BSc Computing
CSY2038 Databases 2

Date for Submission:	Sunday 5th May 2019

Group: 10	**UN Id**
Dipen Maharjan Tsering Khando Lama Tenzin Dhundup Sherpa Sarina Acharya	18406500 18406499 18406552 18406480

Table of Contents

1 Design

1.1 Entity Relationship Diagram (ERD)

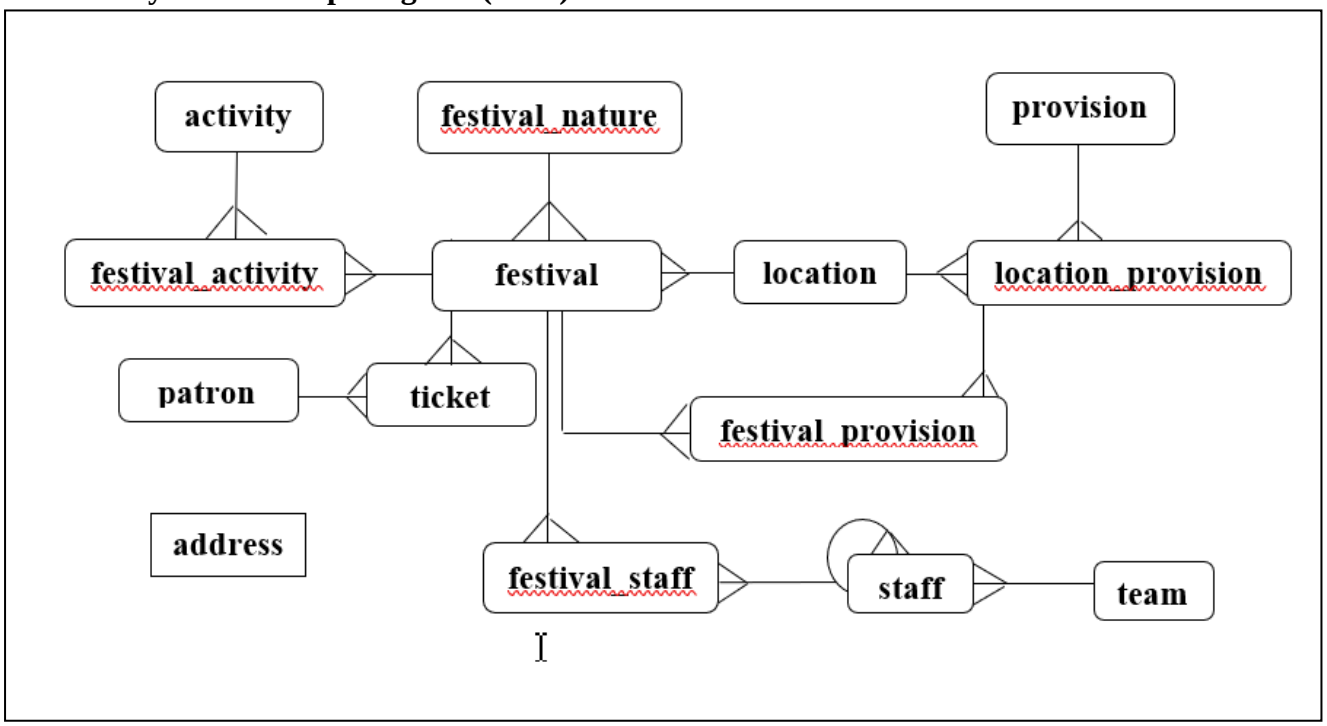


Fig 1.1: Given ERD

1.2 Chosen Entity Relationship Diagram (ERD)

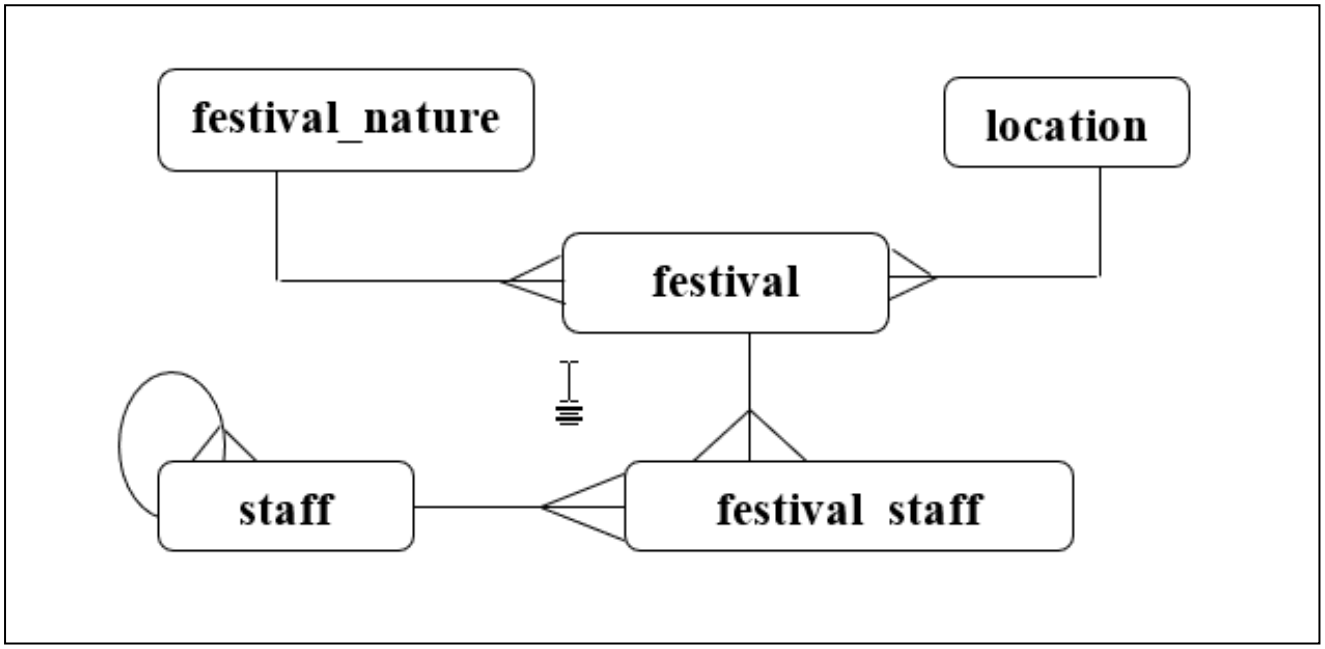


Fig 1.2: Chosen ERD

1.3 Schema

The following figure shows the schema from the chosen ERD given in the figure 1.2 :

