complex and user defined types. So, after looking at the given schema briefly along with the group members discussion, the group came up with five tables and their table specification.

# SELECTED SCHEMA AND TABLE SPECIFICATION

## SCHEMA ENTITY RELATIONSHIP DIAGRAM



After the analysis of the given schema our group decided to use the above shown tables and the detailed description of schema is given below:

## Details of Schema

Number of Tables: Five

Number of Primary keys: Five

Number of Foreign Keys:

Number of Compound Primary Keys:

Number of Compound Foreign Keys:

## Entity Description

Festival: This table stores the details about the festival organized by the staff in the place.

Festival Nature: This table stores the details about the type of the festival.

Staff: This table stores the details about the staff present in the organization.

Location: This table stores the details about the addresses where festival is organized.

Festival Staff: This table stores the details about the festival organized by the staff.

## UDT’s Description

This schema also contains UDT’s like object types, object table, varrays and nested tables.

## Object Types

Address Type: Country, City and Street is stored in this type.

Social Media Type: Media name, username and contact is stored in this type.

Contact Type: Email, phone and mobile is stored in this type.

## Object Table

Addresses: It is created of address type.

## Varray

Contact Varray Type: Variable array of contact type

## Nested Table

Social Media Table Type: Table of social media type

## Creation Order

The below information shows about the order of creation of tables, object type, varray, primary key, foreign keys, sequences, check constraints, upper constraints, procedure, functions, triggers.

## Order of Creation of Object Type

* Address\_type
* Social\_media\_type
* Contact\_type

## Order of Creation of Varray Type

* Contact\_varray\_type

## Order of Creation of Nested Table Type

* Social\_media\_table\_type

## Order of Creation of Tables

* Locations
* Festival\_nature
* Festivals
* Staff
* Festival\_staff

## Order of Creation of Sequences

* Seq\_locations
* Seq\_festival\_nature
* Seq\_staff

## Order of Creation of Primary Keys

* pk\_locations
* pk\_festival\_nature
* pk\_festivals
* pk\_staff
* pk\_festival\_staff

## Order of Creation of Foreign Keys

|  |  |
| --- | --- |
| Constraints Name | Table Name |
| fk\_f\_locations |  |
| fk\_f\_festival\_nature |  |
| fk\_f\_staff |  |
| fk\_f\_festivals |  |
| fk\_s\_staff |  |

## Order of Unique Constraints

* uc\_festival\_nature

## Order of Check Constraints

* ck\_festival\_name
* ck\_festival\_type
* ck\_gender
* ck\_staff\_firstname
* ck\_staff\_surname
* ck\_staff\_title

## Order of Procedure

* proc\_create\_team
* proc\_primary\_team
* proc\_foreign\_team
* proc\_staff
* proc\_update\_staff
* proc\_delete\_staff
* proc\_get\_user\_salary
* proc\_display\_staff
* proc\_insert\_staff
* proc\_festivals
* proc\_avg\_age\_staff
* proc\_staff
* proc\_satff\_gender
* proc\_salary\_func
* proc\_staff\_age
* proc\_fes\_imp\_cursor
* proc\_imp\_cursor
* proc\_exp\_cursor
* proc\_exp\_staff\_salary
* proc\_exp\_cursor

## Order of Functions

* func\_festivals
* func\_get\_avg\_age\_of\_staff
* func\_staff
* func\_check\_staff\_gender
* func\_total\_salary
* func\_get\_staff\_age

## Order of Trigger

* trig\_staff\_age
* trig\_login\_hr
* trig\_operation\_staff
* trig\_audit\_tr
* trig\_message

## Order of Package

* pkg\_search\_staff

## Order of Drop

Dropping Trigger

* DROP TRIGGER trig\_staff\_age;
* DROP TRIGGER trig\_login\_hr;
* DROP TRIGGER trig\_operation\_staff;
* DROP TRIGGER trig\_audit\_tr;
* DROP TRIGGER trig\_message;

Dropping Cursor

* DROP PROCEDURE proc\_fes\_imp\_cursor;
* DROP PROCEDURE proc\_imp\_cursor;
* DROP PROCEDURE proc\_exp\_cursor;
* DROP PROCEDURE proc\_exp\_staff\_salary;

Dropping Function

* DROP FUNCTION func\_festivals;
* DROP FUNCTION func\_get\_staff\_age;
* DROP FUNCTION func\_get\_avg\_age\_of\_staff;
* DROP FUNCTION func\_staff;
* DROP FUNCTION func\_staff\_check\_gender;
* DROP FUNCTION func\_total\_salary;

Dropping Package

* DROP PACKAGE pkg\_search\_staff;

Dropping Procedure

* DROP PROCEDURE proc\_create\_team;
* DROP PROCEDURE proc\_primary\_team;
* DROP PROCEDURE proc\_foreign\_team;
* DROP PROCEDURE proc\_update\_staff;
* DROP PROCEDURE proc\_delete\_staff;
* DROP PROCEDURE proc\_get\_user\_salary;
* DROP PROCEDURE proc\_display\_staff;
* DROP PROCEDURE proc\_insert\_staff;
* DROP PROCEDURE proc\_festivals;
* DROP PROCEDURE proc\_staff\_age;
* DROP PROCEDURE proc\_avg\_age\_staff;
* DROP PROCEDURE proc\_staff;
* DROP PROCEDURE proc\_staff\_gender;
* DROP PROCEDURE proc\_salary\_func;

Dropping constraints

Dropping Foreign Keys

* ALTER TABLE festivals

DROP CONSTRAINT fk\_f\_locations;

* ALTER TABLE staff

DROP CONSTRAINT fk\_s\_staff;

* ALTER TABLE team

DROP CONSTRAINT fk\_t\_staff;

* ALTER TABLE festivals

DROP CONSTRAINT fk\_f\_festival\_nature;

* ALTER TABLE festival\_staff

DROP CONSTRAINT fk\_f\_staff;

* ALTER TABLE festival\_staff

DROP CONSTRAINT fk\_f\_festivals;

Dropping Unique Constraints

* ALTER TABLE festival\_nature

DROP CONSTRAINT uc\_festival\_nature;

Dropping Check Constraints

* ALTER TABLE staff

DROP CONSTRAINT ck\_gender;

* ALTER TABLE staff

DROP CONSTRAINT ck\_staff\_title;

* ALTER TABLE staff

DROP CONSTRAINT ck\_staff\_firstname;

* ALTER TABLE staff

DROP CONSTRAINT ck\_staff\_surname;

* ALTER TABLE festival\_nature

DROP CONSTRAINT ck\_festival\_type;

Dropping Primary keys

* ALTER TABLE locations

DROP CONSTRAINT pk\_locations;

* ALTER TABLE team

DROP CONSTRAINT pk\_team;

* ALTER TABLE festival\_nature

DROP CONSTRAINT pk\_festival\_nature;

* ALTER TABLE festivals

DROP CONSTRAINT pk\_festivals;

* ALTER TABLE festival\_staff

DROP CONSTRAINT pk\_festival\_staff;

* ALTER TABLE festivals

DROP CONSTRAINT ck\_festival\_name;

* ALTER TABLE staff

DROP CONSTRAINT pk\_staff;

Dropping Table

* DROP TABLE LOCATIONS;
* DROP TABLE FESTIVAL\_NATURE;
* DROP TABLE FESTIVALS;
* DROP TABLE STAFF;
* DROP TABLE FESTIVAL\_STAFF;
* DROP TABLE team;

Dropping Type

* DROP TYPE social\_media\_table\_type;
* DROP TYPE contact\_varray\_type;
* DROP TYPE contact\_type;
* DROP TABLE addresses;
* DROP TYPE social\_media\_type;
* DROP TYPE address\_type;
* DROP TYPE team\_varray\_type;
* DROP TABLE hr\_event;

Dropping Sequence

* DROP SEQUENCE seq\_locations;
* DROP SEQUENCE seq\_festival\_nature;
* DROP SEQUENCE seq\_staff;

# Table Specification:

The following table shows the objects used to complete the project in an efficient way:

|  |  |  |
| --- | --- | --- |
| Object Table | Attributes | Datatypes |
| addresses | **Address\_type:** |  |
|  | street | VARCHAR2(25) |
|  | city | VARCHAR2(25) |
|  | country | VARCHAR2(20) |

## Figure 3 : Table of the object table named addresses

|  |  |  |
| --- | --- | --- |
| Type | Attributes | Datatypes |
| Social\_media\_type | Social\_media\_name | VARCHAR2(50) |
|  | Social\_media\_username | VARCHAR2(50) |
|  | Contact | VARCHAR2(15) |

## Figure 4 : Table of the object type named social\_media\_type

|  |  |  |
| --- | --- | --- |
| Type | Attributes | Datatypes |
| contact\_type | email | VARCHAR2(50) |
|  | phone | VARCHAR2(15) |
|  | mobile | VARCHAR2(15) |

Figure 5 : Table of the object type named contact\_type

|  |  |  |
| --- | --- | --- |
| Type | Attributes | Datatypes |
| Contact\_varray\_type | Contact\_type: |  |
|  | email | VARCHAR2(50) |
|  | phone | VARCHAR2(15) |
|  | Mobile | VARCHAR2(15) |

## Figure 6 : Table of the varray type named contact\_varray\_type

|  |  |  |
| --- | --- | --- |
| Type | Attributes | Datatypes |
| Social\_media\_table\_type | Social\_media\_type: |  |
|  | Social\_media\_name | VARCHAR2(50) |
|  | Social\_media\_username | NUMBER(50) |
|  | contact | NUMBER(15) |

### *Figure 7 : Table of the nested\_table\_type named social\_media\_table\_type*

# Table specification of relational tables:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tables | Attributes | Datatype | Constraints | Default | Sequence |
| locations | \*location\_id | NUMBER(11) | pk\_location |  | seq\_location\_id |
|  | Location\_address | Address\_varray\_type |  |  |  |
|  |  |  |  |  |  |
| festival\_nature | \*festival\_nature\_id | NUMBER(11) | Pk\_festival\_nature |  | seq\_festival\_nature\_id |
|  | festival\_type | NUMBER(50) | UPPER  UNIQUE |  |  |
|  |  |  |  |  |  |
| festivals | \*festival\_nature\_id | NUMBER(11) | pk\_festivals  fk\_f\_festival\_nature  NOT NULL |  | seq\_locations |
|  | \*location\_id | NUMBER(11) | Pk\_festivals  Fk\_f\_locations  NOT NULL |  |  |
|  | festival\_name | VARCHAR2(50) | UPPER |  |  |
|  | occurring\_date | DATE |  | SYSDATE |  |
|  |  |  |  |  |  |
| staff | \*staff\_id | NUMBER(11) | pk\_staff |  | Seq\_staff\_id |
|  | staff\_title | VARCHAR2(5) | CHECK IN(MR , MRS , DR , ER ,MISS ) | ‘MRS’ |  |
|  | staff\_firstname | VARCHAR2(30) | NOT NULL, UPPER |  |  |
|  | staff\_surname | VARCHAR2(30) | UPPER,NOT NULL | SYSDATE |  |
|  | staff\_gender | CHAR(1) | CHECK IN(M , F) | ‘F’ |  |
|  | staff\_contact | Contact\_varray\_type | UNIQUE |  |  |
|  | staff\_address | REF OF address\_type |  |  |  |
|  | social\_media\_used | Social\_media\_table\_type |  |  |  |
|  | staff\_blood\_group | CHAR(3) |  |  |  |
|  | staff\_hire\_date | DATE |  | SYSDATE |  |
|  | staff\_salary | NUMBER(8,2) |  |  |  |
|  | staff\_dob | DATE |  |  |  |
|  | \*leader | NUMBER(11) | Fk\_s\_staff  NOT NULL |  |  |
|  |  |  |  |  |  |
| festival\_staff | \*staff\_id | NUMBER(11) | Fk\_f\_staff  pk\_festival\_staff  NOT NULL |  |  |
|  | \*location\_id | NUMBER(11) | Fk\_f\_festivals  pk\_festival\_staff  NOT NULL |  |  |
|  | \*festival\_nature\_id | NUMBER(11) | Fk\_f\_festivals  pk\_festival\_staff  NOT NULL |  |  |

## Figure 8 : Table specification of relational tables

# TESTING:

## Test Plan:

Test was planned during the discussion of schema. Different types of tests were to be conducted by team members during the development. Create, insert, update and all other objects were tested first while creating the database. Following is the test after creating the whole system to check any remaining errors.