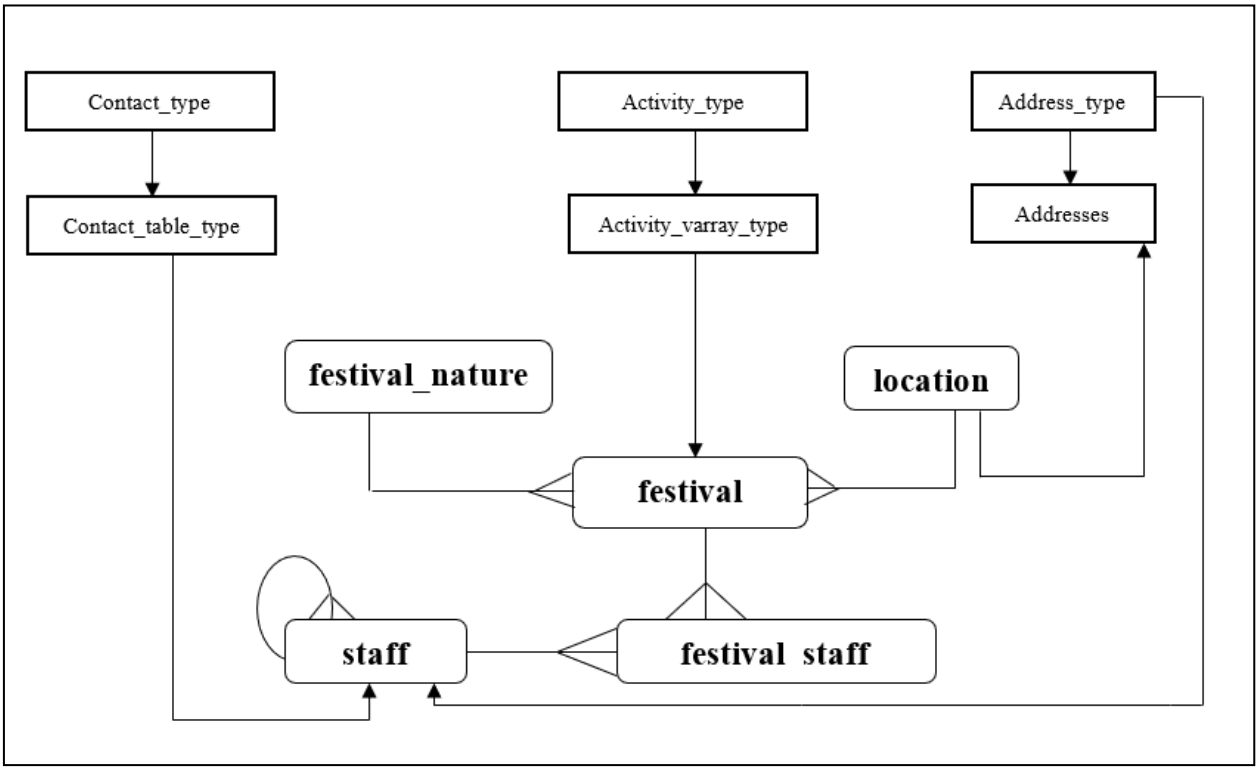


Fig 1.3: Schema

1.4 Table Specification

User-defined Object Types

1.4.1 Address_type

Table 1.1: address_type

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
street	VARCHAR2(25)			
city	VARCHAR2(25)			
country	VARCHAR2(25)			

1.4.2 Contact_type

Table 1.2: contact_type

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
medium	VARCHAR2(25)			
contact_number	VARCHAR2(25)			

1.4.3 Activity_type

Table 1.3: activity_type

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
activity_name	VARCHAR2(25)			
type	VARCHAR2(25)			

Relational Tables

1.4.4 Festival_natures

Table 1.4: festival_natures table

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
festival_id	NUMBER (7)	pk_festival_natures		seq_festival_natures
nature_name	VARCHAR2(25)	NOT NULL		

1.4.5 Locations

Table 1.5: locations table

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
location _no	NUMBER (7)	pk_locations		seq_locations
address	REF address_type			

1.4.6 Festivals

Table 1.6: festivals table

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
location _no	NUMBER (7)	fk_f_locations, pk_festivals, NOT NULL		
festival_id	VARCHAR2(20)	fk_f_festival_natures, pk_festivals, NOT NULL		
festival_name	VARCHAR2(20)	UPPER, NOT NULL		
activities	activity_varray_type			
festival_start_date	DATE			
festival_end_date	DATE			

1.4.7 Staff

Table 1.7: staff table

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
staff_id	NUMBER (7)	pk_staff		seq_staff
reports_to	NUMBER (7)	fk_s_staff		
firstname	VARCHAR2(20)	UPPER		
lastname	VARCHAR2(20)	UPPER		
address	address_type			
contact	contact_table_type	NOT NULL		
staff_gender	CHAR (1)	CHECK IN('M','F','O')	M	
staff_email	VARCHAR2(35)	UNIQUE, NOT NULL		
staff_employed_date	DATE	NOT NULL	SYSDATE	
username	VARCHAR2			
password	VARCHAR2			
salary	NUMBER (10,2)	NOT NULL		
dob	DATE	NOT NULL		

1.4.8 Festival_staff

Table 1.8: festival_staff table

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
staff_id	NUMBER (7)	fk_fs_staff, pk_festival_staff		
location _no,	NUMBER (7)	fk_fs_festivals, pk_festival_staff		
festival_id	NUMBER (7)	fk_fs_festivals, pk_festival_staff		

1.4.9 Activity_logs

The team created a new relational table into the database with attributes, "user_logged" and "date_time" which would record all the users' name who'd log on to the database. Following is the table specification of the table:

Table 1.9: Activity_logs table

ATTRIBUTES	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
user_logged	NUMBER (30)			
date_time	NUMBER (30)			

2 Automation Strategy

The following section contains the well-constructed procedures, functions, triggers, cursors containing in-built functions, loops.

Lists of procedures

1 proc_add_addressess
2 proc_delete_addressess_street
3 proc_delete_addressess_city
4 proc_delete_addressess_state
5 proc_delete_addressess_country
6 proc_add_location
7 proc_delete_location
8 proc_add_festival_nature
9 proc_delete_festival_nature
10 proc_add_festival
11 proc_delete_festival
12 proc_add_staff
13 proc_delete_staff
14 proc_add_fetival_staff
15 proc_delete_festival_staff

Lists of functions

1 func_generate_age_staff
2 func_duration_of_festival
3 func_generate_staff_username
4 func_generate_staff_password

Lists of triggers

1 trig_limit_login
2 trig_check_festival_dates
3 trig_check_insert_update_del
4 trig_stop_pk_update
5 trig_stop_pk_update_loc
6 trig_stop_pk_update_staff
7 trig_logon

List of cursors

1 proc_imp_cursor
2 proc_exp_cursor
3 proc_ckSal

The following section contains the functions, procedures, triggers, and cursors syntax with their expected parameters, description, and return type (datatype).

2.1 Functions

A function can be used as a part of SQL expression i.e. we can use them with select/update/merge commands. One most important characteristic of a function is that unlike procedures, it must return a value (GeeksforGeeks, 2019)

Table 2.1 Functions

SN	Functions	Parameters	Description	Return Type

1.	--FUNCTION
	func_count_staff
	CREATE OR REPLACE FUNCTION
	func_count_staff RETURN NUMBER AS
	vn_count NUMBER(3);
	BEGIN
	SELECT COUNT(staff_id)
	INTO vn_count
	FROM staff;
	RETURN vn_count;
	END func_count_staff;
	/
	--PROCEDURE
	CREATE OR REPLACE PROCEDURE
	proc_count_staff AS
	vn_count NUMBER(3);
	BEGIN
	vn_count := func_count_staff;
	DBMS_OUTPUT.PUT_LINE('THERE ARE '||vn_count||' STAFF RECORDS.');
	END proc_count_staff;
	/
			This function is created to count the number of staff in the database	NUMBER

2.	--FUNCTION
	func_generate_age_staff
	CREATE OR REPLACE FUNCTION
	func_generate_age_staff(in_date_of_birth DATE) RETURN NUMBER IS
	vd_dob DATE;
	vn_calc_age NUMBER(3);
	BEGIN
	vn_calc_age := FLOOR(MONTHS_BETWEEN(SYSDATE, in_date_of_birth)/12);
	RETURN vn_calc_age;
	END func_generate_age_staff;
	/
	--PROCEDURE
	CREATE OR REPLACE PROCEDURE
	proc_generate_age_staff(in_id staff.staff_id%TYPE)IS
	vn_calc_age NUMBER(3);
	vc_firstname staff.firstname%TYPE;
	vd_dob DATE;
		in_date_of_birth	This function is created to calculate the age of staff whose id is provided as a parameter. It displays the name of the staff along with his age.	NUMBER

	BEGIN
	SELECT firstname, dob INTO vc_firstname, vd_dob FROM staff WHERE staff_id= in_id;
	vn_calc_age := func_generate_age_staff(vd_dob);
	DBMS_OUTPUT.PUT_LINE('AGE OF ' || vc_firstname ||' IS ' || vn_calc_age|| ' YEARS OLD.');
	END proc_generate_age_staff;
	/			

3.	--**FUNCTION func_duration_of_festival**
	CREATE OR REPLACE FUNCTION func_duration_of_festival(in_start_date DATE, in_end_date DATE) RETURN NUMBER IS
	vn_calc_duration NUMBER(4);
	BEGIN
	vn_calc_duration:= in_end_date - in_start_date;
	RETURN vn_calc_duration;
	END func_duration_of_festival;
	/
	--**PROCEDURE**
	CREATE OR REPLACE PROCEDURE proc_duration_of_festival(in_fes_name festivals.festival_name%TYPE)IS
	vn_duration NUMBER(3);
	vd_start DATE;
	vd_end DATE;
	BEGIN
	SELECT festival_start_date,festival_end_date INTO vd_start,vd_end FROM festivals WHERE festival_name LIKE in_fes_name||'%';
	--SELECT festival_end_date INTO vd_end FROM festivals WHERE festival_name = in_fes_name;
	vn_duration := func_duration_of_festival(vd_start, vd_end);
	DBMS_OUTPUT.PUT_LINE('THE DURATION OF FESTIVAL IS ' || vn_duration || ' DAYS.');
	END proc_duration_of_festival;
	/
		in_start_date, in_end_date	This function is created to calculate the duration of festival by passing festival start date and end date.	NUMBER

4.	--**FUNCTION func_generate_staff_username** CREATE OR REPLACE FUNCTION func_generate_staff_username(in_staff_id staff.staff_id%TYPE) RETURN VARCHAR2 IS	in_staff_id	Generates the username of the staff whose id Is provided in parameter. It creates the	VARCHAR2

	vc_username VARCHAR2(20);
	
	BEGIN
	SELECT
	CONCAT(SUBSTR(firstname,1,2),
	SUBSTR(lastname,1,2))
	INTO vc_username
	FROM staff
	WHERE staff_id = in_staff_id;
	RETURN vc_username;
	END func_generate_staff_username;
	/
	--**PROCEDURE**
	CREATE OR REPLACE PROCEDURE
	proc_update_username_password(in_staff_id
	staff.staff_id%TYPE) IS
	vc_username VARCHAR2(20);
	vc_password VARCHAR2(20);
	BEGIN
	vc_username :=
	func_generate_staff_username(in_staff_id);
	vc_password
	:=func_generate_staff_password(in_staff_id);
	
	UPDATE staff SET
	username = vc_username,
	password=vc_password
	WHERE staff_id = in_staff_id;
	END proc_update_username_password;
	/
			username automatically by taking the staff first names first two letter and last names first two letter.
			For e.g. firstname = 'DAVID', lastname = 'DOE' then,
			Username will be ' DADO'	

5.	--**FUNCTION**
	-- **func_generate_staff_password**
	CREATE OR REPLACE FUNCTION
	func_generate_staff_password(in_staff_id
	staff.staff_id%TYPE) RETURN VARCHAR2
	IS
	vc_password VARCHAR2(20);
	BEGIN
	SELECT
	CONCAT(CONCAT(UPPER(SUBSTR(userna
	me,1,4)),REPLACE(SUBSTR(dob,1,5),'-')),
	SUBSTR(firstname,1,4))
	INTO vc_password
	FROM staff
	WHERE staff_id = in_staff_id;
	RETURN vc_password;
	END func_generate_staff_password;
	/
	--**PROCEDURE**
		in_staff_id	Generate the password of the staff whose id Is provided in parameter. It creates the password automatically in a following ways:
			First four letter of staff username column in UPPER case, with
			First five letter from dob column of staff, and
			First four letter from staff firstname column.
			
			For e.g. we have,
				VARCHAR2

	CREATE OR REPLACE PROCEDURE proc_update_username_password(in_staff_id staff.staff_id%TYPE) IS
	vc_username VARCHAR2(20);
	vc_password VARCHAR2(20);
	BEGIN
	vc_username := func_generate_staff_username(in_staff_id);
	vc_password :=func_generate_staff_password(in_staff_id);
	
	UPDATE staff SET
	username = vc_username, password=vc_password
	WHERE staff_id = in_staff_id;
	
	END proc_update_username_password;
	/
			firstname = 'DAVID', dob = '01-JAN-2018', username = 'DADO' then, Password will be "DADO01JANDAVI"	

2.2 Procedures

As PL/SQL is a block orientated language, it contains two types of blockS i.e. Unnamed block and named block. They contain three parts.

1. DECLARE
2. BEGIN
3. EXCEPTION

Procedures are the group of PL/SQL statements that can be called by name (ORACLE, 2019)

Table 2.2Procedures

SN	Procedure Syntax	Parameters	Description	Return Type
1.	**proc_add_addresses:** CREATE OR REPLACE PROCEDURE proc_add_addresses(in_street addresses.street%TYPE, in_city addresses.city%TYPE, in_state addresses.state%TYPE, in_country addresses.country%TYPE) IS BEGIN INSERT INTO addresses (street, city, state, country) VALUES (in_street, in_city, in_state, in_country); END proc_add_addresses; / SHOW ERRORS	in_street, in_city, in_state, in_country	Inserts into the addresses table with parameters of street, city, state, country	
2.	**proc_delete_addresses_street:** CREATE OR REPLACE PROCEDURE proc_delete_addresses_street(in_street addresses.street%TYPE) IS BEGIN DELETE FROM addresses WHERE street = in_street; END proc_delete_addresses_street; / SHOW ERRORS	in_street	Deletes the row of addresses table whose street is equal to the value in the parameter while executing the procedure	
3.	**proc_delete_addresses_city:** CREATE OR REPLACE PROCEDURE proc_delete_addresses_city(in_city addresses.city%TYPE) IS BEGIN DELETE FROM addresses WHERE city = in_city; END proc_delete_addresses_city; / SHOW ERRORS	in_city	Deletes the row of addresses table whose city is equal to the value in the parameter while executing the procedure	

4.	**proc_delete_addresses_state:** CREATE OR REPLACE PROCEDURE proc_delete_addresses_state (in_state addresses.state%TYPE) IS BEGIN DELETE FROM addresses WHERE state = in_state; END proc_delete_addresses_state; / SHOW ERRORS	in_state	Deletes the row of addresses table whose state is equal to the value in the parameter while executing the procedure	
5.	**proc_delete_addresses_country:** CREATE OR REPLACE PROCEDURE proc_delete_addresses_country(in_country addresses.country%TYPE) IS BEGIN DELETE FROM addresses WHERE country = in_country; END proc_delete_addresses_country; / SHOW ERRORS	in_country	Deletes the row of addresses table whose country is equal to the value in the parameter while executing the procedure	
6.	**proc_add_location:** CREATE OR REPLACE PROCEDURE proc_add_location (in_city addresses.city%TYPE) IS BEGIN INSERT INTO locations (location_no, address) SELECT seq_locations.NEXTVAL, REF(a) FROM addresses a WHERE city = in_city; END proc_add_location; /	in_city	Inserts into the locations table whose city is equal to the parameter value while executing the procedure	
7.	**proc_delete_location:** CREATE OR REPLACE PROCEDURE proc_delete_location (in_location_no locations.location_no%TYPE) IS BEGIN DELETE FROM locations WHERE location_no = in_location_no; END proc_delete_location; / SHOW ERRORS	in_location_no	Deletes the row of addresses table whose location is equal to the value in the parameter while executing the procedure	
8.	**proc_add_festival_natures:** CREATE OR REPLACE PROCEDURE proc_add_festival_natures (in_nature_name festival_natures.nature_name%TYPE) IS BEGIN	in_nature_name	Inserts the values into the festival_natures table with parameters value nature_name	

	INSERT INTO festival_natures(festival_nature_id,nature_name) VALUES (seq_festival_natures.NEXTVAL, in_nature_name); END proc_add_festival_natures; / SHOW ERRORS			
9.	**proc_delete_festival_nature:** CREATE OR REPLACE PROCEDURE proc_delete_festival_nature (in_festival_nature_id festival_natures.festival_nature_id%TYPE) IS BEGIN DELETE FROM festival_natures WHERE festival_nature_id = in_festival_nature_id; END proc_delete_festival_nature; / SHOW ERRORS	in_festival_nature_id	Deletes the row of festival_nature table whose festival_nature_id is equal to the value in the parameter while executing the procedure	
10	**proc_add_festival:** CREATE OR REPLACE PROCEDURE proc_add_festival (in_location_no festivals.location_no%TYPE, in_festival_nature_id festivals.festival_nature_id%TYPE, in_festival_name festivals.festival_name%TYPE, in_activities festivals.activities%TYPE, in_festival_start_date festivals.festival_start_date%TYPE, in_festival_end_date festivals.festival_end_date%TYPE) IS BEGIN INSERT INTO festivals (location_no, festival_nature_id, festival_name, activities, festival_start_date, festival_end_date) VALUES (in_location_no, in_festival_nature_id, in_festival_name, in_activities, in_festival_start_date, in_festival_end_date); END proc_add_festival; / SHOW ERRORS	in_location_no, in_festival_nature_id, in_festival_name, in_activities, in_festival_start_date, in_festival_end_date	Inserts into the festivals table with parameter values location_no, festival_nature_id, festival_name, acivities, festival_start_date and festival_end_date	
11.	**proc_delete_festival:** CREATE OR REPLACE PROCEDURE proc_delete_festival (in_festival_nature_id festivals.festival_nature_id%TYPE) IS	in_festival_nature_id	Deletes the row of festivals table whose festival_nature_id is equal to the value in the	