## Test Log:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No | Test Description | Expected Result | Actual Outcome | Action |
|  |  |  | **CREATE** |  |
|  | Code to create address\_type:  CREATE OR REPLACE TYPE address\_type AS OBJECT(  country VARCHAR2(25),  city VARCHAR2(50),  street VARCHAR2(255));  / | Type successfully Created |  | No Action needed |
|  | Code to create social media type:  CREATE OR REPLACE TYPE social\_media\_type AS OBJECT(  social\_media\_name VARCHAR2(50),  social\_media\_username VARCHAR2(50),  contact VARCHAR2(15));  / | Type successfully created |  | No Action needed |
|  | Code to create contact type:  CREATE OR REPLACE TYPE contact\_type AS OBJECT(  email VARCHAR2(50),  phone VARCHAR2(15),  mobile VARCHAR2(15));  / | Type successfully created |  | No Action needed |
|  | Code to create object table addresses:  CREATE TABLE addresses OF address\_type(  city DEFAULT 'KATHMANDU',  country DEFAULT 'NEPAL'); | Object table successfully created |  | No Action needed |
|  | Code to create varray object using contact type:  CREATE TYPE contact\_varray\_type AS VARRAY(50) OF contact\_type;  / | Varray type successfully created |  | No Action needed |
|  | Code to create a table type:  CREATE TYPE social\_media\_table\_type AS TABLE OF social\_media\_type;  / | Table type successfully created |  | No Action needed |
|  | Creating relational table locations:  CREATE TABLE locations(  location\_id NUMBER(11),  location\_address address\_type); | Table successfully created |  | No Action needed |
|  | Creating relational table festival\_nature:  CREATE TABLE festival\_nature(  festival\_nature\_id NUMBER(11),  festival\_type VARCHAR2(50)); | Table successfully created |  | No Action needed |
|  | Creating relational table festivals:  CREATE TABLE festivals(  festival\_nature\_id NUMBER(11),  location\_id NUMBER(11),  festival\_name VARCHAR2(50),  occuring\_date DATE); | Table successfully created |  | No Action needed |
|  | Creating relational table staff:  CREATE TABLE staff(  staff\_id NUMBER(11),  staff\_title VARCHAR2(5) DEFAULT 'MRS',  staff\_firstname VARCHAR2(30),  staff\_surname VARCHAR2(30),  staff\_contact contact\_varray\_type,  staff\_address REF address\_type SCOPE IS addresses,  social\_media\_used social\_media\_table\_type)  NESTED TABLE social\_media\_used STORE AS social\_media\_table; | Table successfully created |  | No Action needed |
|  | Creating relational table festival\_staff:  CREATE TABLE festival\_staff(  staff\_id NUMBER(11) NOT NULL,  location\_id NUMBER(11) NOT NULL,  festival\_nature\_id NUMBER(11) NOT NULL); | Table successfully created |  | No Action needed |
|  | Creating Sequence seq\_locations:  CREATE SEQUENCE seq\_locations  INCREMENT BY 1  START WITH 5000  NOCYCLE; | Sequence successfully created |  | No Action needed |
|  | Creating Sequence seq\_festival\_nature:  CREATE SEQUENCE seq\_festival\_nature  INCREMENT BY 1  START WITH 50001  NOCYCLE; | Sequence successfully created |  | No Action needed |
|  | Creating Sequence seq\_staff:  CREATE SEQUENCE seq\_staff  INCREMENT BY 2  START WITH 1000  MAXVALUE 10000000000; | Sequence successfully created |  | No Action needed |
|  |  |  |  |  |
|  |  |  | **ALTER** |  |
|  | ALTER TABLE staff  ADD (staff\_gender CHAR); | Table Altered. |  | No Action needed |
|  | ALTER TABLE STAFF  ADD (leader NUMBER(11)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD (staff\_dob DATE ); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD (staff\_salary NUMBER(8,2)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD (staff\_hire\_date DATE DEFAULT SYSDATE); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD (staff\_blood\_group CHAR(3)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  MODIFY staff\_gender DEFAULT 'M'; | Table Altered. |  | No Action needed |
|  | ALTER TABLE festivals  MODIFY occuring\_date DEFAULT SYSDATE; | Table Altered. |  | No Action needed |
|  | ALTER TABLE locations  ADD CONSTRAINT pk\_locations  PRIMARY KEY(location\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festival\_nature  No Action needed ADD CONSTRAINT pk\_festival\_nature  PRIMARY KEY(festival\_nature\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festivals  ADD CONSTRAINT pk\_festivals  PRIMARY KEY(location\_id,festival\_nature\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD CONSTRAINT pk\_staff  PRIMARY KEY(staff\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festival\_staff  ADD CONSTRAINT pk\_festival\_staff  PRIMARY KEY(staff\_id,location\_id, festival\_nature\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festivals  ADD CONSTRAINT fk\_f\_locations  FOREIGN KEY(location\_id)  REFERENCES locations(location\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festivals  ADD CONSTRAINT fk\_f\_festival\_nature  FOREIGN KEY(festival\_nature\_id)  REFERENCES festival\_nature(festival\_nature\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festival\_staff  ADD CONSTRAINT fk\_f\_staff  FOREIGN KEY(staff\_id)  REFERENCES staff(staff\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festival\_staff  ADD CONSTRAINT fk\_f\_festivals  FOREIGN KEY(location\_id,festival\_nature\_id)  REFERENCES festivals(location\_id,festival\_nature\_id); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festival\_nature  ADD CONSTRAINT uc\_festival\_nature  UNIQUE(festival\_type); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD CONSTRAINT ck\_gender  CHECK(staff\_gender IN ('M','F')); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD CONSTRAINT ck\_staff\_title  CHECK(staff\_title IN ('MR','MRS','DR','ER','MISS')); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD CONSTRAINT ck\_staff\_firstname  CHECK (staff\_firstname = UPPER(staff\_firstname)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD CONSTRAINT ck\_staff\_surname  CHECK (staff\_surname = UPPER(staff\_surname)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festival\_nature  ADD CONSTRAINT ck\_festival\_type  CHECK(festival\_type =UPPER(festival\_type)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE festivals  ADD CONSTRAINT ck\_festival\_name  CHECK(festival\_name =UPPER(festival\_name)); | Table Altered. |  | No Action needed |
|  | ALTER TABLE staff  ADD CONSTRAINT fk\_s\_staff  FOREIGN KEY (leader)  REFERENCES staff(staff\_id); | Table Altered. |  | No Action needed |
|  |  |  | **INSERT** |  |
|  | **INSERTING VALUES IN LOCATIONS**  **INSERT INTO** locations(location\_id,location\_address)  VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','KATHMANDU','PUSPANJALI MARG'));  **VIEWING THE INSERTED DATA FROM LOCATIONS TABLE**  SELECT location\_id,location\_address FROM locations | 1 Row Created |  | No Action needed |
|  | **INSERTING VALUES IN FESTIVAL\_NATURE**  INSERT INTO festival\_nature(festival\_nature\_id,festival\_type)  VALUES(seq\_festival\_nature.NEXTVAL,'EXHIBITION');  **VIEWING THE INSERTED DATA FROM FESTIVAL\_NATURE TABLE**  SELECT festival\_nature\_id,festival\_type FROM festival\_nature; | 1 Row Created |  | No Action needed |
|  | **INSERTING VALUES IN ADDRESSES**  INSERT INTO addresses  VALUES ('NEPAL', 'KATHMANDU','PUSPANJALI MARG');  **VIEWING THE INSERTED DATA FROM ADDRESSES**  SELECT country,city,street FROM addresses; | 1 Row Created |  | No Action needed |
|  | **INSERTING VALUES IN STAFF**  INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)  SELECT seq\_staff.NEXTVAL,'MRS','PRATIMA','DAHAL','F',contact\_varray\_type(  contact\_type('PRATIMA@GMAIL.COM','01-4000090','0989077888'),  contact\_type('DPRATIMA@GMAIL.COM','01-4890900','9808900990')),  social\_media\_table\_type(  social\_media\_type('FACEBOOK','DPRATIMA',''),  social\_media\_type('TWITTER','PRATIMAD','')),  REF(a) FROM addresses a  WHERE a.street= 'KAPAN'; | 1 Row Created |  | No Action needed |
|  | **INSERTING VALUES IN FESTIVALS**  INSERT INTO festivals  VALUES (50001,5000, 'ROBOT SHOWING COMPETITION', '29-OCT-2019');  **VIEWING THE INSERTED DATA IN FESTIVALS TABLE**  SELECT festival\_nature\_id,location\_id,festival\_name,occuring\_date FROM festivals; | 1 Row Created |  | No Action needed |
|  |  |  | **QUERY** |  |
|  | Query to show detail of staff who hired before leader  SELECT s.staff\_id, s.staff\_firstname, staff\_surname FROM staff s  WHERE exists(  SELECT staff\_id FROM staff l  WHERE leader IS NOT NULL  AND s.leader = l.leader  AND s.staff\_hire\_date>l.staff\_hire\_date); | Show all the records of the staff who hired before the leader |  | No action required |
|  | SELECT staff\_firstname,staff\_surname,to\_char(staff\_salary,'99999999') Salary  FROM staff  WHERE staff\_surname != 'SHARMA'; | Show firstname surname and salary of staff whose surname is not SHARMA |  | No Action required |
|  | SELECT fn.festival\_nature\_id  FROM festival\_nature fn  WHERE fn.festival\_type IS NULL; | Sho  festival nature id whose name is null |  | No action needed |
|  | SELECT location\_id,location\_address  FROM locations; | Show location id and address |  | No action needed |
|  | SELECT l.location\_address.city,l.location\_id  FROM locations l; | Show all location id and city |  | No action needed |
|  | SELECT staff\_id,staff\_firstname,staff\_address  FROM staff; | Show staff firstname id and address in oid form |  | No action needed |
|  | SELECT staff\_id,staff\_firstname,deref(staff\_address) staff\_address  FROM staff; | Show all staff id firstname and address in type form |  | No action needed |
|  | SELECT staff\_firstname,staff\_surname,deref(staff\_address).country  FROM staff  MINUS  SELECT staff\_firstname,staff\_surname,deref(staff\_address).country  FROM staff  WHERE deref(staff\_address).country != 'NEPAL'; | Detail of Staff firstname surname and country where country is NEPAL |  | No action needed |
|  | SELECT DISTINCT(s.staff\_id),s.staff\_firstname,s.staff\_address.country country,  s.staff\_address.city city,s.staff\_address.street street,soc.social\_media\_username  FROM staff s,TABLE(s.social\_media\_used) soc; | Show detail of staff id, firstnam, country, city, street and staff username in social media |  | No action needed |
|  | SELECT f.festival\_name, f.occuring\_date, fn.festival\_type  FROM festivals f  INNER JOIN festival\_nature fn ON  f.festival\_nature\_id = fn.festival\_nature\_id  ORDER BY f.festival\_name; | Show all festival name occurring date and festival type in order |  | No action needed |
|  | SELECT f.festival\_name festival\_name, f.occuring\_date, fn.festival\_type festival\_type, l.location\_address.country country,l.location\_address.city city, l.location\_address.street  street  FROM festivals f  INNER JOIN festival\_nature fn ON  f.festival\_nature\_id = fn.festival\_nature\_id  JOIN locations l ON  f.location\_id = l.location\_id  ORDER BY f.festival\_name; | Detail of festival name, occurring date and occurring address |  | No action needed |
|  | SELECT f.festival\_name  FROM festivals f  WHERE f.location\_id IN (  SELECT l.location\_id  FROM locations l  WHERE l.location\_address.city = 'KATHMANDU'); | Show all festival name occurring at Kathmandu |  | No action needed |
|  | SELECT f.festival\_name,s.staff\_firstname,s.staff\_surname  FROM festivals f  FULL OUTER JOIN festival\_staff ff  ON f.festival\_nature\_id = ff.festival\_nature\_id  FULL OUTER JOIN staff s  ON ff.staff\_id = s.staff\_id; | Show all festival name, staff name by which staff organize that festivals |  | No action needed |
|  | SELECT ff.festival\_type  FROM festival\_nature ff  WHERE ff.festival\_nature\_id IN (  SELECT f.festival\_nature\_id  FROM festivals f  WHERE f.location\_id IN(  SELECT l.location\_id  FROM locations l  WHERE l.location\_address.country ='NEPAL')); | Show all festival type held in country nepal |  | No action needed |
|  | SELECT \* FROM addresses a  WHERE REF(a) NOT IN(  SELECT s.staff\_address  FROM staff s  ); | Show address where staff is not available |  | No action needed |
|  | SELECT ff.festival\_type,f.festival\_name  FROM festival\_nature ff  FULL OUTER JOIN festivals f  ON ff.festival\_nature\_id = f.festival\_nature\_id  WHERE ff.festival\_nature\_id IN (  SELECT f.festival\_nature\_id  FROM festivals f  WHERE f.location\_id IN(  SELECT l.location\_id  FROM locations l  WHERE l.location\_address.country IN(  SELECT s.staff\_address.country Country  FROM staff s  WHERE s.staff\_address.country = 'NEPAL'  AND s.staff\_address.city ='KATHMANDU'  UNION  SELECT l.location\_address.country Country  FROM locations l  WHERE l.location\_address.country = 'NEPAL'))); | Show all festival name and type in location Kathmandu Nepal |  | No action needed |
|  | SELECT s.staff\_id,s.staff\_firstname,l.location\_address.country Country  FROM staff s  LEFT OUTER JOIN(  festival\_staff ff INNER JOIN festivals f  ON ff.festival\_nature\_id = f.festival\_nature\_id  AND ff.location\_id = f.location\_id  JOIN locations l  ON l.location\_id = f.location\_id  )  ON s.staff\_id = ff.staff\_id; | Show all detail of staff firstname, id and Country where they live |  | No action needed |
|  | SELECT f.festival\_name, f.occuring\_date  FROM festivals f  WHERE EXISTS(  SELECT location\_id  FROM locations l  WHERE l.location\_address.country = 'NEPAL'  ); | Show festival name where exists in country nepal |  | No action needed |
|  | SELECT s.staff\_firstname,s.staff\_surname  FROM staff s  WHERE s.staff\_address.city IN ('KATHMANDU' , 'POKHARA'); | Staff firstname and surname who lives in Kathmandu and pokhara city |  | No action needed |
|  | SELECT staff\_id,staff\_firstname,staff\_surname,staff\_hire\_date  FROM staff  WHERE staff\_hire\_date > '1-JAN-2012'  AND staff\_hire\_date < SYSDATE ; | Show staff detail whi hired after 1 jan 2012 till today |  | No action needed |
|  | SELECT f.festival\_name,f.occuring\_date,s.staff\_firstname,s.staff\_surname  FROM festivals f INNER JOIN festival\_staff ff  ON f.festival\_nature\_id = ff.festival\_nature\_id  INNER JOIN staff s  ON s.staff\_id = ff.staff\_id  WHERE (SYSDATE-s.staff\_dob) > 20; | Show festival name and staff detail more than 20 years who organize the festival |  | No action needed |
|  | SELECT f.festival\_name, f.occuring\_date, fn.festival\_type  FROM festivals f  INNER JOIN festival\_nature fn ON  f.festival\_nature\_id = fn.festival\_nature\_id  ORDER BY f.festival\_nature\_id DESC; | Show the details of festival and nature type |  | No action needed |
|  | SELECT f.festival\_name festival\_name, f.occuring\_date occurring\_date, fn.festival\_type, l.location\_address.country country,l.location\_address.city city, l.location\_address.street  street  FROM festivals f  INNER JOIN festival\_nature fn ON  f.festival\_nature\_id = fn.festival\_nature\_id  JOIN locations l ON  f.location\_id = l.location\_id  ORDER BY f.occuring\_date ASC; | Show festival name, occurring date, type and location in ascending order |  | No action needed |
|  | SELECT l.location\_address.city CITY  FROM locations l  INNER JOIN festivals f  ON l.location\_id = f.location\_id  FULL OUTER JOIN festival\_staff ff  ON f.festival\_nature\_id = ff.festival\_nature\_id  INTERSECT  SELECT s.staff\_address.city  FROM staff s  LEFT OUTER JOIN festival\_staff fe  ON s.staff\_id = fe.staff\_id; | City where festivals and staff both are present |  | No action needed |
|  | SELECT COUNT(festival\_nature\_id) Serial,festival\_name  FROM festivals  GROUP BY festival\_name; | Count the festival how many times it celebrated |  | No action needed |
|  | SELECT staff\_salary , CEIL(staff\_salary) "CEIL" ,  FLOOR(staff\_salary)"FLOOR",  ROUND(staff\_salary)"ROUND",  TRUNC(staff\_salary)"TRUNCATE"  FROM staff; | Change the salary in ceil floor truncate and round |  | No action needed |
|  | SELECT MAX(to\_char(staff\_salary,'999999D99')) "Max Salary",CEIL(MAX(to\_char(staff\_salary,'999999D99'))) "CEIL" ,  FLOOR(MAX(to\_char(staff\_salary,'999999D99')))"FLOOR",  ROUND(MAX(to\_char(staff\_salary,'999999D99')))"ROUND",  TRUNC(MAX(to\_char(staff\_salary,'999999D99')))"TRUNCATE"  FROM staff; | Truncate, ceil, floor, and round the maximum salary among the staff |  | No action needed |
|  | SELECT MIN(to\_char(staff\_salary,'999999D99')),CEIL(MIN(to\_char(staff\_salary,'999999D99'))) "CEIL" ,  FLOOR(MIN(to\_char(staff\_salary,'999999D99')))"FLOOR",  ROUND(MIN(to\_char(staff\_salary,'999999D99')))"ROUND",  TRUNC(MIN(to\_char(staff\_salary,'999999D99')))"TRUNCATE"  FROM staff; | Truncate, ceil, floor, and round the minimum salary among the staff |  | No action needed |
|  | SELECT \* FROM(  SELECT staff\_firstname , ROUND(staff\_salary) "max and round" , CEIL(staff\_salary) "CEIL" ,FLOOR(staff\_salary)"FLOOR",TRUNC(staff\_salary)"Trunct"  FROM staff  ORDER BY staff\_salary ASC)  WHERE rownum<=1; | Show first staff detail and salary in round ceil floor and truncate |  | No action needed |
|  |  |  | **PROCEDURE** |  |
|  | Creation of Unnamed procedure:  DECLARE  vn\_length NUMBER(2);  vn\_counter NUMBER(3) := 1;  vc\_title VARCHAR2(20);  vc\_firstname VARCHAR2(20);  vc\_surname VARCHAR2(20);  BEGIN  SELECT staff\_title,staff\_firstname,staff\_surname INTO vc\_title,vc\_firstname,vc\_surname FROM staff WHERE staff\_id = 1000;  DBMS\_OUTPUT.PUT\_LINE('--Title--');  vn\_length := LENGTH(vc\_title);  --START THE LOOP  LOOP  --LOOP TERMINATION CONDITION  EXIT WHEN vn\_counter>vn\_length;  DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_title, vn\_counter, 1));  vn\_counter := vn\_counter + 1;  --END OF LOOP  END LOOP;    vn\_counter :=1;  DBMS\_OUTPUT.PUT\_LINE('--Firstname--');  vn\_length := LENGTH(vc\_firstname);  --WHILE LOOP TERMINATION CONDITION  WHILE vn\_counter<=vn\_length LOOP  DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_firstname, vn\_counter, 1));  vn\_counter := vn\_counter + 1;  --END OF LOOP  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('--Surname--');  vn\_length := LENGTH(vc\_surname);  --FOR LOOP TERMINATION CONDITION  FOR vn\_counter IN 1 .. vn\_length LOOP  DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_surname, vn\_counter, 1));  --END OF LOOP  END LOOP;  --END OF PROGRAMME  END;  / | --Title--  M  R  S  --Firstname--  P  R  A  T  I  M  A  --Surname--  D  A  H  A  L  PL/SQL procedure successfully completed. |  | No action Perfomed |
|  | DECLARE  c\_id staff.staff\_id%type := 1000;  c\_name staff.staff\_firstname%type;  BEGIN  SELECT staff\_firstname INTO c\_name  FROM staff  WHERE staff\_id = c\_id;  DBMS\_OUTPUT.PUT\_LINE ('Name: '|| c\_name);  EXCEPTION  WHEN no\_data\_found THEN  dbms\_output.put\_line('No such staff!');  WHEN others THEN  dbms\_output.put\_line('Error!');  END;  / | Name: PRATIMA  PL/SQL procedure successfully completed. |  | No any action required |
| 1. 3 | Creating varray to use in procedure  CREATE OR REPLACE TYPE team\_varray\_type AS VARRAY(50) OF NUMBER(11);  / | Type created. |  | No action needed |
| 1. 4 | CREATE OR REPLACE PROCEDURE proc\_create\_team IS  vc\_command VARCHAR2(2000);  BEGIN  vc\_command:='CREATE TABLE team (team\_id NUMBER(7),team\_name VARCHAR2(100),members team\_varray\_type,staff\_id NUMBER(11))';  EXECUTE IMMEDIATE vc\_command;  END;  / | Procedure created. |  | No action needed |
| 1. 5 | CREATE OR REPLACE PROCEDURE proc\_primary\_team IS  vc\_command VARCHAR2(2000);  BEGIN  vc\_command:='ALTER TABLE team  ADD CONSTRAINT pk\_team  PRIMARY KEY(team\_id) ';  EXECUTE IMMEDIATE vc\_command;  END;  / | Procedure created. |  | No action needed |
| 1. 6 | CREATE OR REPLACE PROCEDURE proc\_foreign\_team IS  vc\_command VARCHAR2(2000);  BEGIN  vc\_command:='ALTER TABLE team  ADD CONSTRAINT fk\_t\_staff  FOREIGN KEY(staff\_id)  REFERENCES staff(staff\_id)';  EXECUTE IMMEDIATE vc\_command;  END;  /  SHOW ERRORS; | Procedure created. |  | No action needed |
| 1. 7 | CREATE OR REPLACE PROCEDURE proc\_staff IS  vc\_staff\_firstname staff.staff\_firstname%TYPE :='AMRIT';  vc\_staff\_surname staff.staff\_surname%TYPE := 'THAPA';  BEGIN  INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname)  VALUES(seq\_staff.NEXTVAL,vc\_staff\_firstname,vc\_staff\_surname);  END proc\_staff;  /  SHOW ERRORS | Procedure created. |  | No action needed |
| 1. 8 | CREATE OR REPLACE PROCEDURE proc\_update\_staff IS  vn\_staff\_id staff.staff\_id%TYPE;  BEGIN  SELECT seq\_staff.CURRVAL  INTO vn\_staff\_id  FROM DUAL;    UPDATE staff SET staff\_firstname ='SAMI'  WHERE staff\_id =vn\_staff\_id;    END proc\_update\_staff;  /  SHOW ERRORS  **Testing**  SELECT staff\_id,staff\_firstname,staff\_surname  FROM staff;  EXEC proc\_update\_staff; | Procedure created. |  | No action needed |
| 1. 9 | CREATE OR REPLACE PROCEDURE proc\_delete\_staff IS  vn\_staff\_id staff.staff\_id%TYPE;  BEGIN  SELECT seq\_staff.CURRVAL  INTO vn\_staff\_id  FROM DUAL;    DELETE FROM staff  WHERE staff\_id=vn\_staff\_id;    END proc\_delete\_staff;  /  SHOW ERRORS; |  |  | No action needed |
| 1. 10 | CREATE OR REPLACE PROCEDURE proc\_get\_user\_salary IS  vn\_staff NUMBER(11) := 1000;  vc\_username VARCHAR2(20);  vn\_salary NUMBER(8,2);  vn\_ceilsalary NUMBER(10);  vn\_floorsalary NUMBER(10);  vn\_roundsalary NUMBER(10);  vn\_truncatesalary NUMBER(10);  vn\_length NUMBER(30);  vn\_counter NUMBER(30) :=3;  BEGIN  SELECT COUNT(\*) INTO vn\_length FROM staff;  DBMS\_OUTPUT.PUT\_LINE('Username Salary Ceil Floor Round Truncate');  LOOP  SELECT staff\_salary INTO vn\_salary FROM staff WHERE staff\_id = vn\_staff;  SELECT CEIL(vn\_salary) INTO vn\_ceilsalary FROM staff WHERE staff\_id = vn\_staff;  SELECT FLOOR(vn\_salary) INTO vn\_floorsalary FROM staff WHERE staff\_id = vn\_staff;  SELECT ROUND(vn\_salary) INTO vn\_roundsalary FROM staff WHERE staff\_id = vn\_staff;  SELECT TRUNC(vn\_salary) INTO vn\_truncatesalary FROM staff WHERE staff\_id = vn\_staff;  SELECT CONCAT(SUBSTR(INITCAP(TRIM(TRAILING ' ' FROM staff\_firstname)),1,2),SUBSTR(INITCAP(staff\_surname),1,5))  INTO vc\_username  FROM staff  WHERE staff\_id = vn\_staff;  EXIT WHEN vn\_counter = vn\_length;  DBMS\_OUTPUT.PUT\_LINE(vc\_username || ' ' || vn\_salary || ' '|| vn\_ceilsalary || ' '|| vn\_floorsalary || ' ' || vn\_roundsalary || ' ' || vn\_truncatesalary);        vn\_staff := vn\_staff+2;  vn\_length := vn\_length-1;  END LOOP;--END OF LOOP  END;  /  SHOW ERRORS;  **TESTING**  SELECT staff\_salary FROM staff;  EXEC proc\_get\_user\_salary; | Procedure created. |  | No action needed |
| 1. 11 | DECLARE  vn\_length NUMBER(2);  vn\_counter NUMBER(3) := 1;  vc\_firstname VARCHAR2(20) := 'ANUJ';  vc\_surname VARCHAR2(20) := 'MAGAR';  BEGIN  vn\_length := LENGTH(vc\_firstname);  --START THE LOOP  LOOP  --LOOP TERMINATION CONDITION  EXIT WHEN vn\_counter>vn\_length;  DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_firstname, vn\_counter, 1));  vn\_counter := vn\_counter + 1;  --END OF LOOP  END LOOP;  --DISPLAY SURNAME  DBMS\_OUTPUT.PUT\_LINE(vc\_surname);  --END OF PROGRAMME  END;  / | A  N  U  J  MAGAR  PL/SQL procedure successfully completed. |  | No action performed |
| 1. 12 | CREATE OR REPLACE PROCEDURE proc\_display\_staff(in\_id staff.staff\_id%TYPE)IS  vc\_name staff.staff\_firstname%TYPE;  vc\_lastname staff.staff\_surname%TYPE;  vc\_gender staff.staff\_gender%TYPE;  vd\_date staff.staff\_hire\_date%TYPE;  vn\_salary staff.staff\_salary%TYPE;  BEGIN  SELECT staff\_firstname INTO vc\_name FROM staff WHERE staff\_id =in\_id;  SELECT staff\_surname INTO vc\_lastname FROM staff WHERE staff\_id =in\_id;  SELECT staff\_gender INTO vc\_gender FROM staff WHERE staff\_id =in\_id;  SELECT staff\_hire\_date INTO vd\_date FROM staff WHERE staff\_id =in\_id;  SELECT staff\_salary INTO vn\_salary FROM staff WHERE staff\_id =in\_id;  DBMS\_OUTPUT.PUT\_LINE('The details of staff with id ' ||in\_id || ' is: ' || vc\_name || ' '|| vc\_lastname ||' gender is: '|| vc\_gender ||' and the date of birth is: '|| vd\_date ||' and salary is: '|| vn\_salary);  END proc\_display\_staff;  /  SHOW ERRORS;  **Testing**  EXEC proc\_display\_staff(1000); | Procedure created. |  | No action perfomed |
| 1. 13 | CREATE OR REPLACE PROCEDURE proc\_insert\_staff(in\_staff\_firstname VARCHAR2,in\_staff\_surname VARCHAR2,in\_staff\_hire\_date DATE,in\_staff\_dob DATE) IS  vc\_firstname staff.staff\_firstname%TYPE;  vc\_surname staff.staff\_surname%TYPE;  vd\_hire\_date staff.staff\_hire\_date%TYPE;  vd\_dob staff.staff\_hire\_date%TYPE;  vc\_command VARCHAR2(1000);  BEGIN  SELECT in\_staff\_firstname INTO vc\_firstname FROM DUAL;  SELECT in\_staff\_surname INTO vc\_surname FROM DUAL;  SELECT in\_staff\_hire\_date INTO vd\_hire\_date FROM DUAL;  SELECT in\_staff\_dob INTO vd\_dob FROM DUAL;  INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_hire\_date,staff\_dob)  VALUES(seq\_staff.NEXTVAL,vc\_firstname,vc\_surname,vd\_hire\_date,vd\_dob);  END proc\_insert\_staff;  /  SHOW ERRORS; | Procedure created. |  | No action perfomed |
|  |  |  | **FUNCTIONS** |  |
| 1. 14 | **Function**  CREATE OR REPLACE FUNCTION func\_festivals  RETURN NUMBER IS  vn\_festivals NUMBER(3);  BEGIN  SELECT COUNT(\*)  INTO vn\_festivals  FROM festivals;  RETURN vn\_festivals;  END func\_festivals;  /  SHOW ERRORS  **Procedure**:  CREATE OR REPLACE PROCEDURE proc\_festivals IS  vc\_festival\_name VARCHAR2(50);  BEGIN  vc\_festival\_name:=func\_festivals;  DBMS\_OUTPUT.PUT\_LINE('The number of festivals is' || func\_festivals);  END proc\_festivals;  /  SHOW ERRORS; | Function created.  Procedure created. |  | No action perfomed |
| 1. 15 | **Function:**  CREATE OR REPLACE FUNCTION func\_get\_avg\_age\_of\_staff  RETURN NUMBER IS  today DATE;  vn\_avg\_age NUMBER(3);  BEGIN  SELECT sysdate INTO today from dual;  SELECT AVG(FLOOR(today-staff\_dob) / 365.2425) INTO vn\_avg\_age FROM staff  WHERE staff\_dob IS NOT NULL;  RETURN vn\_avg\_age;  END func\_get\_avg\_age\_of\_staff;  /  SHOW ERRORS;  **Procedure:**  CREATE OR REPLACE PROCEDURE proc\_avg\_age\_staff IS  vn\_avg\_age NUMBER(3);  BEGIN  vn\_avg\_age:=func\_get\_avg\_age\_of\_staff;  DBMS\_OUTPUT.PUT\_LINE('THE AVERAGE AGE OF STAFF IS ' || vn\_avg\_age || ' YEARS.');  END proc\_avg\_age\_staff;  /  SHOW ERRORS; | Function created.  Procedure created. |  | No  Action perfomed. |
| 1. 16 | **Function:**  CREATE OR REPLACE FUNCTION func\_staff(in\_staff\_id NUMBER) RETURN VARCHAR2 IS  vc\_stff\_un VARCHAR2(10);  BEGIN  SELECT CONCAT(SUBSTR(staff\_firstname,1,2),SUBSTR(staff\_surname,1,5))  INTO vc\_stff\_un  FROM staff  WHERE staff\_id = in\_staff\_id;  RETURN vc\_stff\_un;  END func\_staff;  /  **Procedure:**  CREATE OR REPLACE PROCEDURE proc\_staff(in\_staff\_id staff.staff\_id%TYPE) IS  vc\_staff\_username VARCHAR2(30);  --vc prefix is used for the new variable because we are storing varchar2(string) data  BEGIN  vc\_staff\_username := func\_staff(in\_staff\_id);  DBMS\_OUTPUT.PUT\_LINE ('The staff with user id ' || in\_staff\_id || ' has a username '||vc\_staff\_username);  END proc\_staff;--END OF THE PROCEDURE  / | Function created.  Procedure created. |  | No action needed. |
| 1. 17 | **Function:**  CREATE OR REPLACE FUNCTION func\_staff\_check\_gender(in\_staff\_id NUMBER) RETURN VARCHAR2 IS  vc\_stff\_gender staff.staff\_gender%TYPE;  BEGIN  SELECT staff\_gender INTO vc\_stff\_gender FROM staff  WHERE staff\_id = in\_staff\_id;  RETURN vc\_stff\_gender;  END func\_staff\_check\_gender;  /  **Procedure:**  CREATE OR REPLACE PROCEDURE proc\_staff\_gender(in\_staff\_id staff.staff\_id%TYPE) IS  vc\_staff\_gender staff.staff\_gender%TYPE;  BEGIN  vc\_staff\_gender := func\_staff\_check\_gender(in\_staff\_id);  CASE  WHEN vc\_staff\_gender = 'M' THEN DBMS\_OUTPUT.PUT\_LINE('STAFF IS MALE');  WHEN vc\_staff\_gender = 'F' THEN DBMS\_OUTPUT.PUT\_LINE('STAFF IS FEMALE');  ELSE DBMS\_OUTPUT.PUT\_LINE('OTHERS');  END CASE;  END proc\_staff\_gender;  / | Function created.  Procedure created. |  | No action perfeomed. |
| 1. 18 | **Function:**  CREATE OR REPLACE FUNCTION func\_total\_salary RETURN NUMBER IS  vn\_staff\_salary NUMBER(10,2);  BEGIN  SELECT SUM(staff\_salary)  INTO vn\_staff\_salary  FROM staff;  RETURN vn\_staff\_salary;  END func\_total\_salary;  /  **Procedure:**  CREATE OR REPLACE PROCEDURE proc\_salary\_func IS  vn\_total\_staff\_salary NUMBER(10,2);  BEGIN  vn\_total\_staff\_salary := func\_total\_salary;  DBMS\_OUTPUT.PUT\_LINE ('The total salary of staff is '|| vn\_total\_staff\_salary );  END proc\_salary\_func;  / | Function created.  Procedure created. |  |  |
|  |  |  | **TRIGGERS** |  |
|  | 1. **Functions**   CREATE OR REPLACE FUNCTION func\_get\_staff\_age (in\_staff\_dob DATE)  RETURN NUMBER IS  vn\_staff\_age NUMBER(3);  BEGIN  vn\_staff\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, in\_staff\_dob)/12);  RETURN vn\_staff\_age;  END func\_get\_staff\_age;  /  SHOW ERRORS;   1. **Procedure**   CREATE OR REPLACE PROCEDURE proc\_staff\_age(in\_staff\_id NUMBER) IS  vd\_dob DATE;  vn\_age NUMBER(3);  BEGIN  SELECT staff\_dob  INTO vd\_dob  FROM staff  WHERE staff\_id = in\_staff\_id;  vn\_age := func\_get\_staff\_age(vd\_dob);  DBMS\_OUTPUT.PUT\_LINE('Staff is of age ' || vn\_age);  END;  /  SHOW ERRORS;   * 1. **Testing of procedure**   EXEC proc\_staff\_age(1000);   1. **Trigger**   CREATE OR REPLACE TRIGGER trig\_staff\_age  BEFORE UPDATE OF staff\_dob OR INSERT ON staff  FOR EACH ROW  WHEN(NEW.staff\_dob IS NOT NULL)  DECLARE  vd\_today DATE;  --vn\_age NUMBER(3);  BEGIN  SELECT SYSDATE  INTO vd\_today  FROM DUAL;    IF :NEW.staff\_dob>(vd\_today-(365\*18)) THEN  RAISE\_APPLICATION\_ERROR(-20000,'The staff is underage');  --vn\_age:=func\_get\_staff\_age(:NEW.staff\_dob);  ELSIF :NEW.staff\_dob>SYSDATE THEN  RAISE\_APPLICATION\_ERROR(-20000,'The staff is not born yet');    END IF;    END trig\_staff\_age;  /  SHOW ERRORS;   * 1. **Testing of triggers**   3.1.1 **Inserting underaged value**  INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)  VALUES (seq\_staff.NEXTVAL,'ARYAN','KARKI','01-JAN-2006');  3.2.2 **inserting correct value**  INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)  VALUES (seq\_staff.NEXTVAL,'ARYAN','KARKI','01-JAN-1996'); | Function created.  Procedure Created.  Show the age of the staff and procedure successfully compiled.  Trigger created  Give an error message as the staff is underage.  1 row created |  | No action needed  No action needed  No action needed  No action needed  Insert a date of birth whose age is more than 18.  No action needed |
|  | 1. **Creating table**   CREATE TABLE hr\_event(  event\_type VARCHAR2(30),  login\_date DATE,  logon\_time VARCHAR2(15),  logoff\_date DATE,  logoff\_time VARCHAR2(15)  );   1. **Creating triggers**   CREATE OR REPLACE trigger trig\_login\_hr  AFTER LOGON ON SCHEMA  declare  logons VARCHAR2(30);  BEGIN  DBMS\_OUTPUT.PUT\_LINE('one line Updated by '|| SYSDATE);  INSERT INTO hr\_event VALUES(  ora\_sysevent,  SYSDATE,  TO\_CHAR(sysdate , 'hh24:mi:ss'),  NULL,  NULL  );  COMMIT;  END trig\_login\_hr;  /   * 1. **Testing**   2.1.1 SELECT \* FROM hr\_event;  2.1.2 DISC;  CONNECT username/password;  2.1.3 SELECT \* FROM hr\_event; | Table created.  Trigger Created.  No rows Selected.  1 rows returned. |  | No action needed.  No action needed.  No action needed.  No action needed.  No action needed. |
|  | 1. **Creating trigger**   CREATE OR REPLACE TRIGGER trig\_operation\_staff  BEFORE INSERT OR UPDATE OR DELETE ON STAFF  FOR EACH ROW  ENABLE  DECLARE  c\_user VARCHAR2(15);  BEGIN  SELECT  USER INTO c\_user FROM dual;  IF INSERTING THEN  DBMS\_OUTPUT.PUT\_LINE('One line inserted by '||c\_user ||' on ' || SYSDATE);  ELSIF UPDATING THEN  DBMS\_OUTPUT.PUT\_LINE('One line Updated by '||c\_user||' on ' || SYSDATE);  ELSIF DELETING THEN  DBMS\_OUTPUT.PUT\_LINE('One line Deleted by '||c\_user ||' on ' || SYSDATE);  END IF;  END;  /   * 1. **Testing**   1.1.1 INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)  VALUES (seq\_staff.NEXTVAL,'RANEET','KUNWAR','01-JAN-1996');    1.1.2 UPDATE staff  SET staff\_gender = 'M'  WHERE staff\_id = 1000;  1.1.3 DELETE staff WHERE staff\_id=1028; | Trigger created  1 row created.  One line inserted by username on today date.  1 row updated.  One line updated by username on today date.  1 row deleted.  One line deleted by username on today date. |  | No action needed.  No action needed.  No action needed.  No action needed. |
|  | 1. **Creating trigger**   CREATE OR REPLACE TRIGGER trig\_audit\_tr  AFTER DDL ON DATABASE  BEGIN  DBMS\_OUTPUT.PUT\_LINE('one row '||ora\_sysevent || '...... ON ' || SYSDATE);  END;  /  SHOW ERRORS;   1. **Testing**    1. CREATE TABLE jname(dob DATE);   2.2 ALTER TABLE jname  ADD(salary NUMBER(2,3));  2.3 DROP TABLE jname; | Trigger created.  Table created.  One row create on today date.  Table altered.  One row alter on today date.  Table dropped.  One row alter on today date.  One row drop on today date. |  | No action needed.  No action needed.  No action needed.  No action needed. |
|  | CREATE OR REPLACE TRIGGER trig\_message  AFTER STARTUP ON DATABASE  BEGIN  DBMS\_OUTPUT.PUT\_LINE('Hello..');  END;  / | Trigger created. |  | No action needed. |
|  |  |  | **CURSOR** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LXVII | 1.**IMPLICIT CURSOR**  CREATE OR REPLACE PROCEDURE proc\_fes\_imp\_cursor(in\_festival\_name festivals.festival\_name%TYPE) IS  BEGIN  DELETE FROM festivals WHERE festival\_name = in\_festival\_name;    END proc\_fes\_imp\_cursor;  /  SHOW ERRORS  **2.Testing**  INSERT INTO festivals(festival\_nature\_id,location\_id,festival\_name) VALUES(50006,5000,'DANCE FESTIVALS'**);**  SELECT festival\_nature\_id, location\_id,festival\_name FROM festivals;  EXEC proc\_fes\_imp\_cursor('DANCE FESTIVALS');  SELECT festival\_nature\_id,location\_id,festival\_name FROM festivals; | Cursor created.  1 row created |  | No action needed.  No action needed. |
| LXVIII | **2.IMPLICIT CURSOR**  CREATE OR REPLACE PROCEDURE proc\_imp\_cursor(in\_festival\_name festivals.festival\_name %TYPE) IS  BEGIN  DELETE FROM festivals WHERE festival\_name =in\_festival\_name ;  IF(SQL%FOUND)THEN  DBMS\_OUTPUT.PUT\_LINE('DELETED ' || in\_festival\_name );  ELSEDBMS\_OUTPUT.PUT\_LINE('NO SUCH FESTIVAL');  END IF;END proc\_imp\_cursor;  /  SHOW ERRORS;  No errors  **2.Testing**  SELECT festival\_nature\_id,location\_id,festival\_name FROM festivals;  INSERT INTO festivals(festival\_nature\_id,location\_id,festival\_name)  VALUES(50005,5001,'MUSICAL FESTIVAL');  SELECT festival\_nature\_id,location\_id,festival\_name FROM festivals;  EXEC proc\_imp\_cursor('MUSICAL FESTIVAL');  SELECT festival\_nature\_id,location\_id,festival\_name FROM festivals; | Cursor created  6 rows selected.  1row is inseted.  7 rows returned.  Procedure is executed successfully |  | No action needed. |
| LXVIX  LXVX  LXXVI | 3.**EXPLICIT CURSOR**  CREATE OR REPLACE PROCEDURE proc\_exp\_cursor(in\_staff staff.staff\_salary%TYPE) IS  CURSOR cur\_staff\_salary IS  SELECT staff\_firstname,staff\_salary FROM staff WHERE staff\_salary>=in\_staff;  vn\_rows NUMBER(3):=0;  BEGIN  DBMS\_OUTPUT.PUT\_LINE('STAFF SALARY IS');  DBMS\_OUTPUT.PUT\_LINE('--------------------------');  FOR rec\_cur\_staff\_salary IN cur\_staff\_salary LOOP  DBMS\_OUTPUT.PUT\_LINE(cur\_staff\_salary%ROWCOUNT||' '||rec\_cur\_staff\_salary.staff\_firstname||' '||rec\_cur\_staff\_salary.staff\_salary);  vn\_rows:=cur\_staff\_salary%ROWCOUNT;  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('THERE ARE '||vn\_rows||' ROWS');  END proc\_exp\_cursor;  /  SHOW ERRORS  No errors.  **2.TESTING**  SELECT staff\_firstname,staff\_salary FROM staff  EXEC proc\_exp\_cursor(50000);  EXEC proc\_exp\_cursor(80000);  **4.EXPLICIT CURSOR USING LOOP**  CREATE OR REPLACE PROCEDURE proc\_exp\_staff\_salary(in\_staff\_salary staff.staff\_salary%TYPE) IS  CURSOR cur\_salary IS  SELECT staff\_id, staff\_firstname, staff\_salary FROM staff WHERE staff\_salary>= in\_staff\_salary;  rec\_cur\_salary cur\_salary%ROWTYPE;  BEGIN  OPEN cur\_salary;  FETCH cur\_salary INTO rec\_cur\_salary;  IF cur\_salary%NOTFOUND THEN  DBMS\_OUTPUT.PUT\_LINE('NO SUCH STAFF WITH SUCH SALARY FOUND');  ELSE  WHILE cur\_salary%FOUND LOOP  DBMS\_OUTPUT.PUT\_LINE(rec\_cur\_salary.staff\_id||' '||rec\_cur\_salary.staff\_firstname||'    '||rec\_cur\_salary.staff\_salary);  FETCH cur\_salary INTO rec\_cur\_salary;    END LOOP;  END IF;  END proc\_exp\_staff\_salary;  /  SHOW ERRORS  **2.TESTING**  SELECT staff\_id, staff\_firstname, staff\_salary FROM staff;  EXEC proc\_exp\_staff\_salary(10000);  EXEC proc\_exp\_staff\_salary(99999);  **5.EXPLICIT CURSOR**  CREATE OR REPLACE PROCEDURE proc\_exp\_cursor(in\_festival\_nature\_id festival\_staff.festival\_nature\_id%TYPE) IS  CURSOR cur\_festival IS  SELECT festival\_name,occuring\_date FROM festivals WHERE festival\_nature\_id =in\_festival\_nature\_id ;  vn\_rows NUMBER(3):=0;  BEGIN  DBMS\_OUTPUT.PUT\_LINE('SN COURSES COST');  DBMS\_OUTPUT.PUT\_LINE('---------------------------------');  FOR rec\_cur\_festival IN cur\_festival LOOP  DBMS\_OUTPUT.PUT\_LINE(cur\_festival%ROWCOUNT||' '||rec\_cur\_festival.festival\_name||' '||rec\_cur\_festival.occuring\_date);  vn\_rows:=cur\_festival%ROWCOUNT;  END LOOP;    DBMS\_OUTPUT.PUT\_LINE('THERE ARE '||vn\_rows||' ROWS');  END proc\_exp\_cursor;  SHOW ERRORS  2**.TESTING**  SELECT festival\_name,occuring\_date FROM festivals WHERE festival\_nature\_id =50004 ;  EXEC proc\_exp\_cursor(500000); | Implicit cursor is sucessufull.  10 rows returned.  salary 50000 or more executed than 7 rows returned.  0 rows returned with staff salary more than80000  Procedure successfully executed.  10 rows selected.  Staff with salary above 10000 displayed.  No staff with salary 99999 found.  Procedure create.  2 rows returned.  2 rows returned.  0 rows returned |  | No action requires.  No action required.  No action required.  No action required.  No action required.  No action required.  No action required.  No action required.  No action required. |
| **PACKAGES** | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LXXVII | **1.CREATE PACKAGE HEADER:**  CREATE OR REPLACE PACKAGE pkg\_search\_staff AS  FUNCTION func\_search\_by\_email\_cursor(in\_email VARCHAR2) RETURN NUMBER;  PROCEDURE proc\_search\_by\_email\_cursor(in\_email VARCHAR2);  END pkg\_search\_staff;  /  **CREATE PACKAGE BODY:**  CREATE OR REPLACE PACKAGE BODY pkg\_search\_staff AS  vn\_staff\_id staff.staff\_id%TYPE;  vc\_staff\_firstname staff.staff\_firstname%TYPE;  vc\_staff\_surname staff.staff\_surname%TYPE;  vc\_staff\_gender staff.staff\_gender%TYPE;  exep\_no\_user EXCEPTION; --EXCEPTION  CURSOR cur\_staff\_id IS  SELECT s.staff\_id staff\_id,c.email email FROM staff s, table(s.staff\_contact) c;  rec\_cur\_id cur\_staff\_id%ROWTYPE;  --FUNCTION TO GET USER ID BASED ON EMAIL  FUNCTION func\_search\_by\_email\_cursor(in\_email VARCHAR2)  RETURN NUMBER IS  BEGIN IF cur\_staff\_id %ISOPEN THEN  CLOSE cur\_staff\_id ;  END IF;  OPEN cur\_staff\_id;  FETCH cur\_staff\_id INTO rec\_cur\_id;  --IF THERE IS NO DATA IN STAFF TABLE  IF cur\_staff\_id%NOTFOUND THEN  RETURN 0;  ELSE  --WHILE LOOP TO SCROLL THROUGH ALL DATA  WHILE cur\_staff\_id%FOUND LOOP  IF rec\_cur\_id.email = in\_email THEN  RETURN rec\_cur\_id.staff\_id;  ELSE RETURN 0;  END IF;  END LOOP;  END IF;  CLOSE cur\_staff\_id;  END;  --PROCEDURE WITHIN PACKAGE TO DISPLAY INFORMATION  PROCEDURE proc\_search\_by\_email\_cursor(in\_email VARCHAR2) IS BEGIN  vn\_staff\_id := func\_search\_by\_email\_cursor(in\_email);  IF (vn\_staff\_id=0) THEN  RAISE exep\_no\_user;  ELSE  SELECT staff\_firstname INTO vc\_staff\_firstname FROM staff WHERE staff\_id = vn\_staff\_id;  SELECT staff\_surname INTO vc\_staff\_surname FROM staff WHERE staff\_id = vn\_staff\_id;  SELECT staff\_gender INTO vc\_staff\_gender FROM staff WHERE staff\_id = vn\_staff\_id;  DBMS\_OUTPUT.PUT\_LINE('Staff\_id: '|| vn\_staff\_id ||' FIRSTNAME: ' || vc\_staff\_firstname || ' SURNAME: ' || vc\_staff\_surname);  END IF;  EXCEPTION  WHEN exep\_no\_user THEN  DBMS\_OUTPUT.PUT\_LINE('NOT FOUND');  END;  END pkg\_search\_staff;  /  **2.DISPLAYING PACKAGE**  SELECT OBJECT\_NAME FROM USER\_OBJECTS WHERE OBJECT\_NAME LIKE 'PKG\_%';  **3.TESTING**  EXEC pkg\_search\_staff.proc\_search\_by\_email\_cursor('PRATIMA@GMAIL.COM');  EXEC pkg\_search\_staff.proc\_search\_by\_email\_cursor('ANOOZ@GMAIL.COM'); | PACKAGE HEADER IS CREATED  Package is displayed  Package is tested sucessfully  Package is tested sucessfully |  | No action required. |