	BEGIN DELETE FROM festivals WHERE festival_nature_id=in_festival_nature_id; END proc_delete_festival; / SHOW ERRORS		parameter while executing the procedure	
12.	**proc_add_staff:** CREATE OR REPLACE PROCEDURE proc_add_staff (in_firstname staff.firstname%TYPE, in_lastname staff.lastname%TYPE, in_address staff.address%TYPE, in_contact staff.contact%TYPE, in_staff_gender staff.staff_gender%TYPE, in_staff_email staff.staff_email%TYPE, in_staff_employed_date staff.staff_employed_date%TYPE, in_salary staff.salary%TYPE, in_dob staff.dob%TYPE)IS BEGIN INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender, staff_email, staff_employed_date,salary, dob) VALUES (seq_staff.NEXTVAL,in_firstname, in_lastname, in_address, in_contact, in_staff_gender, in_staff_email, in_staff_employed_date, in_salary, in_dob); END proc_add_staff; / SHOW ERRORS	in_firstname, in_lastname, in_address, in_contact, in_staff_gender, in_staff_email, in_staff_employed_date, in_salary	Inserts into the staff table with parameter values firstname, lastname, address, contact, staff_gender, staff_email, staff_employed_date, staff_salary and dob	
13.	**proc_delete_staff:** CREATE OR REPLACE PROCEDURE proc_delete_staff(in_staff_id staff.staff_id%TYPE) IS BEGIN DELETE FROM staff WHERE staff_id = in_staff_id; END proc_delete_staff; / SHOW ERRORS	in_staff_id	Deletes the row of staff table whose staff_id is equal to the value in the parameter while executing the procedure	
14.	**proc_add_festival_staff:** CREATE OR REPLACE PROCEDURE proc_add_festival_staff (in_staff_id festival_staff.staff_id%TYPE, in_location_no festival_staff.location_no%TYPE, in_festival_nature_id festival_staff.festival_nature_id%TYPE) IS BEGIN	in_staff_id, in_location_no, in_festival_nature_id	Inserts into the festival_staff table with parameter values staff_id, location_no, festival_nature_id	

	INSERT INTO festival_staff (staff_id, location_no, festival_nature_id) VALUES (in_staff_id,in_location_no, in_festival_nature_id); END proc_add_fetival_staff; / SHOW ERRORS			
15.	**proc_delete_festival_staff:** CREATE OR REPLACE PROCEDURE proc_delete_festival_staff(in_staff_id festival_staff.staff_id%TYPE) IS BEGIN DELETE FROM festival_staff WHERE staff_id = in_staff_id; END proc_delete_festival_staff; / SHOW ERRORS	in_staff_id	Deletes the row of festival_staff table whose staff_id is equal to the value in the parameter while executing the procedure	

2.3 Triggers

Trigger is an automatically generating code in response to certain events in the database table. It is used in database to give a certain message to the user if any wrong data is inserted, updated or deleted .Triggers can be of database level or schema level or DML.

Table 2.3Triggers

SN	TRIGGERS	Parameters	DESCRPTION

1.	CREATE OR REPLACE TRIGGER trig_limit_login
	AFTER LOGON ON group10.SCHEMA
	DECLARE
	vn_hour :=NUMBER(2);
	
	BEGIN
	SELECT to_char(SYSDATE,'HH24') INTO vn_hour
	FROM DUAL;
	IF vn_hour NOT BETWEEN 8 AND 18
	THEN
	RAISE_APPLICATION_ERROR(-20001,'Not allowed to logon database before 8 and after 6 ');
	END IF;
	END;
	/
		No	Whenever the user tries to login to the system except the time limit defined by the trigger ,then a certain message
			is fired.

2.	CREATE OR REPLACE TRIGGER trig_check_festival_dates AFTER INSERT OR UPDATE ON festivals FOR EACH ROW WHEN (NEW.festival_end_date < NEW.festival_start_date) BEGIN RAISE_APPLICATION_ERROR(-20004, 'FESTIVAL END DATE IS LOWER THAN START DATE!!!'); END trig_check_festival_dates; /	No	The trigger is fired when the user enters the end date lesser than the start date in the festivals table.

3	CREATE OR REPLACE TRIGGER trig_check_age_del
	AFTER INSERT OR UPDATE OR DELETE OF dob
	ON staff
	FOR EACH ROW
	DECLARE
	vn_age NUMBER(3);
	vd_today DATE;
	
	BEGIN
	SELECT SYSDATE INTO vd_today
	FROM DUAL;
	
	vn_age:=FLOOR(MONTHS_BETWEEN(sysdate,
	:NEW.dob)/12);
	IF INSERTING OR UPDATING THEN
		No	Is fired when the age below 18 and above 60 is inserted to the table staffs.

	IF (vn_age<18) OR (vn_age>60) OR :NEW.dob>vd_today THEN
	
	RAISE_APPLICATION_ERROR(-20002,'THE AGE MUST BE BETWEEN 18 - 60 YEARS');
	ELSE
	
	DBMS_OUTPUT.PUT_LINE('SUCCESSFUL');
	END IF;
	ELSE
	DBMS_OUTPUT.PUT_LINE(' YOU ARE DELETING '|| :OLD.firstname);
	END IF;
	END trig_check_age_del;
	/
	SHOW ERRORS		

4	CREATE OR REPLACE TRIGGER trig_stop_pk_update AFTER UPDATE OF festival_nature_id ON festival_natures BEGIN RAISE_APPLICATION_ERROR(-20003,'You cannot update the primary key'); END trig_stop_pk_update; /	No	This triggers stops from updating primary key in the table'festival_natures'.
5	CREATE OR REPLACE TRIGGER trig_stop_pk_update_loc AFTER UPDATE OF location_no ON locations BEGIN RAISE_APPLICATION_ERROR(-20003,'You cannot update the primary key'); END trig_stop_pk_update_loc; /	No	This triggers stops from updating primary key in the table'locations'.
6	CREATE OR REPLACE TRIGGER trig_stop_pk_update_staff AFTER UPDATE OF staff_id ON staff BEGIN RAISE_APPLICATION_ERROR(-20003,'You cannot update the primary key'); END trig_stop_pk_update_staff; /	No	This triggers stops from updating primary key in the table'staff'.

7.	CREATE OR REPLACE TRIGGER trig_logon
	AFTER LOGON
	ON DATABASE
	BEGIN
	INSERT INTO activity_logs(user_logged, date_time)
	VALUES (USER, SYSDATE);
	END;
	/
		No	This trigger inserts date and time when user logs on the system into the 'activity_logs' table.

2.4 Cursors

A cursor is a temporary work area created in system memory when a SQL statement is executed. A cursor is a set of rows together with a pointer that identifies a current row.

There are the following two types of Cursors:

1. Implicit Cursor
2. Explicit Cursor (C#Corner, 2019)

Table 2.4Cursors

SN	Cursor	Parameters	Description

1	**proc_imp_cursor**
	CREATE OR REPLACE PROCEDURE proc_imp_cursor(in_fname staff.firstname%TYPE,in_lname staff.lastname%TYPE, in_nname staff.firstname%TYPE) IS
	vc_fname VARCHAR2(15);
	vc_lname VARCHAR2(15);
	vc_newname VARCHAR2(15);
	exp1 EXCEPTION;
	BEGIN
	vc_newname := in_nname;
	vc_fname := in_fname;
	vc_lname := in_lname;
	IF in_nname is NULL THEN RAISE exp1;
	UPDATE staff SET firstname=in_nname
	WHERE vc_fname =in_fname AND lastname = in_lname;
	IF SQL%FOUND THEN
	DBMS_OUTPUT.PUT_LINE(vc_fname || ' updated ');
	ELSE
	DBMS_OUTPUT.PUT_LINE('User does not exist');
	END IF;
	END IF;
	EXCEPTION
	WHEN exp1 THEN
	DBMS_OUTPUT.PUT_LINE('User does not exist');
	END proc_imp_cursor;
	/
		in_fname, in_lname, in_nname	Here, the cursor is used to update the row's first name when the given first name and last name by user is found in the database and is updated with the new name given by user. If the names given by user are not found in the database and if the new name to be given is left empty then, system displays the message "User does not exist".