# CONCLUSION

Lastly working with this project was a good experience for a programmer as it was quite difficult to start the project at the beginning but when all the group members performed equal works together and solved all the problems by a group discussion. This project was completed successfully. Further, changes can be made if the sufficient time was given and project can be made even more better and flexible.

# REFERENCES

SolarWinds. 2019. SQL Server Exception Handling with TRY, CATCH and THROW. [ONLINE] Available at: Using Parameters for SQL Server Queries and Stored Procedures. 2019. Using Parameters for SQL Server Queries and Stored Procedures. [ONLINE] Available at: <https://www.mssqltips.com/sqlservertip/2981/using-parameters-for-sql-server-queries-and-stored-procedures/>. [Accessed 04 May 2019]. [Accessed 14 April 2019].

Using Parameters for SQL Server Queries and Stored Procedures. 2019. Using Parameters for SQL Server Queries and Stored Procedures. [ONLINE] Available at: Using Parameters for SQL Server Queries and Stored Procedures. 2019. Using Parameters for SQL Server Queries and Stored Procedures. [ONLINE] Available at: <https://www.mssqltips.com/sqlservertip/2981/using-parameters-for-sql-server-queries-and-stored-procedures/>. [Accessed 04 May 2019].. [Accessed 16 April 2019].

# APPENDIX

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: RUN Testing Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\run\_testing\_09.txt

\*/

--CREATE RUN SCRIPT

@E:\DatabaseYEAR2\assignment\create\_09.txt

--CREATE ALTER SCRIPT

@E:\DatabaseYEAR2\assignment\constraints\_09.txt

--TRIGGER SCRIPT

@E:\DatabaseYEAR2\assignment\trigger\_09.txt

--CREATE INSERT SCRIPT

@E:\DatabaseYEAR2\assignment\insert\_09.txt

--QUERY SCRIPT

@E:\DatabaseYEAR2\assignment\query\_09.txt

--PROCEDURE SCRIPT

@E:\DatabaseYEAR2\assignment\procedure\_09.txt

--FUNCTION SCRIPT

@E:\DatabaseYEAR2\assignment\functions\_09.txt

--CURSOR SCRIPT

@E:\DatabaseYEAR2\assignment\cursor\_09.txt

--PACKAGE SCRIPT

@E:\DatabaseYEAR2\assignment\package\_09.txt

--DROP SCRIPT

@E:\DatabaseYEAR2\assignment\drop\_09.txt

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Create Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\create\_09.txt

\*/

SET SERVEROUTPUT ON;

SET LINESIZE 180

--CREATING THE OBJECT TYPE TO CREATE NESTED TABLE

CREATE OR REPLACE TYPE address\_type AS OBJECT(

country VARCHAR2(25),

city VARCHAR2(50),

street VARCHAR2(255));

/

--CHECKING IF THE OBJECT TYPE HAS BEEN CREATED WITHOUT ERRORS

SHOW ERRORS

--TESTING IF OBJECT\_TABLE CREATED

DESC USER\_OBJECTS

SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_TYPE LIKE 'TABLE';

--CREATING THE OBJECT TYPE TO CREATE SIMPLE TABLE TYPE

CREATE OR REPLACE TYPE social\_media\_type AS OBJECT(

social\_media\_name VARCHAR2(50),

social\_media\_username VARCHAR2(50),

contact VARCHAR2(15));

/

--CHECKING IF THE OBJECT TYPE HAS BEEN CREATED WITHOUT ERRORS

SHOW ERRORS

--TESTING IF OBJECT\_TABLE CREATED

DESC USER\_OBJECTS

SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_TYPE LIKE 'TABLE';

--CREATING THE OBJECT TYPE TO CREATE VARRAY

CREATE OR REPLACE TYPE contact\_type AS OBJECT(

email VARCHAR2(50),

phone VARCHAR2(15),

mobile VARCHAR2(15));

/

--CHECKING IF THE OBJECT TYPE HAS BEEN CREATED WITHOUT ERRORS

SHOW ERRORS

--TESTING IF OBJECT\_TTYPE CREATED

DESC USER\_OBJECTS

SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_TYPE LIKE 'TYPE';

--CREATING OBJECT TABLE NAMED ADDRESSES BASED ON OBJECT\_TYPE address\_type

CREATE TABLE addresses OF address\_type(

city DEFAULT 'KATHMANDU',

country DEFAULT 'NEPAL');

--TESTING FOR ADDRESSES

SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_TYPE LIKE 'TABLE';

--CREATING A VARRAY WITH 50 ELEMENTS

CREATE TYPE contact\_varray\_type AS VARRAY(50) OF contact\_type;

/

--CHECKING IF THE OBJECT TYPE HAS BEEN CREATED WITHOUT ERRORS

SHOW ERRORS

--CREATING OBJECT TABLE TYPE NAMED social\_media\_table\_type

CREATE TYPE social\_media\_table\_type AS TABLE OF social\_media\_type;

/

--CHECKING IF THE OBJECT TYPE HAS BEEN CREATED WITHOUT ERRORS

SHOW ERRORS

--TESTING IF OBJECT\_TABLE CREATED

SELECT OBJECT\_NAME, OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_TYPE L

--DESCRIBING the social\_media\_table\_type

DESC social\_media\_table\_type;

--CREATING TABLE locations

CREATE TABLE locations(

location\_id NUMBER(11),

location\_address address\_type);

--SHOWING THE STRUCTURE OF LOCATIONS

DESC locations;

--CHECKING THE DATA FROM LOCATIONS

SELECT \* FROM locations;

--CREATING TABLE festival\_nature

CREATE TABLE festival\_nature(

festival\_nature\_id NUMBER(11),

festival\_type VARCHAR2(50));

--SHOWING THE STRUCTURE OF FESTIVAL\_NATURE

DESC festival\_nature

--CHECKING THE DATA FROM FESTIVAL\_NATURE

SELECT \* FROM festival\_nature;

--CREATING TABLE festivals

CREATE TABLE festivals(

festival\_nature\_id NUMBER(11),

location\_id NUMBER(11),

festival\_name VARCHAR2(50),

occuring\_date DATE);

--SHOWING THE STRUCTURE OF FESTIVALS

DESC festivals;

--CHECKING THE DATA FROM FESTIVALS

SELECT \* FROM festivals;

--CREATING TABLE staff

CREATE TABLE staff(

staff\_id NUMBER(11),

staff\_title VARCHAR2(5) DEFAULT 'MRS',

staff\_firstname VARCHAR2(30),

staff\_surname VARCHAR2(30),

staff\_contact contact\_varray\_type,

staff\_address REF address\_type SCOPE IS addresses,

social\_media\_used social\_media\_table\_type)

NESTED TABLE social\_media\_used STORE AS social\_media\_table;

--SHOWING THE STRUCTURE OF STAFF

DESC staff;

--CHECKING THE DATA FROM STAFF

SELECT \* FROM staff;

--CREATING TABLE festival\_staff

CREATE TABLE festival\_staff(

staff\_id NUMBER(11) NOT NULL,

location\_id NUMBER(11) NOT NULL,

festival\_nature\_id NUMBER(11) NOT NULL);

--SHOWING THE STRUCTURE OF FESTIVAL\_STAFF

DESC festival\_staff;

--CHECKING THE DATA FROM FESTIVAL\_STAFF

SELECT \* FROM festival\_staff;

--CREATING SEQUENCE seq\_locations

CREATE SEQUENCE seq\_locations

INCREMENT BY 1

START WITH 5000

NOCYCLE;

--CREATING SEQUENCE seq\_festival\_nature

CREATE SEQUENCE seq\_festival\_nature

INCREMENT BY 1

START WITH 50001

NOCYCLE;

--CREATING SEQUENCE seq\_staff

CREATE SEQUENCE seq\_staff

INCREMENT BY 2

START WITH 1000

MAXVALUE 10000000000;

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Alter Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\constraints\_09.txt;

\*/

--ALTERING STAFF TABLE TO ADD NEW COLUMN staff\_gender

ALTER TABLE staff

ADD (staff\_gender CHAR);

--ALTERING STAFF TABLE TO ADD NEW COLUMN staff\_blood\_group

ALTER TABLE staff

ADD (staff\_blood\_group CHAR(3));

--ALTERING STAFF TABLE TO ADD NEW COLUMN staff\_hire\_date

ALTER TABLE staff

ADD (staff\_hire\_date DATE DEFAULT SYSDATE);

--ALTERING STAFF TABLE TO ADD NEW COLUMN staff\_salary

ALTER TABLE staff

ADD (staff\_salary NUMBER(8,2));

--ALTERING STAFF TABLE TO ADD NEW COLUMN staff\_dob

ALTER TABLE staff

ADD (staff\_dob DATE );

--ALTERING STAFF TABLE TO ADD NEW COLUMN leader

ALTER TABLE STAFF

ADD (leader NUMBER(11));

--ADDING DEFAULT VALUES

ALTER TABLE staff

MODIFY staff\_gender DEFAULT 'M';

ALTER TABLE festivals

MODIFY occuring\_date DEFAULT SYSDATE;

--Primary key

--for locations

ALTER TABLE locations

ADD CONSTRAINT pk\_locations

PRIMARY KEY(location\_id);

ALTER TABLE festival\_nature

ADD CONSTRAINT pk\_festival\_nature

PRIMARY KEY(festival\_nature\_id);

ALTER TABLE festivals

ADD CONSTRAINT pk\_festivals

PRIMARY KEY(location\_id,festival\_nature\_id);

ALTER TABLE staff

ADD CONSTRAINT pk\_staff

PRIMARY KEY(staff\_id);

ALTER TABLE festival\_staff

ADD CONSTRAINT pk\_festival\_staff

PRIMARY KEY(staff\_id,location\_id, festival\_nature\_id);

--FOREIGN KEY

ALTER TABLE festivals

ADD CONSTRAINT fk\_f\_locations

FOREIGN KEY(location\_id)

REFERENCES locations(location\_id);

ALTER TABLE festivals

ADD CONSTRAINT fk\_f\_festival\_nature

FOREIGN KEY(festival\_nature\_id)

REFERENCES festival\_nature(festival\_nature\_id);

ALTER TABLE festival\_staff

ADD CONSTRAINT fk\_f\_staff

FOREIGN KEY(staff\_id)

REFERENCES staff(staff\_id);

ALTER TABLE festival\_staff

ADD CONSTRAINT fk\_f\_festivals

FOREIGN KEY(location\_id,festival\_nature\_id)

REFERENCES festivals(location\_id,festival\_nature\_id);

--UNIQUE KEY

ALTER TABLE festival\_nature

ADD CONSTRAINT uc\_festival\_nature

UNIQUE(festival\_type);

--CHECK CONSTRAINT

ALTER TABLE staff

ADD CONSTRAINT ck\_gender

CHECK(staff\_gender IN ('M','F'));

ALTER TABLE staff

ADD CONSTRAINT ck\_staff\_title

CHECK(staff\_title IN ('MR','MRS','DR','ER','MISS'));

ALTER TABLE staff

ADD CONSTRAINT ck\_staff\_firstname

CHECK (staff\_firstname = UPPER(staff\_firstname));

ALTER TABLE staff

ADD CONSTRAINT ck\_staff\_surname

CHECK (staff\_surname = UPPER(staff\_surname));

ALTER TABLE festival\_nature

ADD CONSTRAINT ck\_festival\_type

CHECK(festival\_type =UPPER(festival\_type));

ALTER TABLE festivals

ADD CONSTRAINT ck\_festival\_name

CHECK(festival\_name =UPPER(festival\_name));

--FOREIGN KEY

ALTER TABLE staff

ADD CONSTRAINT fk\_s\_staff

FOREIGN KEY (leader)

REFERENCES staff(staff\_id);

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Insert Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\insert\_09.txt

\*/

--DESCRIBING THE STRUCTURE OF LOCATIONS TABLE

DESC locations;

--INSERTING VALUES IN LOCATIONS

INSERT INTO locations(location\_id,location\_address)

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','KATHMANDU','PUSPANJALI MARG'));

INSERT INTO locations(location\_id,location\_address)

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','POKHARA','XYZ STREET'));

INSERT INTO locations(location\_id,location\_address)

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','LUMBINI','ABC MARG'));

INSERT INTO locations

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','KATHMANDU','KAPAN'));

INSERT INTO locations

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','DANG','MARG'));

INSERT INTO locations

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','POKHARA','PRITHVI MARG'));

INSERT INTO locations

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','KATHMANDU','PRITHVI MARG'));

INSERT INTO locations

VALUES(seq\_locations.NEXTVAL,address\_type('NEPAL','DANG','VALLEY-TOP MARG'));

--VIEWING THE INSERTED DATA FROM LOCATIONS TABLE

SELECT location\_id,location\_address FROM locations

--DESCRIBING THE STRUCTURE OF FESTIVAL\_NATURE TABLE

DESC festival\_nature;

--INSERTING VALUES IN FESTIVAL\_NATURE

INSERT INTO festival\_nature(festival\_nature\_id,festival\_type)

VALUES(seq\_festival\_nature.NEXTVAL,'EXHIBITION');

INSERT INTO festival\_nature(festival\_nature\_id,festival\_type)

VALUES(seq\_festival\_nature.NEXTVAL,'FOOD FESTIVAL');

INSERT INTO festival\_nature(festival\_nature\_id,festival\_type)

VALUES(seq\_festival\_nature.NEXTVAL,'GAMES');

INSERT INTO festival\_nature

VALUES(seq\_festival\_nature.NEXTVAL,'PARTIES');

INSERT INTO festival\_nature

VALUES(seq\_festival\_nature.NEXTVAL,'MUSICAL');

INSERT INTO festival\_nature

VALUES(seq\_festival\_nature.NEXTVAL,'FILM FESTIVAL');

--VIEWING THE INSERTED DATA FROM FESTIVAL\_NATURE TABLE

SELECT festival\_nature\_id,festival\_type FROM festival\_nature;

--DESCRIBING THE STRUCTURE OF ADDRESSES TABLE

DESC addresses;

--INSERTING VALUES IN ADDRESSES

INSERT INTO addresses

VALUES ('NEPAL', 'KATHMANDU','PUSPANJALI MARG');

INSERT INTO addresses

VALUES ('NEPAL', 'POKHARA','XYZ STREET');

INSERT INTO addresses

VALUES ('NEPAL', 'LUMBINI','ABC MARG');

INSERT INTO addresses

VALUES ('NEPAL', 'KATHMANDU','KAPAN');

INSERT INTO addresses

VALUES ('NEPAL', 'DANG','MARG');

INSERT INTO addresses

VALUES ('INDIA', 'HP','17 AVENUE');

INSERT INTO addresses

VALUES ('NEPAL','POKHARA','PRITHVI MARG');

INSERT INTO addresses

VALUES ('NEPAL','KATHMANDU','PRITHVI MARG');

INSERT INTO addresses

VALUES ('NEPAL','DANG','VALLEY-TOP MARG');

--VIEWING THE INSERTED DATA FROM ADDRESSES

SELECT country,city,street FROM addresses;

--DESCRIBING THE STRUCTURE OF STAFF TABLE

DESC staff;

--INSERTING VALUES IN STAFF

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MRS','PRATIMA','DAHAL','F',contact\_varray\_type(

contact\_type('PRATIMA@GMAIL.COM','01-4000090','0989077888'),

contact\_type('DPRATIMA@GMAIL.COM','01-4890900','9808900990')),

social\_media\_table\_type(

social\_media\_type('FACEBOOK','DPRATIMA',''),

social\_media\_type('TWITTER','PRATIMAD','')),

REF(a) FROM addresses a

WHERE a.street= 'KAPAN';

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MRS','SWASTIKA','ADHIKARI','F',contact\_varray\_type(

contact\_type('SWASTIKA@GMAIL.COM','01-40989090','0989077998'),

contact\_type('ASWASTIKA@GMAIL.COM','01-48090900','980990990')),

social\_media\_table\_type(

social\_media\_type('INSTAGRAM','ASWASTIKA',''),

social\_media\_type('VIBER','SWASTIKAA','')),

REF(a) FROM addresses a

WHERE a.street= 'PUSPANJALI MARG';

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MRS','NITU','DAS','F',contact\_varray\_type(

contact\_type('NITU@GMAIL.COM','01-409000','0989977888'),

contact\_type('DNITU@GMAIL.COM','01-4809090','980890990')),

social\_media\_table\_type(

social\_media\_type('FACEBOOK','DNITU',''),

social\_media\_type('TWITTER','NITUD','')),

REF(a) FROM addresses a

WHERE a.street= 'XYZ STREET';

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MRS','DIKSHYA','GHALE','F',contact\_varray\_type(

contact\_type('DIKKU@GMAIL.COM','01-4900090','0989077988'),

contact\_type('GDIKKU@GMAIL.COM','01-4890900','9808908990')),

social\_media\_table\_type(

social\_media\_type('FACEBOOK','GDIKKU',''),

social\_media\_type('TWITTER','DIKSHYA','')),

REF(a) FROM addresses a

WHERE a.street= 'ABC MARG';

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MRS','SARENA','DAHAL','F',contact\_varray\_type(

contact\_type('SARENA@GMAIL.COM','01-4990090','0989007888'),

contact\_type('DSARENA@GMAIL.COM','01-489000','980000990')),

social\_media\_table\_type(

social\_media\_type('INSTAGRAM','DSARENA',''),

social\_media\_type('TWITTER','SARENAD','')),

REF(a) FROM addresses a

WHERE a.street= 'MARG';

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MR','CHRIS','HEMSWORTH','M',contact\_varray\_type(

contact\_type('HEMS@GMAIL.COM','01-49912','123324234'),

contact\_type('CHEMS@GMAIL.COM','01-489000','980000990')),

social\_media\_table\_type(

social\_media\_type('INSTAGRAM','HEMS.CHRIS',''),

social\_media\_type('TWITTER','H.CHRIS','')),

REF(a) FROM addresses a

WHERE a.street= '17 AVENUE';

INSERT INTO staff(staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_gender,staff\_contact,social\_media\_used,staff\_address)

SELECT seq\_staff.NEXTVAL,'MRS','SCARLETT','EVANS','F',contact\_varray\_type(

contact\_type('SCAR@GMAIL.COM','01-4991892','564789556'),

contact\_type('CHEMS@GMAIL.COM','01-489000','568453548')),

social\_media\_table\_type(

social\_media\_type('INSTAGRAM','SCAR.JOHAN',''),

social\_media\_type('TWITTER','SC.EVANS','')),

REF(a) FROM addresses a

WHERE a.street= 'XYZ STREET';

--FORMATTING COLUMN SIZE

SET LINESIZE 180;

COLUMN staff\_firstname FORMAT A15;

COLUMN staff\_surname FORMAT A15;

COLUMN staff\_contact FORMAT A30;

COLUMN social\_media\_used FORMAT A30;

COLUMN staff\_address FORMAT A30;

--VIEWING THE INSERTED DATA IN STAFF TABLE

SELECT staff\_id,staff\_title,staff\_firstname,staff\_surname,staff\_contact,staff\_address,staff\_gender,staff\_blood\_group,staff\_hire\_date,staff\_salary,staff\_dob FROM staff;

--DESCRIBING THE STRUCTURE OF festivals

DESC festivals;