2	**proc_exp_cursor** CREATE OR REPLACE PROCEDURE proc_exp_cursor(in_salary staff.salary%TYPE)IS vn_rowcount NUMBER(2):=0; CURSOR cur_salary IS SELECT firstname,lastname,salary FROM staff WHERE salary>in_salary;	in_salary	The code to the left column is an explicit cursor which displays the first name, last name, and salary of the staff if the salary of the staff in the database is greater than the salary

	BEGIN
	DBMS_OUTPUT.PUT_LINE('S.No'||' '||'First Name'||' '||'Last Name'||' '||'Salary');
	FOR rec_cur_salary IN cur_salary LOOP
	IF rec_cur_salary.salary > in_salary THEN
	DBMS_OUTPUT.PUT_LINE(cur_salary%ROWCOUNT||' '||rec_cur_salary.firstname||' '|| rec_cur_salary.lastname||'
	'||rec_cur_salary.salary);
	vn_rowcount := cur_salary%ROWCOUNT;
	END IF;
	END LOOP;
	DBMS_OUTPUT.PUT_LINE('THERE ARE '|| vn_rowcount ||' STAFF ');
	END proc_exp_cursor;
	/
			provided as a parameter and counts the number of those staff.

3	**proc_ckSal**
	CREATE OR REPLACE PROCEDURE proc_ckSal(in_salary staff.salary%TYPE)IS
	CURSOR cur_salary IS
	SELECT staff_id,firstname,lastname,salary
	FROM staff
	WHERE salary>in_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('SALARY RANGE TOO HIGH');
	ELSE
	WHILE cur_salary%FOUND LOOP
	DBMS_OUTPUT.PUT_LINE(rec_cur_salary.staff_id ||' '|| rec_cur_salary.firstname||' ' || rec_cur_salary.lastname||' ' || rec_cur_salary.salary);
	FETCH cur_salary INTO rec_cur_salary;
	END LOOP;
	END IF;
	CLOSE cur_salary;
	END proc_ckSal;
	/
		in_salary	Checks the salary provided as parameter and if the salary is given higher than the salary in the database than it displays the message “Salary range too high”, whereas if the salary is given within a range of salary as in database than it displays first name, last name, and salary of those staff.

3 Testing

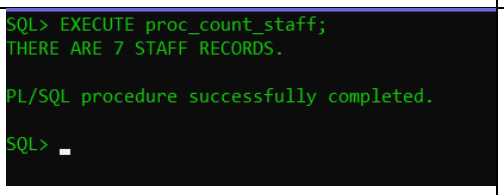
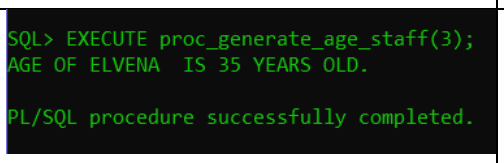
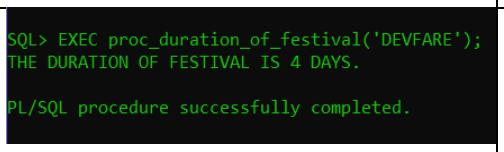
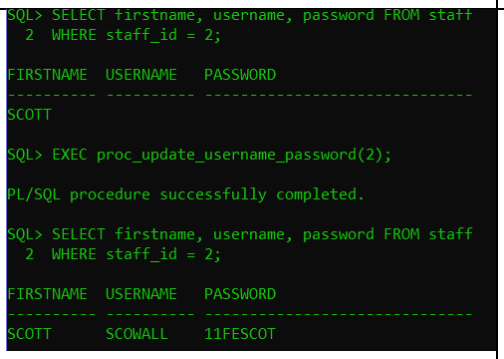
3.1 Test plan

The group made triggers based on schema level, database level and DML. For the schema level the group made the logon check triggers to limit the time for the user to login to the system. Similarly, in database level the trigger is made to show the users and the date when they logged in. Since, the users cannot update the primary keys , the group decided to check the update of the primary keys on tables staff, locations and festival_natures. To limit the age (not below 18 or above 60) of the staff the trigger 'trig_check_age_del' was implemented.

3.2 Test case

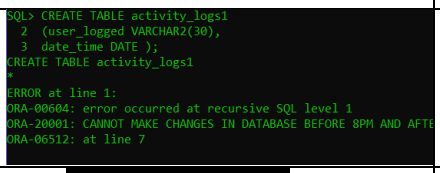
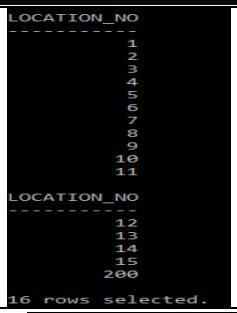
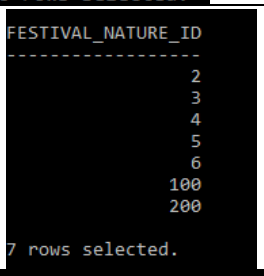
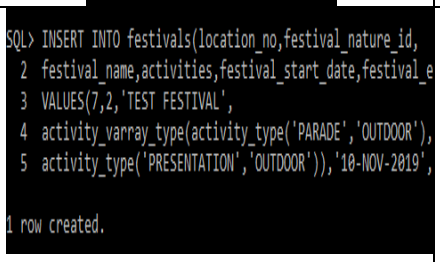
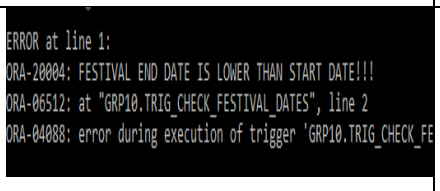
3.2.1 Functions testing

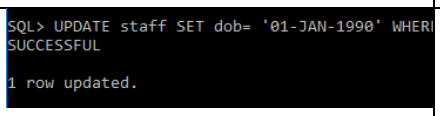
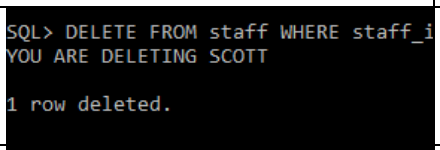
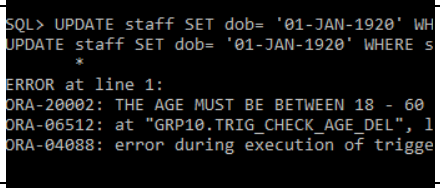
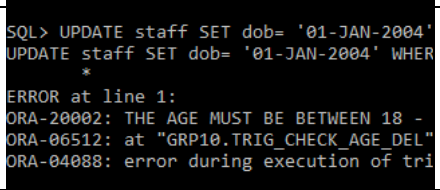
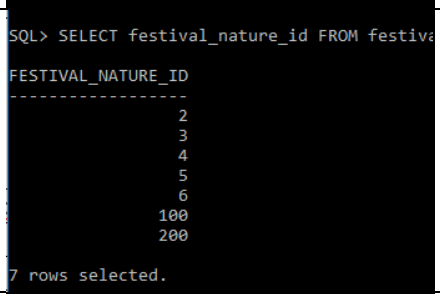
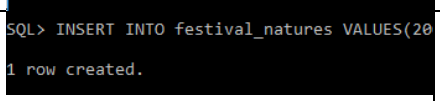
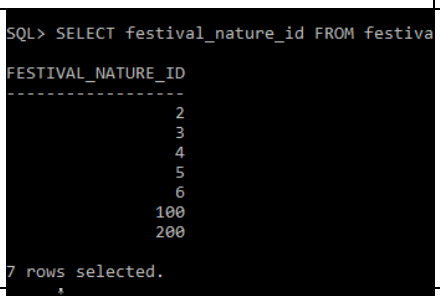
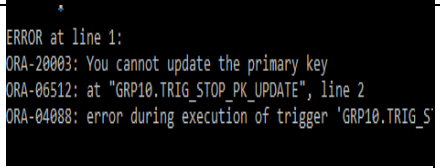
Table 3.1Testing of functions

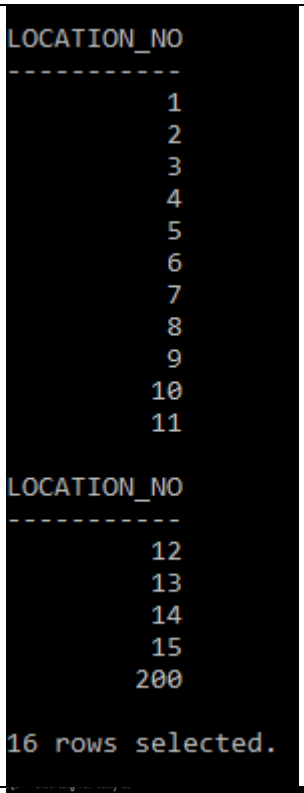
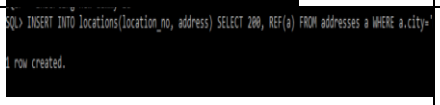
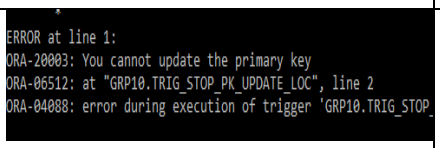
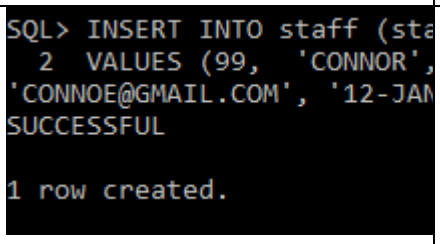
SN	Function Name	Test	Expected Output	Actual Output	Action
1	func_count_staff	EXECUTE proc_count_staff;	THERE ARE 7 STAFF RECORDS.		
2	func_generate_age_staff	EXECUTE proc_generate_age_staff(3);	AGE OF ELVENA IS 35 YEARS OLD.		
3	func_duration_of_festival	EXEC proc_duration_of_festival('DEVFARE');	THE DURATION OF FESTIVAL IS 4 DAYS.		
4	proc_exp_cursor	EXEC proc_exp_cursor(99000);	THERE ARE 0 STAFF.		
5	func_generate_staff_username func_generate_staff_password	SELECT firstname, username, password FROM staff WHERE staff_id = 2; proc_update_username_password SELECT firstname, username, password FROM staff WHERE staff_id = 2;	Shows table without username and password first. Then, after the execution of procedure username and password are showed.		

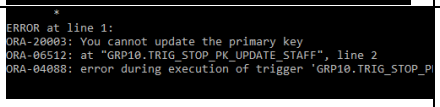
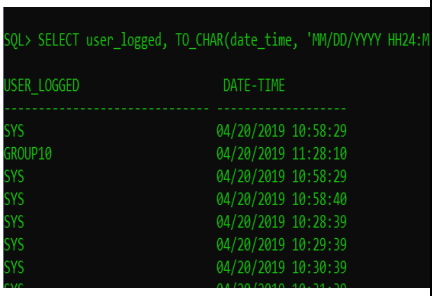
3.2.2 Triggers testing

Table 3.2 Testing of Triggers

SN	Trigger Name	Test	Expected Results	Actual Results	Action
1.	trig_limit_login	Connect group10	CANNOT MAKE CHANGES IN DATABASE BEFORE 8PM AND AFTER 6PM.		
2.	trig_check_festival_dates	SELECT location_no FROM locations;	16 rows returned		
		SELECT festival_nature_id FROM festival_natures;	7 rows selected		
		INSERT INTO festivals(location_no, festival_nature_id, festival_name,activities,festival_start_date, festival_end_date) VALUES(7,2,'TEST FESTIVAL', activity_varray_type(activity_type('PARADE','OUTDOOR'), activity_type('PRESENTATION','OUTDOOR')),'10-NOV-2019','20-NOV-2019');	1 row created		
		INSERT INTO festivals(location_no, festival_nature_id, festival_name, activities, festival_start_date,festival_end_date) VALUES(8,2,'TEST FESTIVAL',	FESTIVAL END DATE IS LOWER THAN START DATE!!!		

		activity_varray_type(activity_type('PARADE','OUTDOOR'), activity_type('PRESENTATION','OUTDOOR')), '30-NOV-2020','20-NOV-2019');			
3	trig_check_age_del	UPDATE staff SET dob= '01-JAN-1990' WHERE staff_id =1;	1 row updated		
		DELETE FROM staff WHERE staff_id=2;	1 row deleted		
		UPDATE staff SET dob= '01-JAN-1920' WHERE staff_id =1;	The Age must be between 18 - 60 years		
		UPDATE staff SET dob= '01-JAN-2004' WHERE staff_id =1;	The Age must be between 18 - 60 years		
4	trig_stop_pk_update	SELECT festival_nature_id FROM festival_natures;	7 rows selected		
		INSERT INTO festival_natures VALUES(200,'TEST');	1 row inserted		
		SELECT festival_nature_id FROM festival_natures;	7 rows selected		
		UPDATE festival_natures SET festival_nature_id=222 WHERE festival_nature_id=200;	You cannot update the primary key		

5	trig_stop_pk_update_loc	SELECT location_no FROM locations;	16 rows selected		
		INSERT INTO locations(location_no, address) SELECT 200, REF(a) FROM addresses a WHERE a.city='HARTFORD';	1 row created		
		UPDATE locations SET location_no =209 WHERE location_no=200;	You cannot update the primary key		
6	trig_stop_pk_update_staff	INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender, staff_email, staff_employed_date, salary, dob) VALUES (99, 'CONNOR', 'PAYNE', address_type('4478 TRAILS END ROAD','ALBANY','NEW YORK','USA'), contact_table_type(contact_type('MOBILE','+1 129-123-0099')), 'M', 'CONNOE@GMAIL.COM', '12-JAN-	1 row created		

		2018', '14000', '11-NOV-1982');			
		SELECT staff_id , firstname FROM staff;	8 rows selected		
		UPDATE staff SET staff_id=1000 WHERE staff_id=100;	You cannot update the primary key		
7	trig_logon	SELECT user_logged, TO_CHAR(date_time, 'MM/DD/YYYY HH24:MI:SS') AS " DATE-TIME" FROM activity_logs ;	Displays the logon table on database level		

Note:

Indicates Intentional Wrong Input

3.2.3 Cursor testing

Table 3.3: Testing of Cursor

SN	Cursor Name	Test	Expected Output	Actual Output	Action
1	proc_imp_cursor	EXEC proc_imp_cursor('DAVID','DOE','SHYAM');	DAVID UPDATED TO SHYAM	SQL> EXEC proc_imp_cursor('DAVID','DOE','SHYAM'); DAVID UPDATED PL/SQL procedure successfully completed.	
2	proc_imp_cursor	EXEC proc_imp_cursor('TENZIN','SHERPA','');	NEW NAME NOT INSERTED	SQL> EXEC proc_imp_cursor('TENZIN','SHERPA',''); NEW NAME NOT INSERTED PL/SQL procedure successfully completed.	
3	proc_exp_cursor	EXEC proc_exp_cursor(10000);	THERE ARE 3 STAFF	SQL> EXEC proc_exp_cursor(10000); S.No First Name Last Name Salary 1 SHYAM DOE 13000 2 MENAKA MORRISON 11000 3 CONNOR PAYNE 14000 THERE ARE 3 STAFF. PL/SQL procedure successfully completed.	
4	proc_exp_cursor	EXEC proc_exp_cursor(99000);	THERE ARE 0 STAFF.	SQL> EXEC proc_exp_cursor(99000); S.No First Name Last Name Salary THERE ARE 0 STAFF.	
5	proc_ckSal	EXEC proc_ckSal(6000);	SHOWS TABLE with 8 rows	SQL> EXEC proc_ckSal(6000); S.No First Name Last Name Salary 1 SHYAM DOE 13000 2 SCOTT WALLACE 10000 3 ELVENA JONES 8000 4 KEVIA PAYNE 9000 5 MENAKA MORRISON 11000 6 CONNOR PAYNE 14000 8 RAM SHRESTHA 10000	
6	proc_ckSal	EXEC proc_ckSal(100000);	SALARY RANGE TOO HIGH	SQL> EXEC proc_ckSal(100000); S.No First Name Last Name Salary SALARY RANGE TOO HIGH	