--INSERTING VALUES IN FESTIVALS

INSERT INTO festivals

VALUES (50001,5000, 'ROBOT SHOWING COMPETITION', '29-OCT-2019');

INSERT INTO festivals

VALUES (50002,5001, 'CONCERT', '29-OCT-2019');

INSERT INTO festivals

VALUES (50003,5002, 'FOOD FESTIVAL', '19-MAR-2019');

INSERT INTO festivals

VALUES (50004,5003, 'VIDEO GAMES', '26-APR-2019');

INSERT INTO festivals

VALUES (50004,5004, 'VIDEO GAMES', '26-APR-2019');

INSERT INTO festivals

VALUES (50004,5000, 'BEER FESTIVAL', '29-JAN-2019');

--VIEWING THE INSERTED DATA IN FESTIVALS TABLE

SELECT festival\_nature\_id,location\_id,festival\_name,occuring\_date FROM festivals;

--DESCRIBING THE STRUCTURE OF FESTIVAL\_STAFF

DESC festival\_staff;

--INSERTING VALUES IN FESTIVAL\_STAFF

INSERT INTO festival\_staff(location\_id, festival\_nature\_id, staff\_id)

VALUES (5000, 50001, 1000 );

INSERT INTO festival\_staff(location\_id, festival\_nature\_id, staff\_id)

VALUES (5001, 50002, 1002 );

INSERT INTO festival\_staff(location\_id, festival\_nature\_id, staff\_id)

VALUES (5002, 50003, 1004 );

INSERT INTO festival\_staff(location\_id, festival\_nature\_id, staff\_id)

VALUES (5003, 50004, 1006 );

INSERT INTO festival\_staff(location\_id, festival\_nature\_id, staff\_id)

VALUES (5004, 50004, 1008 );

--VIEWING THE INSERTED DATA IN FESTIVAL\_STAFF

SELECT location\_id,festival\_nature\_id,staff\_id FROM festival\_staff;

--UPDATING THE VALUES IN STAFF

UPDATE staff

SET staff\_blood\_group = 'A+',staff\_hire\_date = '18-FEB-2015',staff\_salary = 20000,staff\_dob = '19-MAR-1995'

WHERE staff\_id = 1000;

UPDATE staff

SET staff\_blood\_group = 'AB+',staff\_hire\_date = '20-APR-2014',staff\_salary = 30000,staff\_dob = '20-APR-1994'

WHERE staff\_id = 1002;

UPDATE staff

SET staff\_blood\_group = 'B+',staff\_hire\_date = '18-FEB-2016',staff\_salary = 50000,staff\_dob = '1-NOV-1993'

WHERE staff\_id = 1004;

UPDATE staff

SET staff\_blood\_group = 'AB-',staff\_hire\_date = '23-NOV-2017',staff\_salary = 40000,staff\_dob = '19-DEC-1992'

WHERE staff\_id = 1006;

UPDATE staff

SET staff\_blood\_group = 'B-',staff\_hire\_date = '29-DEC-2015',staff\_salary = 60000,staff\_dob = '12-MAR-1991'

WHERE staff\_id = 1008;

UPDATE staff

SET staff\_salary = 50000

WHERE staff\_id = 1010;

UPDATE staff

SET staff\_salary = 60000

WHERE staff\_id = 1012;

UPDATE staff

SET leader= 1002;

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Query Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\query\_09.txt

\*/

--FORMATTING COLUMN SIZE

COLUMN festival\_name FORMAT A25;

COLUMN staff\_firstname FORMAT A15;

COLUMN staff\_surname FORMAT A15;

--VIEWING SEQUENCE

SELECT sequence\_name

FROM user\_sequences;

--VIEWING TABLE

SELECT \* FROM TAB;

--VIEWING CONSTRAINST

SELECT constraint\_name

FROM user\_constraints;

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'PK%';

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'FK%';

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'UC%';

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'CK%';

--VIEWING THE DATA FROM THE TABLES

/\*

SELECT \* FROM staff;

SELECT \* FROM festival\_nature;

SELECT \* FROM locations;

SELECT \* FROM festivals;

SELECT \* FROM festival\_staff;

SELECT \* FROM addresses;

\*/

--DISPLAY DETAILS OF STAFF WHO ARE HIRED BEFORE THEIR LEADER

SELECT s.staff\_id, s.staff\_firstname, staff\_surname FROM staff s

WHERE exists(

SELECT staff\_id FROM staff l

WHERE leader IS NOT NULL

AND s.leader = l.leader

AND s.staff\_hire\_date>l.staff\_hire\_date);

--DISPLAYING THE SELECTED FIELDS FROM TABLES

--DESCRIBING STAFF TABLE

DESC staff;

--QUERYING USING SIMPLE METHOD

--QUERYING THE STAFF'S NAME AND SALARY WHO SURNAME IS NOT EQUAL TO SHARMA

SELECT staff\_firstname,staff\_surname,to\_char(staff\_salary,'999999D99') Salary

FROM staff

WHERE staff\_surname != 'SHARMA';

/\*

STAFF\_FIRSTNAME STAFF\_SURNAME SALARY

--------------- --------------- ----------

PRATIMA DAHAL 20002.00

SWASTIKA ADHIKARI 30003.00

NITU DAS 50005.00

DIKSHYA GHALE 40004.00

SARENA DAHAL 60006.00

CHRIS HEMSWORTH 50005.00

\*/

--DESCRIBING FESTIVALS TABLE

DESC festivals;

--QUERYING THE FESTIVALS TABLE USING NOT LIKE AND

SELECT festival\_name FROM festivals

WHERE location\_id <> 5001

AND occuring\_date NOT LIKE '%2018';

/\*

FESTIVAL\_NAME

-----------------------------------

ROBOT SHOWING COMPETITION

FOOD FESTIVAL

VIDEO GAMES

VIDEO GAMES

BEER FESTIVAL

\*/

--DESCRIBING STAFF TABLE

DESC staff;

--QUERYING FESTIVAL NATURE ID WHERE FESTIVAL TYPE IS NULL

SELECT fn.festival\_nature\_id

FROM festival\_nature fn

WHERE fn.festival\_type IS NULL;

--no rows selected

COLUMN location\_address FORMAT A60

--QUERYING THE LOCATIONS TABLE USING POOR METHOD

SELECT location\_id,location\_address

FROM locations;

/\*

LOCATION\_ID LOCATION\_ADDRESS(COUNTRY, CITY, STREET)

----------- ------------------------------------------------------------

5000 ADDRESS\_TYPE('NEPAL', 'KATHMANDU', 'PUSPANJALI MARG')

5001 ADDRESS\_TYPE('NEPAL', 'POKHARA', 'XYZ STREET')

5002 ADDRESS\_TYPE('NEPAL', 'LUMBINI', 'ABC MARG')

5003 ADDRESS\_TYPE('NEPAL', 'KATHMANDU', 'KAPAN')

5004 ADDRESS\_TYPE('NEPAL', 'DANG', 'MARG')

5005 ADDRESS\_TYPE('NEPAL', 'POKHARA', 'PRITHVI MARG')

5006 ADDRESS\_TYPE('NEPAL', 'KATHMANDU', 'PRITHVI MARG')

5007 ADDRESS\_TYPE('NEPAL', 'DANG', 'VALLEY-TOP MARG')

8 rows selected.

\*/

--QUERYING the locations table using better method to show site\_id and state\_ref

SELECT l.location\_address.city,l.location\_id

FROM locations l;

COLUMN staff\_address FORMAT A30

--QUERYING THE LOCATIONS TABLE USING POOR METHOD

SELECT staff\_id,staff\_firstname,staff\_address

FROM staff;

/\*

STAFF\_ID STAFF\_FIRSTNAME

---------- --------------------

STAFF\_ADDRESS

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------

1008 SARENA

0000220208999BBA2C717B494F9E9090D079476740DDDCF916BCDD4A179BC140815AE79DA0

1010 CHRIS

00002202080BFADDC91D124E2989AEA58FBDB0F072DDDCF916BCDD4A179BC140815AE79DA0

\*/

COLUMN staff\_address FORMAT A40

--QUERYING THE STAFF TABLE USING BETTER METHOD

SELECT staff\_id,staff\_firstname,deref(staff\_address) staff\_address

FROM staff;

/\*

STAFF\_ID STAFF\_FIRSTNAME

---------- --------------------

DEREF(STAFF\_ADDRESS)(COUNTRY, CITY, STREET)

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------

1008 SARENA

ADDRESS\_TYPE('NEPAL', 'DANG', 'MARG')

1010 CHRIS

ADDRESS\_TYPE('INDIA', 'HP', '17 AVENUE')

\*/

--QUERYING THE STAFF TABLE USING BETTER METHOD WITH MINUS

SELECT staff\_firstname,staff\_surname,deref(staff\_address).country

FROM staff

MINUS

SELECT staff\_firstname,staff\_surname,deref(staff\_address).country

FROM staff

WHERE deref(staff\_address).country != 'NEPAL';

/\*

STAFF\_FIRSTNAME STAFF\_SURNAME DEREF(STAFF\_ADDRESS).COUN

-------------------- -------------------- -------------------------

DIKSHYA GHALE NEPAL

NITU DAS NEPAL

PRATIMA DAHAL NEPAL

SARENA DAHAL NEPAL

SWASTIKA ADHIKARI NEPAL

\*/

COLUMN street FORMAT A20

COLUMN city FORMAT A20

--QUERYING THE STAFF TABLE USING BEST METHOD

SELECT DISTINCT(s.staff\_id),s.staff\_firstname,s.staff\_address.country country,

s.staff\_address.city city,s.staff\_address.street street,soc.social\_media\_username

FROM staff s,TABLE(s.social\_media\_used) soc;

/\*

STAFF\_ID STAFF\_FIRSTNAME COUNTRY CITY STREET SOCIAL\_MEDIA\_USERNAME

---------- -------------------- ------------------------- -------------------- -------------------- --------------------------------------------------

1010 CHRIS INDIA HP 17 AVENUE H.CHRIS

1004 NITU NEPAL POKHARA XYZ STREET NITUD

1006 DIKSHYA NEPAL LUMBINI ABC MARG DIKSHYA

1000 PRATIMA NEPAL KATHMANDU KAPAN DPRATIMA

1008 SARENA NEPAL DANG MARG DSARENA

1010 CHRIS INDIA HP 17 AVENUE HEMS.CHRIS

1012 SCARLETT NEPAL POKHARA XYZ STREET SC.EVANS

1012 SCARLETT NEPAL POKHARA XYZ STREET SCAR.JOHAN

1002 SWASTIKA NEPAL KATHMANDU PUSPANJALI MARG ASWASTIKA

1004 NITU NEPAL POKHARA XYZ STREET DNITU

1002 SWASTIKA NEPAL KATHMANDU PUSPANJALI MARG SWASTIKAA

STAFF\_ID STAFF\_FIRSTNAME COUNTRY CITY STREET SOCIAL\_MEDIA\_USERNAME

---------- -------------------- ------------------------- -------------------- -------------------- --------------------------------------------------

1006 DIKSHYA NEPAL LUMBINI ABC MARG GDIKKU

1000 PRATIMA NEPAL KATHMANDU KAPAN PRATIMAD

1008 SARENA NEPAL DANG MARG SARENAD

\*/

--DESCRIBING FESTIVALS TABLE

DESC festivals;

--QUERYING THE FESTIVALS TABLE USING INNER JOIN AND ORDER BY

SELECT f.festival\_name, f.occuring\_date, fn.festival\_type

FROM festivals f

INNER JOIN festival\_nature fn ON

f.festival\_nature\_id = fn.festival\_nature\_id

ORDER BY f.festival\_name;

/\*

FESTIVAL\_NAME OCCURING\_DATE FESTIVAL\_TYPE

----------------------------------- -------------------- --------------------------------------------------

BEER FESTIVAL 29-JAN-19 MUSICAL

CONCERT 29-OCT-19 MEALS,EATING AND DRINKING

FOOD FESTIVAL 19-MAR-19 GAMES

ROBOT SHOWING COMPETITION 29-OCT-19 EXHIBITION

VIDEO GAMES 26-APR-19 PARTIES

VIDEO GAMES 26-APR-19 PARTIES

\*/

--DESCRIBING FESTIVALS TABLE

DESC festivals;

COLUMN festival\_name FORMAT A25;

COLUMN festival\_type FORMAT A25;

--QUERYING THE FESTIVALS TABLE USING INNER JOIN,JOIN AND ORDER BY

SELECT f.festival\_name festival\_name, f.occuring\_date, fn.festival\_type festival\_type, l.location\_address.country country,l.location\_address.city city, l.location\_address.street

street

FROM festivals f

INNER JOIN festival\_nature fn ON

f.festival\_nature\_id = fn.festival\_nature\_id

JOIN locations l ON

f.location\_id = l.location\_id

ORDER BY f.festival\_name;

/\*

FESTIVAL\_NAME OCCURING\_DATE FESTIVAL\_TYPE

COUNTRY CITY STREET

----------------------------------- -------------------- -------------------------------------------------- -------------------- -------------------- --------------------

BEER FESTIVAL 29-JAN-19 MUSICAL

NEPAL KATHMANDU PUSPANJALI MARG

CONCERT 29-OCT-19 MEALS,EATING AND DRINKING

NEPAL POKHARA XYZ STREET

FOOD FESTIVAL 19-MAR-19 GAMES

NEPAL LUMBINI ABC MARG

ROBOT SHOWING COMPETITION 29-OCT-19 EXHIBITION

NEPAL KATHMANDU PUSPANJALI MARG

VIDEO GAMES 26-APR-19 PARTIES

NEPAL DANG MARG

VIDEO GAMES 26-APR-19 PARTIES

NEPAL KATHMANDU KAPAN

\*/

--DESCRIBING LOCATIONS TABLE

DESC locations;

--QUERYING ALL FESTIVALS CELEBRATED AT KATHMANDU CITY

SELECT f.festival\_name

FROM festivals f

WHERE f.location\_id IN (

SELECT l.location\_id

FROM locations l

WHERE l.location\_address.city = 'KATHMANDU');

/\*

FESTIVAL\_NAME

-----------------------------------

ROBOT SHOWING COMPETITION

BEER FESTIVAL

VIDEO GAMES

\*/

--QUERYING ALL STAFF ORGANIZING FESTIVALS USING FULL OUTER JOIN AND ON

SELECT f.festival\_name,s.staff\_firstname,s.staff\_surname

FROM festivals f

FULL OUTER JOIN festival\_staff ff

ON f.festival\_nature\_id = ff.festival\_nature\_id

FULL OUTER JOIN staff s

ON ff.staff\_id = s.staff\_id;

/\*

FESTIVAL\_NAME STAFF\_FIRSTNAME STAFF\_SURNAME

----------------------------------- -------------------- --------------------

CHRIS HEMSWORTH

CHRIS HEMSWORTH

ROBOT SHOWING COMPETITION PRATIMA DAHAL

CONCERT SWASTIKA ADHIKARI

FOOD FESTIVAL NITU DAS

VIDEO GAMES DIKSHYA GHALE

VIDEO GAMES DIKSHYA GHALE

VIDEO GAMES SARENA DAHAL

VIDEO GAMES SARENA DAHAL

CHRIS HEMSWORTH

BEER FESTIVAL

\*/

--QUERYING ALL FESTIVAL TYPE HELD ON COUNTRY NEPAL

SELECT ff.festival\_type

FROM festival\_nature ff

WHERE ff.festival\_nature\_id IN (

SELECT f.festival\_nature\_id

FROM festivals f

WHERE f.location\_id IN(

SELECT l.location\_id

FROM locations l

WHERE l.location\_address.country ='NEPAL'));

/\*

FESTIVAL\_TYPE

--------------------------------------------------

EXHIBITION

MEALS,EATING AND DRINKING

GAMES

PARTIES

MUSICAL

\*/

--DESCRIBING THE STAFF TABLE

DESC staff;

--QUERYING THE ADDRESS WHERE STAFF IS NOT AVAILABLE

SELECT \* FROM addresses a

WHERE REF(a) NOT IN(

SELECT s.staff\_address

FROM staff s

);

/\*

no rows selected

\*/

SET LINESIZE 180;

--QUERYING ALL

SELECT ff.festival\_type,f.festival\_name

FROM festival\_nature ff

FULL OUTER JOIN festivals f

ON ff.festival\_nature\_id = f.festival\_nature\_id

WHERE ff.festival\_nature\_id IN (

SELECT f.festival\_nature\_id

FROM festivals f

WHERE f.location\_id IN(

SELECT l.location\_id

FROM locations l

WHERE l.location\_address.country IN(

SELECT s.staff\_address.country Country

FROM staff s

WHERE s.staff\_address.country = 'NEPAL'

AND s.staff\_address.city ='KATHMANDU'

UNION

SELECT l.location\_address.country Country

FROM locations l

WHERE l.location\_address.country = 'NEPAL'

)

)

);

/\*

FESTIVAL\_TYPE FESTIVAL\_NAME

-------------------------------------------------- --------------------------------------------------

GAMES FOOD FESTIVAL

EXHIBITION ROBOT SHOWING COMPETITION

MUSICAL BEER FESTIVAL

MEALS,EATING AND DRINKING CONCERT

PARTIES VIDEO GAMES

PARTIES VIDEO GAMES

6 rows selected.

\*/

--VIEWING STAFF\_FIRSTNAME,STAFF\_ID,LOCATION USING LEFT OUTER AND INNER JOIN

SELECT s.staff\_id,s.staff\_firstname,l.location\_address.country Country

FROM staff s

LEFT OUTER JOIN(

festival\_staff ff INNER JOIN festivals f

ON ff.festival\_nature\_id = f.festival\_nature\_id

AND ff.location\_id = f.location\_id

JOIN locations l

ON l.location\_id = f.location\_id

)

ON s.staff\_id = ff.staff\_id;

/\*

STAFF\_ID STAFF\_FIRSTNAME COUNTRY

---------- ------------------------------ --------------------

1000 PRATIMA NEPAL

1002 SWASTIKA NEPAL

1004 NITU NEPAL

1006 DIKSHYA NEPAL

1008 SARENA NEPAL

1010 CHRIS

1040 CHRIS

1042 CHRIS

8 rows selected.

\*/

--VIEWING FESTIVAL NAME AND OCCURING DATE OF NEPAL ONLY USING EXISTS KEYWORD

SELECT f.festival\_name, f.occuring\_date

FROM festivals f

WHERE EXISTS(

SELECT location\_id

FROM locations l

WHERE l.location\_address.country = 'NEPAL'

);

/\*

FESTIVAL\_NAME OCCURING\_DATE

-------------------------------------------------- --------------------

ROBOT SHOWING COMPETITION 29-OCT-19

CONCERT 29-OCT-19

FOOD FESTIVAL 19-MAR-19

VIDEO GAMES 26-APR-19

VIDEO GAMES 26-APR-19

BEER FESTIVAL 29-JAN-19

6 rows selected.

\*/

----------------------------------------------------

-- USING WHERE CONDITION AND TABLE ALIAS

SELECT s.staff\_firstname,s.staff\_surname

FROM staff s

WHERE s.staff\_address.city IN ('KATHMANDU' , 'POKHARA');

/\*

STAFF\_FIRSTNAME STAFF\_SURNAME

------------------------------ ------------------------------

PRATIMA DAHAL

SWASTIKA ADHIKARI

NITU DAS

\*/

--not between like null desc asc join with order function like count,sum etc

--SHOWS STAFFS TABLE WITH THE GIVEN ATTRIBUTES IN THE GIVEN CONDITION.

SELECT staff\_id,staff\_firstname,staff\_surname,staff\_hire\_date

FROM staff

WHERE staff\_hire\_date > '1-JAN-2012'

AND staff\_hire\_date < SYSDATE ;

/\*

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME STAFF\_HIR

---------- ------------------------------ ------------------------------ ---------

1040 CHRIS HEMSWORTH 15-APR-19

1042 CHRIS HEMSWORTH 15-APR-19

1000 PRATIMA DAHAL 18-FEB-15

1002 SWASTIKA ADHIKARI 20-APR-14

1004 NITU DAS 18-FEB-16

1006 DIKSHYA GHALE 23-NOV-17

1008 SARENA DAHAL 29-DEC-15

1010 CHRIS HEMSWORTH 08-APR-19

8 rows selected.