3.2.4 Procedures Testing

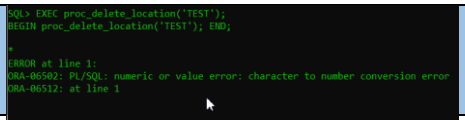
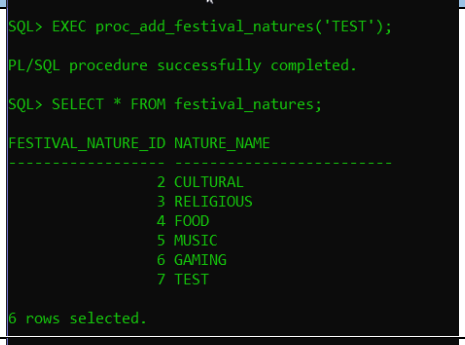
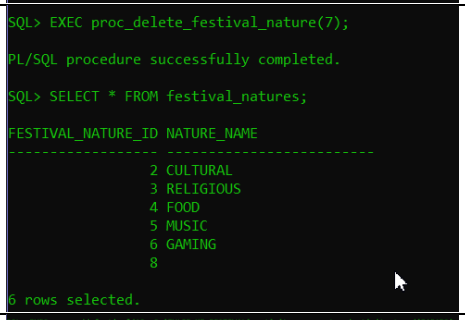
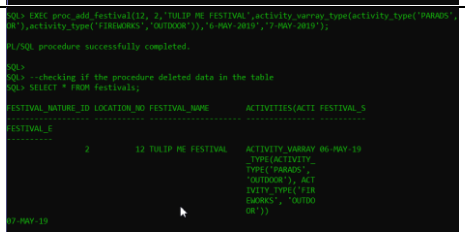
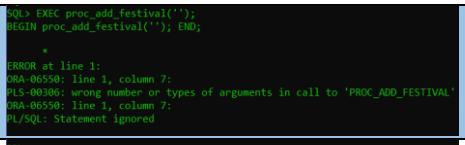
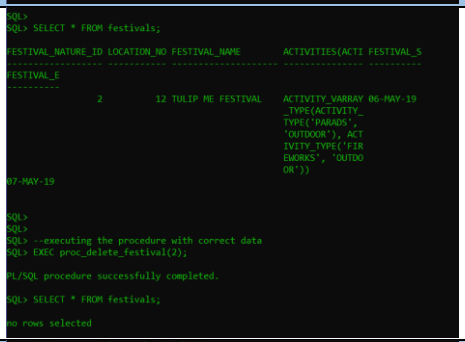
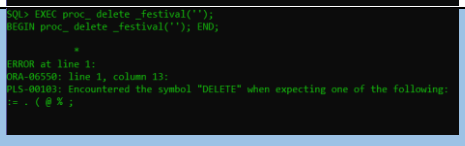
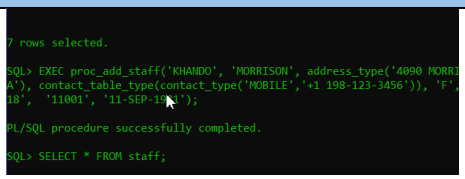
Table 3.4: Testing of Procedures

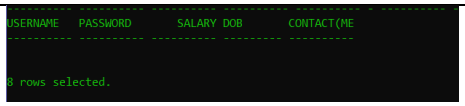
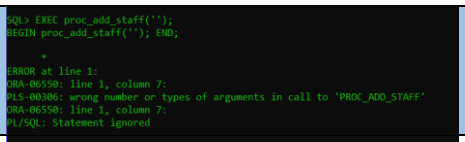
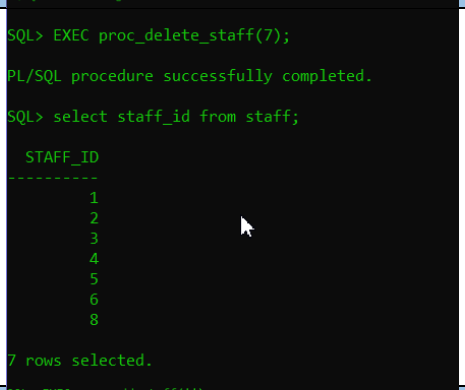
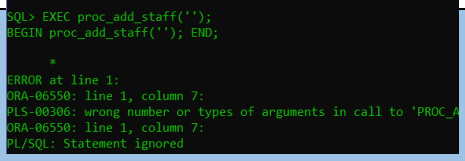
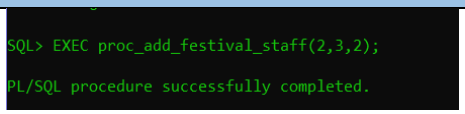
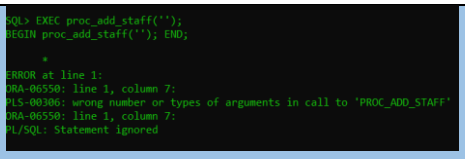
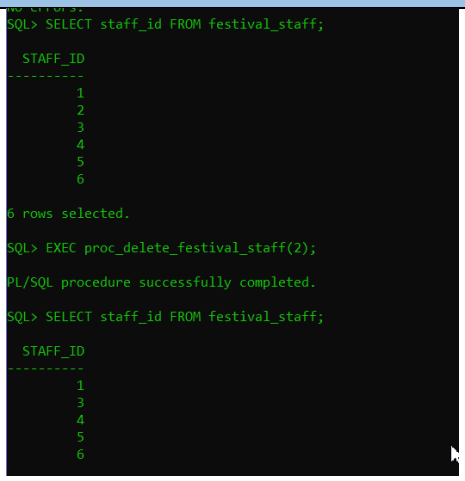
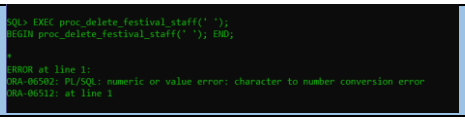
SN	Procedure Name	Test	Expected Results	Actual Results	Action

1.	Anonymous block	DECLARE
		vc_festival_name festivals.festival_name%TYPE;
		BEGIN
		SELECT festival_name INTO vc_festival_name
		FROM festivals WHERE festival_nature_id=4 AND location_no=1;
		DBMS_OUTPUT.PUT_LINE(vc_festival_name);
		EXCEPTION WHEN no_data_found THEN
		DBMS_OUTPUT.PUT_LINE('----------NO DATA FOUND----------');
		END;
		/
			CIRCLEVILLE PUMPKIN SHOW	CIRCLEVILLE PUMPKIN SHOW
				
				PL/SQL procedure successfully completed.	

2.	proc_add_addresses	EXEC proc_add_addresses('SOUTH SCIOTO','CIRCLEVILLE','OHIO','USA');	PL/SQL procedure successfully completed	SQL> EXEC proc_add_addresses('SOUTH SCIOTO','CIRCLEVILLE','OHIO','U PL/SQL procedure successfully completed.	
3.	proc_add_addresses	EXEC proc_add_addressess('CIRCLEVILLE','OHIO','USA');	Execution Error	SQL> EXEC proc_add_addressess('CIRCLEVILLE','OHIO','USA'); BEGIN proc_add_addressess('CIRCLEVILLE','OHIO','USA'); END; * ERROR at line 1: ORA-06550: line 1, column 7: PLS-00201: identifier 'PROC_ADD_ADDRESSESS' must be declar ORA-06550: line 1, column 7: PL/SQL: Statement ignored	
4.	proc_delete_addresses_street	EXEC proc_delete_addresses_street('SOUTH SCIOTO');	PL/SQL procedure successfully completed	SQL> EXEC proc_delete_addresses_street('SOUTH SCIOTO PL/SQL procedure successfully completed.	
5.	proc_delete_addresses_street	EXEC proc_delete_addresses_street(1);	Execution Error	SQL> EXEC proc_delete_addressess(1); BEGIN proc_delete_addressess(1); END; * ERROR at line 1: ORA-06550: line 1, column 7: PLS-00201: identifier 'PROC_DELETE_ADDRESSESS' must be decl ORA-06550: line 1, column 7: PL/SQL: Statement ignored	
6.	proc_delete_addresses_city	EXEC proc_delete_addresses_city('CIRCLEVILLE');	PL/SQL procedure successfully completed	SQL> EXEC proc_delete_addresses_city('CIRCLEVILLE PL/SQL procedure successfully completed.	
7.	proc_delete_addresses_city	EXEC proc_delete_addresses_city(01);	no changes	15 rows selected. SQL> EXEC proc_delete_addresses_city(01); PL/SQL procedure successfully completed. SQL> select * from addresses;	