\*/

--SHOW ALL DETAILS OF FESTIVALS AND STAFF WHO IS GREATER THAN 20 YRS

SELECT f.festival\_name,f.occuring\_date,s.staff\_firstname,s.staff\_surname

FROM festivals f INNER JOIN festival\_staff ff

ON f.festival\_nature\_id = ff.festival\_nature\_id

INNER JOIN staff s

ON s.staff\_id = ff.staff\_id

WHERE (SYSDATE-s.staff\_dob) > 20;

/\*

FESTIVAL\_NAME OCCURING\_DATE STAFF\_FIRSTNAME STAFF\_SURNAME

-------------------------------------------------- -------------------- ------------------------------ ------------------------------

ROBOT SHOWING COMPETITION 29-OCT-19 PRATIMA

DAHAL

CONCERT 29-OCT-19 SWASTIKA

ADHIKARI

FOOD FESTIVAL 19-MAR-19 NITU

DAS

VIDEO GAMES 26-APR-19 DIKSHYA

GHALE

VIDEO GAMES 26-APR-19 SARENA

DAHAL

VIDEO GAMES 26-APR-19 DIKSHYA

GHALE

VIDEO GAMES 26-APR-19 SARENA

DAHAL

7 rows selected.

\*/

----SHOWING THE DETAILS OF FESTIVAL NAME,OCCURING DATE AND FESTIVAL TYPE

SELECT f.festival\_name, f.occuring\_date, fn.festival\_type

FROM festivals f

INNER JOIN festival\_nature fn ON

f.festival\_nature\_id = fn.festival\_nature\_id

ORDER BY f.festival\_nature\_id DESC;

/\*

FESTIVAL\_NAME OCCURING\_DATE FESTIVAL\_TYPE

-------------------------------------------------- -------------------- --------------------------------------------------

BEER FESTIVAL 29-JAN-19 MUSICAL

VIDEO GAMES 26-APR-19 PARTIES

VIDEO GAMES 26-APR-19 PARTIES

FOOD FESTIVAL 19-MAR-19 GAMES

CONCERT 29-OCT-19 MEALS,EATING AND DRINKING

ROBOT SHOWING COMPETITION 29-OCT-19 EXHIBITION

6 rows selected.

\*/

--QUERYING FESTIVAL DETAILS FROM FESTIVAL TABLE USING INNER JOIN AND ORDER BY

COLUMN festival\_name FORMAT A30;

COLUMN festival\_type FORMAT A30;

COLUMN street FORMAT A30

COLUMN city FORMAT A30

SELECT f.festival\_name festival\_name, f.occuring\_date occurring\_date, fn.festival\_type, l.location\_address.country country,l.location\_address.city city, l.location\_address.street

street

FROM festivals f

INNER JOIN festival\_nature fn ON

f.festival\_nature\_id = fn.festival\_nature\_id

JOIN locations l ON

f.location\_id = l.location\_id

ORDER BY f.occuring\_date ASC;

/\*FESTIVAL\_NAME OCCURING\_DATE FESTIVAL\_TYPE COUNTRY CITY STREET

-------------------------------------------------- -------------------- -------------------------------------------------- -------------------- --------------------

--------------------

BEER FESTIVAL 29-JAN-19 MUSICAL

NEPAL KATHMANDU PUSPANJALI MARG

FOOD FESTIVAL 19-MAR-19 GAMES

NEPAL LUMBINI ABC MARG

VIDEO GAMES 26-APR-19 PARTIES

NEPAL KATHMANDU KAPAN

VIDEO GAMES 26-APR-19 PARTIES

NEPAL DANG MARG

ROBOT SHOWING COMPETITION 29-OCT-19 EXHIBITION

NEPAL KATHMANDU PUSPANJALI MARG

CONCERT 29-OCT-19 MEALS,EATING AND DRINKING NEPAL POKHARA XYZ STREET

6 rows selected.

\*/

--SELECT ALL FESTIVALS CELEBRATED AT KATHMANDU CITY

SELECT f.festival\_name

FROM festivals f

WHERE f.location\_id IN (

SELECT l.location\_id

FROM locations l

WHERE l.location\_address.city = 'KATHMANDU');

/\*

OUTPUT

FESTIVAL\_NAME

--------------------------------------------------

ROBOT SHOWING COMPETITION

BEER FESTIVAL

VIDEO GAMES

\*/

--SELECT ALL STAFF ORGANIZING FESTIVALS

SELECT f.festival\_name,s.staff\_firstname,s.staff\_surname

FROM festivals f

FULL OUTER JOIN festival\_staff ff

ON f.festival\_nature\_id = ff.festival\_nature\_id

FULL OUTER JOIN staff s

ON ff.staff\_id = s.staff\_id;

/\*

output

FESTIVAL\_NAME STAFF\_FIRSTNAME STAFF\_SURNAME

-------------------------------------------------- ------------------------------ ------------------------------

CHRIS HEMSWORTH

CHRIS HEMSWORTH

ROBOT SHOWING COMPETITION PRATIMA DAHAL

CONCERT SWASTIKA ADHIKARI

FOOD FESTIVAL NITU DAS

VIDEO GAMES DIKSHYA GHALE

VIDEO GAMES DIKSHYA GHALE

VIDEO GAMES SARENA DAHAL

VIDEO GAMES SARENA DAHAL

CHRIS HEMSWORTH

\*/

--SELECT ALL FESTIVAL TYPE HELD ON CONTRY NEPAL

SELECT ff.festival\_type

FROM festival\_nature ff

WHERE ff.festival\_nature\_id IN (

SELECT f.festival\_nature\_id

FROM festivals f

WHERE f.location\_id IN(

SELECT l.location\_id

FROM locations l

WHERE l.location\_address.country ='NEPAL'));

/\*

OUTPUT

FESTIVAL\_TYPE

--------------------------------------------------

EXHIBITION

MEALS,EATING AND DRINKING

GAMES

PARTIES

MUSICAL

\*/

--ADDRESS WHERE STAFF IS NOT AVAILABLE

SELECT \* FROM addresses a

WHERE REF(a) NOT IN(

SELECT s.staff\_address

FROM staff s

);

/\*

NO ROWS SELECTED

\*/

--SELECT ALL FESTIVAL TYPE AND FESTIVAL NAME USING WHERE CLAUSE AND UNIION, IN

SELECT ff.festival\_type,f.festival\_name

FROM festival\_nature ff

FULL OUTER JOIN festivals f

ON ff.festival\_nature\_id = f.festival\_nature\_id

WHERE ff.festival\_nature\_id IN (

SELECT f.festival\_nature\_id

FROM festivals f

WHERE f.location\_id IN(

SELECT l.location\_id

FROM locations l

WHERE l.location\_address.country IN(

SELECT s.staff\_address.country Country

FROM staff s

WHERE s.staff\_address.country = 'NEPAL'

AND s.staff\_address.city ='KATHMANDU'

UNION

SELECT l.location\_address.country Country

FROM locations l

WHERE l.location\_address.country = 'NEPAL'

)

)

);

/\*

OUTPUT

FESTIVAL\_TYPE FESTIVAL\_NAME

-------------------------------------------------- --------------------------------------------------

GAMES FOOD FESTIVAL

EXHIBITION ROBOT SHOWING COMPETITION

MUSICAL BEER FESTIVAL

MEALS,EATING AND DRINKING CONCERT

PARTIES VIDEO GAMES

PARTIES VIDEO GAMES

\*/

----SELECT STAFF NAME AND LOCATION FROM LOCATION TABLE

SELECT s.staff\_id,s.staff\_firstname,l.location\_address.country Country

FROM staff s

LEFT OUTER JOIN(

festival\_staff ff INNER JOIN festivals f

ON ff.festival\_nature\_id = f.festival\_nature\_id

AND ff.location\_id = f.location\_id

JOIN locations l

ON l.location\_id = f.location\_id

)

ON s.staff\_id = ff.staff\_id;

/\*

OUTPUT

STAFF\_ID STAFF\_FIRSTNAME COUNTRY

---------- ------------------------------ -------------------------

1000 PRATIMA NEPAL

1002 SWASTIKA NEPAL

1004 NITU NEPAL

1006 DIKSHYA NEPAL

1008 SARENA NEPAL

1010 CHRIS

1040 CHRIS

1042 CHRIS

\*/

--SELECT FESTIVAL NAME AND OCCURING DATE IN GIVEN LOCATIONS

SELECT f.festival\_name, f.occuring\_date

FROM festivals f

WHERE EXISTS(

SELECT location\_id

FROM locations l

WHERE l.location\_address.country = 'NEPAL'

);

/\*

OUTPUT

FESTIVAL\_NAME OCCURING\_

-------------------------------------------------- ---------

ROBOT SHOWING COMPETITION 29-OCT-19

CONCERT 29-OCT-19

FOOD FESTIVAL 19-MAR-19

VIDEO GAMES 26-APR-19

VIDEO GAMES 26-APR-19

BEER FESTIVAL 29-JAN-19

\*/

--QUERYING STAFF ID, FIRSTNAME,AND COUNTRY OF STAFF USING INTERSECT

SELECT l.location\_address.city CITY

FROM locations l

INNER JOIN festivals f

ON l.location\_id = f.location\_id

FULL OUTER JOIN festival\_staff ff

ON f.festival\_nature\_id = ff.festival\_nature\_id

INTERSECT

SELECT s.staff\_address.city

FROM staff s

LEFT OUTER JOIN festival\_staff fe

ON s.staff\_id = fe.staff\_id;

--(TechOnTheNet,2019)

/\*

OUTPUT

CITY

--------------------------------------------------

DANG

KATHMANDU

LUMBINI

POKHARA

\*/

-- QUERYING STAFF ID, FIRSTNAME,AND COUNTRY OF STAFF USING RIGHT JOIN, INNER JOIN

SELECT s.staff\_id,s.staff\_firstname,s.staff\_address.country Country,fn.festival\_type

FROM festival\_nature fn

RIGHT OUTER JOIN(

festivals f INNER JOIN festival\_staff ff

ON f.festival\_nature\_id = ff.festival\_nature\_id

JOIN staff s

ON s.staff\_id = ff.staff\_id

)

ON fn.festival\_nature\_id = f.festival\_nature\_id;

/\*

OUTPUT

CITY

--------------------------------------------------

DANG

KATHMANDU

LUMBINI

POKHARA

\*/

--QUERYING TO COUNT THE FESTIVAL NATURE ID AND FESTIVAL NAME

SELECT COUNT(festival\_nature\_id) Serial,festival\_name

FROM festivals

GROUP BY festival\_name;

/\*

OUTPUT

COUNT(FESTIVAL\_NATURE\_ID) FESTIVAL\_NAME

------------------------- --------------------------------------------------

1 BEER FESTIVAL

1 CONCERT

2 VIDEO GAMES

1 ROBOT SHOWING COMPETITION

1 FOOD FESTIVAL

\*/

--UPDATE STAFF TABLE WITH staff\_salary+(staff\_salary\*0.0001s

UPDATE staff

SET staff\_salary = staff\_salary+(staff\_salary\*0.0001);

/\*

OUTPUT

one line Updated by DB on 19-APR-19

\*/

--TO SHOW THE SALARY OF THE STAFF

SELECT staff\_salary , CEIL(staff\_salary) "CEIL" ,

FLOOR(staff\_salary)"FLOOR",

ROUND(staff\_salary)"ROUND",

TRUNC(staff\_salary)"TRUNCATE"

FROM staff;

/\*

OUTPUT

STAFF\_SALARY CEIL FLOOR ROUND TRUNCATE

------------ ---------- ---------- ---------- ----------

39964 39964 39964 39964 39964

59945.99 59946 59945 59946 59945

99909.99 99910 99909 99910 99909

79927.99 79928 79927 79928 79927

119891.99 119892 119891 119892 119891

\*/

-- QUERYING TO DISPLAY THE MAXMIMUM SALARY OF THE STAFF

SELECT MAX(to\_char(staff\_salary,'999999D99')) "Max Salary",CEIL(MAX(to\_char(staff\_salary,'999999D99'))) "CEIL" ,

FLOOR(MAX(to\_char(staff\_salary,'999999D99')))"FLOOR",

ROUND(MAX(to\_char(staff\_salary,'999999D99')))"ROUND",

TRUNC(MAX(to\_char(staff\_salary,'999999D99')))"TRUNCATE"

FROM staff;

/\*

OUTPUT

MAX(STAFF\_SALARY) CEIL FLOOR ROUND TRUNCATE

----------------- ---------- ---------- ---------- ----------

119891.99 119892 119891 119892 119891

\*/

--QUERYING TO DISPLAY THE MINIMUM SALARY OF THE STAFF

SELECT MIN(to\_char(staff\_salary,'999999D99')),CEIL(MIN(to\_char(staff\_salary,'999999D99'))) "CEIL" ,

FLOOR(MIN(to\_char(staff\_salary,'999999D99')))"FLOOR",

ROUND(MIN(to\_char(staff\_salary,'999999D99')))"ROUND",

TRUNC(MIN(to\_char(staff\_salary,'999999D99')))"TRUNCATE"

FROM staff;

/\*

OUTPUT

MIN(STAFF\_SALARY) CEIL FLOOR ROUND TRUNCATE

----------------- ---------- ---------- ---------- ----------

39964 39964 39964 39964 39964

\*/

--

--QUERY TO DISPLAY THE CEIL ROUND TRUNCATE AND FLOOR OF THE SALARY

SELECT \* FROM(

SELECT staff\_firstname , ROUND(staff\_salary) "max and round" , CEIL(staff\_salary) "CEIL" ,FLOOR(staff\_salary)"FLOOR",TRUNC(staff\_salary)"Trunct"

FROM staff

ORDER BY staff\_salary ASC)

WHERE rownum<=1;

/\*

OUTPUT

STAFF\_FIRSTNAME max and round CEIL FLOOR Trunct

------------------------------ ------------- ---------- ---------- ----------

PRATIMA 39964 39964 39964 39964

\*/

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Procedure Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\procedure\_09.txt

\*/

--ENABLING SERVEROUTPUT TO DISPLAY SERVER MESSAGES

SET SERVEROUTPUT ON;

--DESCRIBING THE TABLE STAFF

DESC TABLE staff;

--DESCRIBING THE TABLE STAFF

DESC staff;

--VIEWING THE DATA OF STAFF

SELECT \* FROM staff;

-----1------

--UNNAMED BLOCK

--USE OF LOOP,WHILE,FOR

-----GET FIRSTNAME SURNAME ND TITLE

DECLARE

vn\_length NUMBER(2);

vn\_counter NUMBER(3) := 1;

vc\_title VARCHAR2(20);

vc\_firstname VARCHAR2(20);

vc\_surname VARCHAR2(20);

BEGIN

SELECT staff\_title,staff\_firstname,staff\_surname INTO vc\_title,vc\_firstname,vc\_surname FROM staff WHERE staff\_id = 1000;

DBMS\_OUTPUT.PUT\_LINE('--Title--');

vn\_length := LENGTH(vc\_title);

--START THE LOOP

LOOP

--LOOP TERMINATION CONDITION

EXIT WHEN vn\_counter>vn\_length;

DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_title, vn\_counter, 1));

vn\_counter := vn\_counter + 1;

--END OF LOOP

END LOOP;

vn\_counter :=1;

DBMS\_OUTPUT.PUT\_LINE('--Firstname--');

vn\_length := LENGTH(vc\_firstname);

--WHILE LOOP TERMINATION CONDITION

WHILE vn\_counter<=vn\_length LOOP

DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_firstname, vn\_counter, 1));

vn\_counter := vn\_counter + 1;

--END OF LOOP

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--Surname--');

vn\_length := LENGTH(vc\_surname);

--FOR LOOP TERMINATION CONDITION

FOR vn\_counter IN 1 .. vn\_length LOOP

DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_surname, vn\_counter, 1));

--END OF LOOP

END LOOP;

--END OF PROGRAMME

END;

/

-------------2------------------

--RETURNS THE NAME OF STAFF WITH PARTICULAR ID

DECLARE

c\_id staff.staff\_id%type := 1000;

c\_name staff.staff\_firstname%type;

BEGIN

SELECT staff\_firstname INTO c\_name

FROM staff

WHERE staff\_id = c\_id;

DBMS\_OUTPUT.PUT\_LINE ('Name: '|| c\_name);

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such staff!');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

/

----------------------------------\*\*\*\*\*\*\*\*\*3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------------------------

--CREATE VARRAY TEAM\_VARRAY\_TYPE FOR TABLE TEAM

CREATE OR REPLACE TYPE team\_varray\_type AS VARRAY(50) OF NUMBER(11);

/

--CREATING TABLE USING PROCEDURE

CREATE OR REPLACE PROCEDURE proc\_create\_team IS

vc\_command VARCHAR2(2000);

BEGIN

vc\_command:='CREATE TABLE team (team\_id NUMBER(7),team\_name VARCHAR2(100),members team\_varray\_type,staff\_id NUMBER(11))';

EXECUTE IMMEDIATE vc\_command;

END;

/

--EXECUTE THE PROCEDURE

EXEC proc\_create\_team;

-------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---------------------------------

--CREATING PRIMARY KEY

CREATE OR REPLACE PROCEDURE proc\_primary\_team IS

vc\_command VARCHAR2(2000);

BEGIN

vc\_command:='ALTER TABLE team

ADD CONSTRAINT pk\_team

PRIMARY KEY(team\_id) ';

EXECUTE IMMEDIATE vc\_command;

END;

/

--EXECUTE THE PROCEDURE

EXEC proc\_primary\_team;

------------------------------------------------------------------------

--CREATING FOREIGN KEY

CREATE OR REPLACE PROCEDURE proc\_foreign\_team IS

vc\_command VARCHAR2(2000);

BEGIN

vc\_command:='ALTER TABLE team

ADD CONSTRAINT fk\_t\_staff

FOREIGN KEY(staff\_id)

REFERENCES staff(staff\_id)';

EXECUTE IMMEDIATE vc\_command;

END;

/

SHOW ERRORS;

--NO ERRORS

--TESTING

--SELECT ALL PRIMARY KEY

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'FK%';

--4 rows returned

--EXECUTE THE PROCEDURE

EXEC proc\_foreign\_team;

--SELECT ALL PRIMARY KEY

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'FK%';

--5 rows returned

------------------------------\*\*\*\*\*\*\*\*\*\*4\*\*\*\*\*\*\*\*------------------------

--CREATING A PROCEDURE NAMED proc\_staff

CREATE OR REPLACE PROCEDURE proc\_staff IS

vc\_staff\_firstname staff.staff\_firstname%TYPE :='AMRIT';

vc\_staff\_surname staff.staff\_surname%TYPE := 'THAPA';

BEGIN

INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname)

VALUES(seq\_staff.NEXTVAL,vc\_staff\_firstname,vc\_staff\_surname);

END proc\_staff;

/

SHOW ERRORS

-------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------------------------------

--CREATING PROCEDURE NAMED proc\_update\_staff

CREATE OR REPLACE PROCEDURE proc\_update\_staff IS

vn\_staff\_id staff.staff\_id%TYPE;

BEGIN

SELECT seq\_staff.CURRVAL

INTO vn\_staff\_id

FROM DUAL;

UPDATE staff SET staff\_firstname ='SAMI'

WHERE staff\_id =vn\_staff\_id;

END proc\_update\_staff;

/

SHOW ERRORS

-----------------------------------6\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------------

--CREATING A PROCEDURE NAMED proc\_delete\_staff

CREATE OR REPLACE PROCEDURE proc\_delete\_staff IS

vn\_staff\_id staff.staff\_id%TYPE;

BEGIN

SELECT seq\_staff.CURRVAL

INTO vn\_staff\_id

FROM DUAL;

DELETE FROM staff

WHERE staff\_id=vn\_staff\_id;

END proc\_delete\_staff;

/

SHOW ERRORS;

--procedure created sucessfully

------------------------------7\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---------------------------------

--CREATING PROCEDURE TO GET THE SALARY OF THE STAFF

CREATE OR REPLACE PROCEDURE proc\_get\_user\_salary IS

vn\_staff NUMBER(11) := 1000;

vc\_username VARCHAR2(20);

vn\_salary NUMBER(8,2);

vn\_ceilsalary NUMBER(10);

vn\_floorsalary NUMBER(10);

vn\_roundsalary NUMBER(10);

vn\_truncatesalary NUMBER(10);

vn\_length NUMBER(30);

vn\_counter NUMBER(30) :=3;

BEGIN

SELECT COUNT(\*) INTO vn\_length FROM staff;

DBMS\_OUTPUT.PUT\_LINE('Username Salary Ceil Floor Round Truncate');

LOOP

SELECT staff\_salary INTO vn\_salary FROM staff WHERE staff\_id = vn\_staff;

SELECT CEIL(vn\_salary) INTO vn\_ceilsalary FROM staff WHERE staff\_id = vn\_staff;

SELECT FLOOR(vn\_salary) INTO vn\_floorsalary FROM staff WHERE staff\_id = vn\_staff;

SELECT ROUND(vn\_salary) INTO vn\_roundsalary FROM staff WHERE staff\_id = vn\_staff;

SELECT TRUNC(vn\_salary) INTO vn\_truncatesalary FROM staff WHERE staff\_id = vn\_staff;

SELECT CONCAT(SUBSTR(INITCAP(TRIM(TRAILING ' ' FROM staff\_firstname)),1,2),SUBSTR(INITCAP(staff\_surname),1,5))

INTO vc\_username

FROM staff

WHERE staff\_id = vn\_staff;

--START THE LOOP

--CONDITION FOR TERMINATE THE LOOP

EXIT WHEN vn\_counter = vn\_length;

DBMS\_OUTPUT.PUT\_LINE(vc\_username || ' ' || vn\_salary || ' '|| vn\_ceilsalary || ' '|| vn\_floorsalary || ' ' || vn\_roundsalary || ' ' || vn\_truncatesalary);

vn\_staff := vn\_staff+2;

vn\_length := vn\_length-1;

END LOOP;--END OF LOOP

END;

/

--Procedure created.

SHOW ERRORS;

--no errors

-----8---------

--CREATING UNNAMED PROCEDURE USING LOOP

DECLARE

vn\_length NUMBER(2);

vn\_counter NUMBER(3) := 1;

vc\_firstname VARCHAR2(20) := 'ANUJ';

vc\_surname VARCHAR2(20) := 'MAGAR';

BEGIN

vn\_length := LENGTH(vc\_firstname);

--START THE LOOP

LOOP

--LOOP TERMINATION CONDITION

EXIT WHEN vn\_counter>vn\_length;

DBMS\_OUTPUT.PUT\_LINE(SUBSTR(vc\_firstname, vn\_counter, 1));

vn\_counter := vn\_counter + 1;

--END OF LOOP

END LOOP;

--DISPLAY SURNAME

DBMS\_OUTPUT.PUT\_LINE(vc\_surname);

--END OF PROGRAMME

END;

/

-----------------------------------------9\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------

--CREATING PROCEDURE TO DISPLAY DETAILS OF STAFF

CREATE OR REPLACE PROCEDURE proc\_display\_staff(in\_id staff.staff\_id%TYPE)IS

vc\_name staff.staff\_firstname%TYPE;

vc\_lastname staff.staff\_surname%TYPE;

vc\_gender staff.staff\_gender%TYPE;

vd\_date staff.staff\_hire\_date%TYPE;

vn\_salary staff.staff\_salary%TYPE;

BEGIN

SELECT staff\_firstname INTO vc\_name FROM staff WHERE staff\_id =in\_id;

SELECT staff\_surname INTO vc\_lastname FROM staff WHERE staff\_id =in\_id;

SELECT staff\_gender INTO vc\_gender FROM staff WHERE staff\_id =in\_id;

SELECT staff\_hire\_date INTO vd\_date FROM staff WHERE staff\_id =in\_id;

SELECT staff\_salary INTO vn\_salary FROM staff WHERE staff\_id =in\_id;

DBMS\_OUTPUT.PUT\_LINE('The details of staff with id ' ||in\_id || ' is: ' || vc\_name || ' '|| vc\_lastname ||' gender is: '|| vc\_gender ||' and the date of birth is: '|| vd\_date ||' and salary is: '|| vn\_salary);

END proc\_display\_staff;

/

SHOW ERRORS;

-------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*10\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-------------------

--CHECKING THE STATUS OF THE STAFF TABLE

DESC staff;

SELECT staff\_id,staff\_firstname,staff\_surname ,staff\_hire\_date,staff\_dob

FROM staff;

--CREATING PROCEDURE NAMED proc\_insert\_staff

CREATE OR REPLACE PROCEDURE proc\_insert\_staff(in\_staff\_firstname VARCHAR2,in\_staff\_surname VARCHAR2,in\_staff\_hire\_date DATE,in\_staff\_dob DATE) IS

vc\_firstname staff.staff\_firstname%TYPE;

vc\_surname staff.staff\_surname%TYPE;

vd\_hire\_date staff.staff\_hire\_date%TYPE;

vd\_dob staff.staff\_hire\_date%TYPE;

vc\_command VARCHAR2(1000);

BEGIN

SELECT in\_staff\_firstname INTO vc\_firstname FROM DUAL;

SELECT in\_staff\_surname INTO vc\_surname FROM DUAL;

SELECT in\_staff\_hire\_date INTO vd\_hire\_date FROM DUAL;

SELECT in\_staff\_dob INTO vd\_dob FROM DUAL;

INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_hire\_date,staff\_dob)

VALUES(seq\_staff.NEXTVAL,vc\_firstname,vc\_surname,vd\_hire\_date,vd\_dob);

END proc\_insert\_staff;

/

SHOW ERRORS;

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Procedure Testing Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\procedure\_testing09.txt

\*/

---1---

--TESTING FOR UNNAMED BLOCK

--Unamed blocked sucessfully compiled showing the title, firstname and surname

--OUTPUT OF THE PROCEDURE UNNAMED BLOCK

/\*

--Title--

M

R

S

--Firstname--

P

R

A

T

I

M

A

--Surname--

D

A

H

A

L

--PL/SQL procedure successfully completed.

\*/

------2---

/\*UNNAMED BLOCK

--RETURNS THE NAME OF STAFF WITH PARTICULAR ID

Name: PRATIMA

PL/SQL procedure successfully completed.

\*/

-----3------

--TESTING

--SELECT ALL TABLE NAME FROM TAB

SELECT TNAME FROM TAB;

--7 ROWS RETURNED

--EXECUTE THE PROCEDURE

EXEC proc\_create\_team;

--SELECT ALL TABLE NAME FROM TAB

SELECT TNAME FROM TAB;

--8 ROWS RETURNED

----4---

--TESTING

--SELECT ALL PRIMARY KEY

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'PK%';

--5 rows returned

--EXECUTE THE PROCEDURE

EXEC proc\_primary\_team;

--SELECT ALL PRIMARY KEY

SELECT constraint\_name

FROM user\_constraints

WHERE constraint\_name

LIKE 'PK%';

--6 rows returned

------5-------

/\* TESTING PROCEDURE

\*/

--CHECKING THE STATUS OF THE STAFF TABLE

DESC staff;

/\*OUTPUT

Name Null? Type

----------------------------------------- -------- ----------------------------

STAFF\_ID NOT NULL NUMBER(11)

STAFF\_TITLE VARCHAR2(5)

STAFF\_FIRSTNAME VARCHAR2(30)

STAFF\_SURNAME VARCHAR2(30)

STAFF\_CONTACT CONTACT\_VARRAY\_TYPE

STAFF\_ADDRESS REF OF ADDRESS\_TYPE

SOCIAL\_MEDIA\_USED SOCIAL\_MEDIA\_TABLE\_TYPE

STAFF\_GENDER CHAR(1)

STAFF\_BLOOD\_GROUP CHAR(3)

STAFF\_HIRE\_DATE DATE

STAFF\_SALARY NUMBER(8,2)

STAFF\_DOB DATE

\*/

SELECT staff\_id,staff\_firstname,staff\_surname

FROM staff;

/\*10 rows selected

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME

---------- ------------------------------ ------------------------------

1088 ARYAN KARKI

1094 RANEET KUNWAR

1000 PRATIMA DAHAL

1002 SWASTIKA ADHIKARI

1004 NITU DAS

1006 DIKSHYA GHALE

1008 SARENA DAHAL

1010 CHRIS HEMSWORTH

1016 ANUJA MAGAR

1042 AMRIT THAPA

\*/

--EXECUTING proc\_staff

EXEC proc\_staff;

--PL/SQL procedure successfully completed.

--VIEWING THE PROCEDURE TO SEE IF THE DATA IS INSERTED OR NOT

SELECT staff\_id,staff\_firstname,staff\_surname

FROM staff;

/\* 11 rows selected

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME

---------- ------------------------------ ------------------------------

1088 ARYAN KARKI

1094 RANEET KUNWAR

1000 PRATIMA DAHAL

1002 SWASTIKA ADHIKARI

1004 NITU DAS

1006 DIKSHYA GHALE

1008 SARENA DAHAL

1010 CHRIS HEMSWORTH

1016 ANUJA MAGAR

1042 AMRIT THAPA

1120 AMRIT THAPA

\*/

-----5-----

--VIEWING WHETHER THE PROCEDURE IS UPDATED OR NOT

SELECT staff\_id,staff\_firstname,staff\_surname

FROM staff;

/\*OUTPUT

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME

---------- ------------------------------ ------------------------------

1088 ARYAN KARKI

1094 RANEET KUNWAR

1000 PRATIMA DAHAL

1002 SWASTIKA ADHIKARI

1004 NITU DAS

1006 DIKSHYA GHALE

1008 SARENA DAHAL

1010 CHRIS HEMSWORTH

1016 ANUJA MAGAR

1042 AMRIT THAPA

1120 AMRIT THAPA

11 rows selected.

\*/

--EXECUTING THE PROCEDURE proc\_update\_staff

EXEC proc\_update\_staff;

--VIEWING WHETHER THE PROCEDURE IS UPDATED OR NOT

SELECT staff\_id,staff\_firstname,staff\_surname

FROM staff;

/\*procedure is updated

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME

---------- ------------------------------ ------------------------------

1088 ARYAN KARKI

1094 RANEET KUNWAR

1000 PRATIMA DAHAL

1002 SWASTIKA ADHIKARI

1004 NITU DAS

1006 DIKSHYA GHALE

1008 SARENA DAHAL

1010 CHRIS HEMSWORTH

1016 ANUJA MAGAR

1042 AMRIT THAPA

1120 SAMI THAPA

11 rows selected.

\*/

----6-----

/\*TEST proc\_delete\_staff

\*/

--CHECKING THE STATUS OF STAFF TABLE

DESC staff;

SELECT staff\_id,staff\_firstname,staff\_surname

FROM staff;

/\*

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME

---------- ------------------------------ ------------------------------

1088 ARYAN KARKI

1094 RANEET KUNWAR

1000 PRATIMA DAHAL

1002 SWASTIKA ADHIKARI

1004 NITU DAS

1006 DIKSHYA GHALE

1008 SARENA DAHAL

1010 CHRIS HEMSWORTH

1016 ANUJA MAGAR

1042 AMRIT THAPA

1120 SAMI THAPA

11 rows selected.

\*/

--EXECUTING proc\_delete\_staff

EXEC proc\_delete\_staff;

--PL/SQL procedure successfully completed.

--CHECKING THE STATUS OF THE STAFF TABLE

SELECT staff\_id,staff\_firstname,staff\_surname

FROM staff;

/\*

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME

---------- ------------------------------ ------------------------------

1088 ARYAN KARKI

1094 RANEET KUNWAR

1000 PRATIMA DAHAL

1002 SWASTIKA ADHIKARI

1004 NITU DAS

1006 DIKSHYA GHALE

1008 SARENA DAHAL

1010 CHRIS HEMSWORTH

1016 ANUJA MAGAR

1042 AMRIT THAPA

10 rows selected.

\*/

-----7-----

/\*TESTING proc\_get\_user\_salary\*/

--VIEWING THE SALARY OF THE STAFF

SELECT staff\_salary FROM staff;

/\*OUTPUT

STAFF\_SALARY

------------

20002

30003

50005

40004

60006

50005

60006

7 rows selected.

\*/

--EXECUTING proc\_get\_user\_salary

EXEC proc\_get\_user\_salary;

/\*

Username Salary Ceil Floor Round Truncate

PrDahal 20000 20000 20000 20000 20000

SwAdhik 30000 30000 30000 30000 30000

NiDas 50000 50000 50000 50000 50000

DiGhale 40000 40000 40000 40000 40000

PL/SQL procedure successfully completed.

\*/

--VIEWING THE SALARY OF THE STAFF

SELECT staff\_salary FROM staff;

/\*

STAFF\_SALARY

------------

20000

30000

50000

40000

60000

50000

60000

7 rows selected.

\*/

-------8--------

/\*

A

N

U

J

MAGAR

\*/

-------9--------

-------\*\*\*\*\*\*\*\*\*\*TESTING\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-----------

SELECT staff\_firstname

FROM staff;

/\*

STAFF\_FIRSTNAME

------------------------------

PRATIMA

SWASTIKA

NITU

DIKSHYA

SARENA

CHRIS

SCARLETT

7 rows selected.

\*/

---EXECUTING THE ABOVE PROCEDURE

EXEC proc\_display\_staff(1000);

--procedure is executed giving the details of staff id 1000

/\*

The details of staff with id 1000 is: PRATIMA DAHAL gender is: F and the date of

birth is: 18-FEB-15 and salary is: 20000

\*/

------10-------

set long 2000

set linesize 200

EXEC proc\_insert\_staff('AMRIT','THAPA','01-FEB-2000','01-FEB-1980');

--procedure executed sucessfully

--CHECKING THE STATUS OF THE STAFF TABLE

DESC staff;

--VIEWING WHETHER THE DATA IS INSERTED OR NOT

SELECT staff\_id,staff\_firstname,staff\_surname ,staff\_hire\_date,staff\_dob

FROM staff;

/\*

STAFF\_ID STAFF\_FIRSTNAME STAFF\_SURNAME STAFF\_HIR STAFF\_DOB

---------- ------------------------------ ------------------------------ --------- ---------

1000 PRATIMA DAHAL 18-FEB-15 19-MAR-95

1002 SWASTIKA ADHIKARI 20-APR-14 20-APR-94

1004 NITU DAS 18-FEB-16 01-NOV-93

1006 DIKSHYA GHALE 23-NOV-17 19-DEC-92

1008 SARENA DAHAL 29-DEC-15 12-MAR-91

1010 CHRIS HEMSWORTH 19-APR-19

1012 SCARLETT EVANS 19-APR-19

1014 AMRIT THAPA 01-FEB-00 01-FEB-80

1016 AMRIT THAPA 01-FEB-00 01-FEB-80

1018 AMRIT THAPA 01-FEB-00 01-FEB-80

10 rows selected.

\*/

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Functions Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\alters\_09.txt

\*/

----------------------------------------1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-----------------------------

--DESCRIBING THE STRUCTURE OF FESTIVAL TABLE

DESC festivals;

--CREATING FUNCTION NAMED func\_festivals

CREATE OR REPLACE FUNCTION func\_festivals

RETURN NUMBER IS

vn\_festivals NUMBER(3);

BEGIN

SELECT COUNT(\*)

INTO vn\_festivals

FROM festivals;

RETURN vn\_festivals;

END func\_festivals;

/

SHOW ERRORS

--FUNCTION CREATED WITH NO ERRORS

--CREATING PROCEDURE TO RUN ABOVE FUNCTION FUNC\_FESTIVALS

CREATE OR REPLACE PROCEDURE proc\_festivals IS

vc\_festival\_name VARCHAR2(50);

BEGIN

vc\_festival\_name:=func\_festivals;

DBMS\_OUTPUT.PUT\_LINE('The number of festivals is' || func\_festivals);

END proc\_festivals;

/

SHOW ERRORS;

--EXECUTING THE PROCEDURE proc\_festivals

EXEC proc\_festivals;

------------------------------2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------------------------------

--DESCRIBING THE TABLE STAFF

DESC staff;

--VIEWING THE DATA OF STAFF

SELECT \* FROM staff;

--CREATING FUNCTION TO GET AVERAGE AGE OF STAFF

CREATE OR REPLACE FUNCTION func\_get\_avg\_age\_of\_staff

RETURN NUMBER IS

today DATE;

vn\_avg\_age NUMBER(3);

BEGIN

SELECT sysdate INTO today from dual;

SELECT AVG(FLOOR(today-staff\_dob) / 365.2425) INTO vn\_avg\_age FROM staff

WHERE staff\_dob IS NOT NULL;

RETURN vn\_avg\_age;

END func\_get\_avg\_age\_of\_staff;

/

SHOW ERRORS;

--FUNCTION CREATED WITH NO ERRORS

--CREATING PROCEDURE TO TEST WHETHER FUNCTION WORKES OR NOT

CREATE OR REPLACE PROCEDURE proc\_avg\_age\_staff IS

vn\_avg\_age NUMBER(3);

BEGIN

vn\_avg\_age:=func\_get\_avg\_age\_of\_staff;

DBMS\_OUTPUT.PUT\_LINE('THE AVERAGE AGE OF STAFF IS ' || vn\_avg\_age || ' YEARS.');

END proc\_avg\_age\_staff;

/

SHOW ERRORS;

--PROCEDURE CREATED WITH NO ERRORS

EXEC proc\_avg\_age\_staff;

--PROCEDURE SUCESSFULLY CREATED SHOWING THE AVERAGE AGE OF STAFF

----------------------------3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-------------------------------------------------

--CREATE A FUNCTION TO CREATE A USERNAME FROM THE FIRST NAME AND THE SURNAME OF THE TABLE STAFF

CREATE OR REPLACE FUNCTION func\_staff(in\_staff\_id NUMBER) RETURN VARCHAR2 IS

vc\_stff\_un VARCHAR2(10);

BEGIN

SELECT CONCAT(SUBSTR(staff\_firstname,1,2),SUBSTR(staff\_surname,1,5))

INTO vc\_stff\_un

FROM staff

WHERE staff\_id = in\_staff\_id;

RETURN vc\_stff\_un;

END func\_staff;

/

--FIND THE ERRORS

SHOW ERRORS;

---------------------

--CREATE A PROCEDURE FOR A FUNCTION func\_staff TO EXECUTE THE FUNCTION

CREATE OR REPLACE PROCEDURE proc\_staff(in\_staff\_id staff.staff\_id%TYPE) IS

vc\_staff\_username VARCHAR2(30);

--vc prefix is used for the new variable because we are storing varchar2(string) data

BEGIN

vc\_staff\_username := func\_staff(in\_staff\_id);

DBMS\_OUTPUT.PUT\_LINE ('The staff with user id ' || in\_staff\_id || ' has a username '||vc\_staff\_username);

END proc\_staff;--END OF THE PROCEDURE

/

EXEC proc\_staff(1000);

--procedure sucessfully executed showing the staff id with with username.

--TESTING

-- 1)BEFORE EXECUTING THE PROCEDURE

SELECT \* FROM STAFF WHERE staff\_id = 1000;

--1 rows return

-- 2) EXECUTING THE PROCEDURE PROC\_STAFF

EXEC PROC\_STAFF(1000);

--1 rows returned with message There staff with user id 1000 has a username PRDAHAL

-- 3) AFTER EXECUTING THE PROCEDURE

SELECT \* FROM STAFF;

-- 1 rows returned

-------------------------------4\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*---------------------------------------------------------------------

--CREATE A FUNCTION TO CREATE A USERNAME FROM THE FIRST NAME AND THE SURNAME OF THE TABLE STAFF

CREATE OR REPLACE FUNCTION func\_staff\_check\_gender(in\_staff\_id NUMBER) RETURN VARCHAR2 IS

vc\_stff\_gender staff.staff\_gender%TYPE;

BEGIN

SELECT staff\_gender INTO vc\_stff\_gender FROM staff

WHERE staff\_id = in\_staff\_id;

RETURN vc\_stff\_gender;

END func\_staff\_check\_gender;

/

--FIND THE ERRORS

SHOW ERRORS;

--No errors

--CREATE A PROCEDURE FOR A FUNCTION func\_staff TO EXECUTE THE FUNCTION

CREATE OR REPLACE PROCEDURE proc\_staff\_gender(in\_staff\_id staff.staff\_id%TYPE) IS

vc\_staff\_gender staff.staff\_gender%TYPE;

--vc prefix is used for the new variable because we are storing varchar2(string) data

BEGIN

vc\_staff\_gender := func\_staff\_check\_gender(in\_staff\_id);

CASE

WHEN vc\_staff\_gender = 'M' THEN DBMS\_OUTPUT.PUT\_LINE('STAFF IS MALE');

WHEN vc\_staff\_gender = 'F' THEN DBMS\_OUTPUT.PUT\_LINE('STAFF IS FEMALE');

ELSE DBMS\_OUTPUT.PUT\_LINE('OTHERS');

END CASE;

END proc\_staff\_gender;--END OF THE PROCEDURE

/

EXEC proc\_staff\_gender(1000);

--PROCEDURE SUCESSFULLY EXECUTED

-----------------------------5----------------------------------------------------

--create a function func\_instructor\_ct which return no of instructor

CREATE OR REPLACE FUNCTION func\_total\_salary RETURN NUMBER IS

vn\_staff\_salary NUMBER(10,2);

BEGIN

SELECT SUM(staff\_salary)

INTO vn\_staff\_salary

FROM staff;

RETURN vn\_staff\_salary;

END func\_total\_salary;

/

--FUNCTION CREATED

--VIEW ERRORS

SHOW ERRORS;

--CREATE THE PROCEDURE TO TEST THE FUNCTION WHICH PRINT THE NO OF INSTRUCTOR

CREATE OR REPLACE PROCEDURE proc\_salary\_func IS

vn\_total\_staff\_salary NUMBER(10,2);

--vN prefix is used for the new variable because we are storing NUMBER data

BEGIN

vn\_total\_staff\_salary := func\_total\_salary;

DBMS\_OUTPUT.PUT\_LINE ('The total salary of staff is '|| vn\_total\_staff\_salary );

END proc\_salary\_func;--END OF THE PROCEDURE

/

--testing

-- 1) Select without execute

SELECT \* FROM staff;

--8 rows returned

-- 2) Execute the procedure

EXEC proc\_salary\_func;

-- 3)Select after execute

SELECT \* FROM staff;

--8 rows returned

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Procedure Testing Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\function\_testing\_09.txt

\*/

-----1------

--EXECUTING THE PROCEDURE proc\_festivals

EXEC proc\_festivals;

--OUTPUT

/\*

The number of festivals is 5

PL/SQL procedure successfully completed.

\*/

---------2----------------

--EXECUTING THE PROCEDURE proc\_avg\_age\_staff

EXEC proc\_avg\_age\_staff;

/\*

THE AVERAGE AGE OF STAFF IS 31 YEARS.

PL/SQL procedure successfully completed.

\*/

-------3-------

--TESTING PROC\_STAFF

-- 1)BEFORE EXECUTING THE PROCEDURE

SELECT \* FROM STAFF WHERE staff\_id = 1000;

--1 rows return

-- 2) EXECUTING THE PROCEDURE PROC\_STAFF

EXEC PROC\_STAFF(1000);

--1 rows returned with message There staff with user id 1000 has a username PRDAHAL

-- 3) AFTER EXECUTING THE PROCEDURE

SELECT \* FROM STAFF;

-- 1 rows returned

---------4------

--TESTING proc\_staff\_gender

EXEC proc\_staff\_gender(1000);

/\*

STAFF IS FEMALE

PL/SQL procedure successfully completed.

\*/

------5-----

--TESTING

-- 1) SELECT WITHOUT EXECUTE

SELECT \* FROM staff;

--8 rows returned

-- 2) EXECUTE THE PROCEDURE

EXEC proc\_salary\_func;

/\*

The total salary of staff is 310000

PL/SQL procedure successfully completed.

\*/

-- 3)SELECT AFTER EXECUTE

SELECT \* FROM staff;

--8 rows returned

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Trigger Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\trigger\_09.txt

\*/

--Group 9

--TRIGGER

SET SERVEROUTPUT ON;

--ENABLING SERVEROUTPUT TO DISPLAY SERVER MESSAGES

SET SERVEROUTPUT ON;

----------------------1\*\*\*\*\*\*\*\*\*\*\*\*\*-------------------

--DESCRIBING THE TABLE STAFF

DESC staff;

--function to get staff age

CREATE OR REPLACE FUNCTION func\_get\_staff\_age (in\_staff\_dob DATE)

RETURN NUMBER IS

vn\_staff\_age NUMBER(3);

BEGIN

vn\_staff\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, in\_staff\_dob)/12);

RETURN vn\_staff\_age;

END func\_get\_staff\_age;

/

SHOW ERRORS;

--function creared with no errors

--TESTING FOR FUNCTION

--VIEWING THE DATA OF STAFF

SELECT \* FROM staff;

---------

--CREATING PROCEDURE TO TEST WHETHER FUNCTION WORKES OR NOT

CREATE OR REPLACE PROCEDURE proc\_staff\_age(in\_staff\_id NUMBER) IS

vd\_dob DATE;

vn\_age NUMBER(3);

BEGIN

SELECT staff\_dob

INTO vd\_dob

FROM staff

WHERE staff\_id = in\_staff\_id;

vn\_age := func\_get\_staff\_age(vd\_dob);

DBMS\_OUTPUT.PUT\_LINE('Staff is of age ' || vn\_age);

END;

/

SHOW ERRORS;

--procedure created with no errors

--EXECUTING THE PROCEDURE

EXEC proc\_staff\_age(1000);

--procedures sucessfully completed

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--CREATING A TRIGGER TO VALIDATE STAFF AGE

CREATE OR REPLACE TRIGGER trig\_staff\_age

BEFORE UPDATE OF staff\_dob OR INSERT ON staff

FOR EACH ROW

WHEN(NEW.staff\_dob IS NOT NULL)

DECLARE

vd\_today DATE;

--vn\_age NUMBER(3);

BEGIN

SELECT SYSDATE

INTO vd\_today

FROM DUAL;

IF :NEW.staff\_dob>(vd\_today-(365\*18)) THEN

RAISE\_APPLICATION\_ERROR(-20000,'The staff is underage');

--vn\_age:=func\_get\_staff\_age(:NEW.staff\_dob);

ELSIF :NEW.staff\_dob>SYSDATE THEN

RAISE\_APPLICATION\_ERROR(-20000,'The staff is not born yet');

END IF;

END trig\_staff\_age;

/

SHOW ERRORS;

--trigger created with no errors

-------------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2\*\*\*\*\*\*\*\*\*\*\*\*\*---------------------------------------------------------

--CREATING TABLE hr\_event

CREATE TABLE hr\_event(

event\_type VARCHAR2(30),

login\_date DATE,

logon\_time VARCHAR2(15),

logoff\_date DATE,

logoff\_time VARCHAR2(15)

);

--CREATING TRIGGER TO SHOW LOG IN DETAILS OF STAFF

CREATE OR REPLACE trigger trig\_login\_hr

AFTER LOGON ON SCHEMA

DECLARE

logons VARCHAR2(30);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('one line Updated by '|| SYSDATE);

INSERT INTO hr\_event VALUES(

ora\_sysevent,

SYSDATE,

TO\_CHAR(SYSDATE , 'hh24:mi:ss'),

NULL,

NULL

);

COMMIT;

END trig\_login\_hr;

/

--(Oracle,2019)

------------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------------------------------

--CREATING TRIGGER trig\_operation\_staff

CREATE OR REPLACE TRIGGER trig\_operation\_staff

BEFORE INSERT OR UPDATE OR DELETE ON STAFF

FOR EACH ROW

ENABLE

DECLARE

c\_user VARCHAR2(15);

BEGIN

SELECT

USER INTO c\_user FROM dual;

IF INSERTING THEN

DBMS\_OUTPUT.PUT\_LINE('One line inserted by '||c\_user ||' on ' || SYSDATE);

ELSIF UPDATING THEN

DBMS\_OUTPUT.PUT\_LINE('One line Updated by '||c\_user||' on ' || SYSDATE);

ELSIF DELETING THEN

DBMS\_OUTPUT.PUT\_LINE('One line Deleted by '||c\_user ||' on ' || SYSDATE);

END IF;

END;

/

--TRIGGERS CREATED WITH NO ERRORS

-----------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*4\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----------------------------------------

--CREATING TRIGGER trig\_audit\_tr

CREATE OR REPLACE TRIGGER trig\_audit\_tr

AFTER DDL ON DATABASE

BEGIN

DBMS\_OUTPUT.PUT\_LINE('one row '||ora\_sysevent || '...... ON ' || SYSDATE);

END;

/

--triggers created

SHOW ERRORS;

--no errors

------------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--------------------------------------------

--CREATING TRIGGER trig\_message

CREATE OR REPLACE TRIGGER trig\_message

AFTER STARTUP ON DATABASE

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello..');

END;

/

--(Oracle,2019)

--triggers created

SHOW ERRORS;

--no errors

SELECT \* FROM hr\_event;

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Trigger Testing Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\trigger\_testing09.txt

\*/

-----------1----------

--TESTING FOR TRIGGER

SELECT \* FROM staff;

--8 rows selected

--INSERTING THE VALUES IN STAFF

INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)

VALUES (seq\_staff.NEXTVAL,'ARYAN','KARKI','01-JAN-2006');

--NO VALUE INSERTED

INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)

VALUES (seq\_staff.NEXTVAL,'ARYAN','KARKI','01-JAN-2026');

--SMALL MISTAKE

INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)

VALUES (seq\_staff.NEXTVAL,'ARYAN','KARKI','01-JAN-1996');

SELECT \* FROM staff;

--9 rows selected

-------2----------

--TESTING

--1)BEFORE EXECUTION

SELECT \* FROM hr\_event;

--NO ROWS SELECTED

--2)DISCONNECT FROM SQL PLUS

DISC;

--3)CONNECT SCHEMA

CONNECT dbs/babulal

--4)SELECT FROM HR\_EVENT TABLE

SELECT \* FROM hr\_event;

--1 ROWS SELECTED

-----------3------------------

--TESTING

desc staff

--1)CHECK ON OPERATION INSERT

INSERT INTO staff(staff\_id,staff\_firstname,staff\_surname,staff\_dob)

VALUES (seq\_staff.NEXTVAL,'RANEET','KUNWAR','01-JAN-1996');

--ONE LINE INSERTED BY PRATIMA ON 17-APR-19

--1 ROW CREATED

--2)CHECK ON OPERATION UPDATE

UPDATE staff

SET staff\_gender = 'M'

WHERE staff\_id = 1000;

--ONE LINE UPDATED BY PRATIMA ON 17-APR-19

--1 ROW UPDATED

--3)CHECK ON OPERATION DELETE

DELETE staff WHERE staff\_id=1028;

--ONE LINE DELETED BY PRATIMA ON 17-APR-19

--1 ROW DELETED

------------------4--------------

--CREATING NEW TABLE

CREATE TABLE jname(dob DATE);

--OUTPUT

--one row CREATE...... ON 19-APR-19

--UPDATE TABLE

ALTER TABLE jname

ADD(salary NUMBER(2,3));

--OUTPUT

--one row ALTER...... ON 19-APR-19

--DROPPPING TABLE

DROP TABLE jname;

--OUTPUT

--one row DROP...... ON 19-APR-19

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Package Creation Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\package\_09.txt

\*/

--PACKAGE TO SHOW STAFF DETAILS BY EMAIL

--PACKAGE HEADER

CREATE OR REPLACE PACKAGE pkg\_search\_staff AS

FUNCTION func\_search\_by\_email\_cursor(in\_email VARCHAR2) RETURN NUMBER;

PROCEDURE proc\_search\_by\_email\_cursor(in\_email VARCHAR2);

END pkg\_search\_staff;

/

--PACKAGE BODY

CREATE OR REPLACE PACKAGE BODY pkg\_search\_staff AS

vn\_staff\_id staff.staff\_id%TYPE;

vc\_staff\_firstname staff.staff\_firstname%TYPE;

vc\_staff\_surname staff.staff\_surname%TYPE;

vc\_staff\_gender staff.staff\_gender%TYPE;

exep\_no\_user EXCEPTION; --EXCEPTION

CURSOR cur\_staff\_id IS

SELECT s.staff\_id staff\_id,c.email email FROM staff s, table(s.staff\_contact) c;

rec\_cur\_id cur\_staff\_id%ROWTYPE;

--FUNCTION TO GET USER ID BASED ON EMAIL

FUNCTION func\_search\_by\_email\_cursor(in\_email VARCHAR2)

RETURN NUMBER IS

BEGIN

IF cur\_staff\_id %ISOPEN THEN

CLOSE cur\_staff\_id ;

END IF;

OPEN cur\_staff\_id;

FETCH cur\_staff\_id INTO rec\_cur\_id;

--IF THERE IS NO DATA IN STAFF TABLE

IF cur\_staff\_id%NOTFOUND THEN

RETURN 0;

ELSE

--WHILE LOOP TO SCROLL THROUGH ALL DATA

WHILE cur\_staff\_id%FOUND LOOP

IF rec\_cur\_id.email = in\_email THEN

RETURN rec\_cur\_id.staff\_id;

ELSE RETURN 0;

END IF;

END LOOP;

END IF;

CLOSE cur\_staff\_id;

END;

--PROCEDURE WITHIN PACKAGE TO DISPLAY INFORMATION

PROCEDURE proc\_search\_by\_email\_cursor(in\_email VARCHAR2) IS

BEGIN

vn\_staff\_id := func\_search\_by\_email\_cursor(in\_email);

IF (vn\_staff\_id=0) THEN

RAISE exep\_no\_user;

ELSE

SELECT staff\_firstname INTO vc\_staff\_firstname FROM staff WHERE staff\_id = vn\_staff\_id;

SELECT staff\_surname INTO vc\_staff\_surname FROM staff WHERE staff\_id = vn\_staff\_id;

SELECT staff\_gender INTO vc\_staff\_gender FROM staff WHERE staff\_id = vn\_staff\_id;

DBMS\_OUTPUT.PUT\_LINE('Staff\_id: '|| vn\_staff\_id ||' FIRSTNAME: ' || vc\_staff\_firstname || ' SURNAME: ' || vc\_staff\_surname);

END IF;

EXCEPTION

WHEN exep\_no\_user THEN

DBMS\_OUTPUT.PUT\_LINE('NOT FOUND');

END;

END pkg\_search\_staff;

/

--(Docs.oracle.com, 2019)

--(Oracle Docs,2019)

SHOW ERRORS;

--DISPLAYING PACKAGE NAME

SELECT OBJECT\_NAME FROM USER\_OBJECTS WHERE OBJECT\_NAME LIKE 'PKG\_%';

--TESTING OF PACKAGE

EXEC pkg\_search\_staff.proc\_search\_by\_email\_cursor('PRATIMA@GMAIL.COM');

--STAFF INFORMATION DISPLAYED

--TESTING PACKAGE USING INVALID DATA

EXEC pkg\_search\_staff.proc\_search\_by\_email\_cursor('ANOOZ@GMAIL.COM');

--RELATIVE INFORMATION RETURNED

/\*

Group No:09

Group Members: Anuj Budha Magar, Amrit Thapa, Pratima Dahal, Swastika Adhikari

Script: Drop Script

Module : csy2038 - Database 2

Module Tutor: Sangita Satyal

Path: @E:\DatabaseYEAR2\assignment\drop\_09.txt

\*/

--DROPPING TRIGGER

DROP TRIGGER trig\_staff\_age;

DROP TRIGGER trig\_login\_hr;

DROP TRIGGER trig\_operation\_staff;

DROP TRIGGER trig\_audit\_tr;

DROP TRIGGER trig\_message;

--DROPPING CURSOR

DROP PROCEDURE proc\_fes\_imp\_cursor;

DROP PROCEDURE proc\_imp\_cursor;

DROP PROCEDURE proc\_exp\_cursor;

DROP PROCEDURE proc\_exp\_staff\_salary;

--DROPPING FUNCTION

DROP FUNCTION func\_festivals;

DROP FUNCTION func\_get\_staff\_age;

DROP FUNCTION func\_get\_avg\_age\_of\_staff;

DROP FUNCTION func\_staff;

DROP FUNCTION func\_staff\_check\_gender;

DROP FUNCTION func\_total\_salary;

--DROPPING PACKAGE

DROP PACKAGE pkg\_search\_staff;

--DROPPING PROCEDURE

DROP PROCEDURE proc\_create\_team;

DROP PROCEDURE proc\_primary\_team;

DROP PROCEDURE proc\_foreign\_team;

DROP PROCEDURE proc\_update\_staff;

DROP PROCEDURE proc\_delete\_staff;

DROP PROCEDURE proc\_get\_user\_salary;

DROP PROCEDURE proc\_display\_staff;

DROP PROCEDURE proc\_insert\_staff;

DROP PROCEDURE proc\_festivals;

DROP PROCEDURE proc\_staff\_age;

DROP PROCEDURE proc\_avg\_age\_staff;

DROP PROCEDURE proc\_staff;

DROP PROCEDURE proc\_staff\_gender;

DROP PROCEDURE proc\_salary\_func;

--DROPPING CONSTRAINTS

--DROPPING FOREIGN KEYS

ALTER TABLE festivals

DROP CONSTRAINT fk\_f\_locations;

ALTER TABLE staff

DROP CONSTRAINT fk\_s\_staff;

ALTER TABLE team

DROP CONSTRAINT fk\_t\_staff;

ALTER TABLE festivals

DROP CONSTRAINT fk\_f\_festival\_nature;

ALTER TABLE festival\_staff

DROP CONSTRAINT fk\_f\_staff;

ALTER TABLE festival\_staff

DROP CONSTRAINT fk\_f\_festivals;

--DROPPING UNIQUE CONSTRAINT

ALTER TABLE festival\_nature

DROP CONSTRAINT uc\_festival\_nature;

--DROPPING CHECK CONSTRAINT

ALTER TABLE staff

DROP CONSTRAINT ck\_gender;

ALTER TABLE staff

DROP CONSTRAINT ck\_staff\_title;

ALTER TABLE staff

DROP CONSTRAINT ck\_staff\_firstname;

ALTER TABLE staff

DROP CONSTRAINT ck\_staff\_surname;

ALTER TABLE festival\_nature

DROP CONSTRAINT ck\_festival\_type;

--DROPPING PRIMARY KEYS

ALTER TABLE locations

DROP CONSTRAINT pk\_locations;

ALTER TABLE team

DROP CONSTRAINT pk\_team;

ALTER TABLE festival\_nature

DROP CONSTRAINT pk\_festival\_nature;

ALTER TABLE festivals

DROP CONSTRAINT pk\_festivals;

ALTER TABLE festival\_staff

DROP CONSTRAINT pk\_festival\_staff;

ALTER TABLE festivals

DROP CONSTRAINT ck\_festival\_name;

ALTER TABLE staff

DROP CONSTRAINT pk\_staff;

--DROPPING TABLE

DROP TABLE LOCATIONS;

DROP TABLE FESTIVAL\_NATURE;

DROP TABLE FESTIVALS;

DROP TABLE STAFF;

DROP TABLE FESTIVAL\_STAFF;

DROP TABLE team;

--DROPPING TYPE

DROP TYPE social\_media\_table\_type;

DROP TYPE contact\_varray\_type;

DROP TYPE contact\_type;

DROP TABLE addresses;

DROP TYPE social\_media\_type;

DROP TYPE address\_type;

DROP TYPE team\_varray\_type;

DROP TABLE hr\_event;

--DROPPING SEQUENCES

DROP SEQUENCE seq\_locations;

DROP SEQUENCE seq\_festival\_nature;

DROP SEQUENCE seq\_staff;

PURGE RECYCLEBIN;

# REFERENCES

1. Docs.oracle.com. (2019). *Handling PL/SQL Errors*. [online] Available at: https://docs.oracle.com/cd/A97630\_01/appdev.920/a96624/07\_errs.htm [Accessed 18 April 2019].
2. Docs.oracle.com. (2019). *PL/SQL Packages*. [online] Available at: https://docs.oracle.com/database/121/LNPLS/packages.htm [Accessed 19 April 2019].
3. 9 Using Triggers. 2019. 9 Using Triggers. [ONLINE] Available at: https://docs.oracle.com/cd/B13789\_01/appdev.101/b10795/adfns\_tr.htm. [Accessed 20 April 2019].
4. CREATE TRIGGER. 2019. CREATE TRIGGER. [ONLINE] Available at: https://docs.oracle.com/cd/B19306\_01/server.102/b14200/statements\_7004.htm. [Accessed 20 April 2019]
5. TechOnTheNet. 2019. Oracle / PLSQL: Joins. [ONLINE] Available at: https://www.techonthenet.com/oracle/joins.php. [Accessed 17 April 2019].