				ARTFORD, CT 06103 T 15 rows selected. SQL>
8.	proc_delete_addresses_state	EXEC proc_delete_addresses_state('OHIO');	deleted	SQL> EXEC proc_delete_addresses_state('OHIO' PL/SQL procedure successfully completed. ARTFORD, CT 06103 T 13 rows selected.
9.	proc_delete_addresses_state	EXEC proc_delete_addresses_state(01);	no changes	15 rows selected. SQL> EXEC proc_delete_addresses_state(01); PL/SQL procedure successfully completed. SQL> select * from addresses; ARTFORD, CT 06103 T 15 rows selected. SQL>
10.	proc_delete_addresses_country	EXEC proc_delete_addresses_country ('USA');	deleted	SQL> EXEC proc_delete_addresses_country('USA PL/SQL procedure successfully completed.
11.	proc_delete_addresses_country	EXEC proc_delete_addresses_country (01);	no changes	13 rows selected. SQL> EXEC proc_delete_addresses_country(01); PL/SQL procedure successfully completed. SQL> select * from addresses; 13 rows selected. SQL>
12.	proc_add_location	EXEC proc_add_location('TEST');	location added	SQL> EXEC proc_add_location('TEST'); PL/SQL procedure successfully complete
13.	proc_add_location	EXEC proc_add_location(01); SELECT location_no, l.address.street street,l.address.city city, l.address.state state, l.address.country country FROM locations l;	no row added	15 rows selected. SQL> --no row added SQL> EXEC proc_add_location(01); PL/SQL procedure successfully completed. SQL> SELECT location_no, l.address.street street,l.address. 2 FROM locations l; LOCATION_NO STREET CITY STATE COUNTRY ---------- ---------- ---------- ---------- ---------- 14 3800 NAPOLEON LN IOWA CITY IOWA USA 15 100 COLUMBUS BLVD, H HARTFORD CONNECTICU USA ARTFORD, CT 06103 T 15 rows selected. SQL>
14.	proc_delete_location	EXEC proc_delete_location(15); SELECT location_no, l.address.street street,l.address.city city, l.address.state state, l.address.country country	deleted	PL/SQL procedure successfully completed. SQL> SELECT location_no from locations; LOCATION_NO ---------- 1 2 3 4 5 6 7 8 9 10 11 LOCATION_NO ---------- 12 13 14 15

		FROM locations l;			
15.	proc_delete_location	EXEC proc_delete_location('TEST');	Error expected		
16.	proc_add_festival_natures	EXEC proc_add_festival_natures('TEST'); SELECT * FROM festival_natures;	New row added		
17.	proc_delete_festival_nature	EXEC proc_delete_festival_nature(7); SELECT * FROM festival_natures;	Deleted		
18.	proc_add_festival	EXEC proc_add_festival(3, 2,'TULIP ME FESTIVAL',activity_varray_type(activity_type('PARADS','OUTDOOR'),activity_type('FIREWORKS','OUTDOOR')),'6-MAY-2019','7-MAY-2019');	New row added		
19.	proc_add_festival	EXEC proc_add_festival('');	Error expected		
20.	proc_delete_festival	EXEC proc_ delete _festival(2);	Row deleted		
21.	proc_delete_festival	EXEC proc_ delete _festival('');	Error expected		
22.	proc_add_staff	EXEC proc_add_staff('KHANDO', 'MORRISON', address_type('4090 MORRIS STREET','FOWLERTON','TEXAS','USA'), contact_table_type(contact_typ	New row added		

		e('MOBILE','+1 198-123-3456')), 'F', 'KHANDO@HOTMAIL.COM', '18-JAN-2018', '11001', '11-SEP-1981');			
23.	proc_add_staff	EXEC proc_add_staff('');	Error expected		
24.	proc_delete_staff	EXEC proc_delete_staff(7);	row deleted		
25.	proc_delete_staff	EXEC proc_add_staff('');	Error expected		
26.	proc_add_festival_staff	EXEC proc_add_festival_staff(2,3,2);	Row added		
27.	proc_add_festival_staff	EXEC proc_add_staff('');	Error expected		
28.	proc_delete_festival_staff	EXEC proc_delete_festival_staff(2);	row deleted		
29.	proc_delete_festival_staff	EXEC proc_delete_festival_staff(' ');	Error expected		

3.2.5 Packages Testing

A package is a schema object that groups logically related PL/SQL types, variables, and subprograms. Packages usually have two parts, a specification (spec) and a body; sometimes the body is unnecessary. The specification is the interface to the package. It declares the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. The body defines the queries for the cursors and the code for the subprograms. (ORACLE, 2019)

Table 3.5: Testing of Package

SN	Packages	Testing	Expected Results	Actual Results

1.	Adding data to festival_natures table	DECLARE id festival_natures.festival_nature_id%type:=8;
		BEGIN
		c_package.proc_add_fn(100,'TEST');
		END;
		/
			PL/SQL procedure successfully completed.	PL/SQL procedure successfully completed.
				
				SQL>

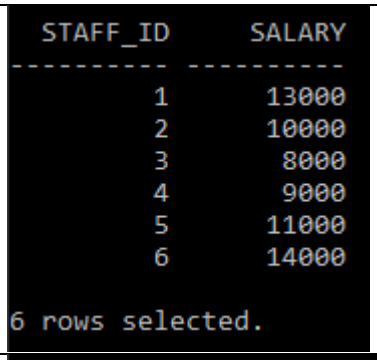
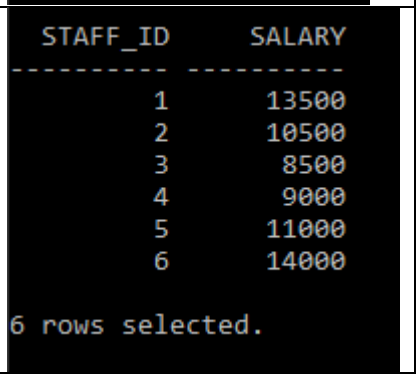
2.	Deleting data of festival_natures table	DECLARE id festival_natures.festival_nature_id%type:=8;
		BEGIN
		c_package.proc_del_fn(99);
		END;
		/
			PL/SQL procedure successfully completed.	PL/SQL procedure successfully completed.
				
				SQL>

3.	Listing data of festival_natures table	DECLARE id festival_natures.festival_nature_id%type:=8; BEGIN c_package.proc_list_fn; END; /	List all the data from festival_natures table	FESTIVAL NATURE(1): CULTURAL FESTIVAL NATURE(2): RELIGIOUS FESTIVAL NATURE(3): FOOD FESTIVAL NATURE(4): MUSIC FESTIVAL NATURE(5): GAMING FESTIVAL NATURE(6): TEST FESTIVAL NATURE(7): TEST PL/SQL procedure successfully completed.

3.2.6 Bulk Bind (FORALL)

Binds Whole arrays of values in a single operation, rather than using loop to performed the FETCH, INSERT, UPDATE and DELETE operation multiple times (Database Techies, 2019)

Table 3.6: Testing of Bulk Binding

SN	Bulk Binding	Testing	Expected Results	Actual Results
1.	testing before executing the bulk binding code	SELECT staff_id, salary FROM staff;	6 rows selected with their original salary	
2.	testing after the execution of bulk binding code	SELECT staff_id, salary FROM staff;	6 rows selected with the increase of 500 in the salary of staff_id 1, 2 and 3	

Bibliography

C#Corner. (2019). Retrieved march 20, 2019, from C#Corner: https://www.c-sharpcorner.com/UploadFile/f0b2ed/cursors-in-sql/

Database Techies. (2019). Retrieved march 20, 2019, from Database Techies: http://databasetechies.com/bulk-collect-bulk-bind/

GeeksforGeeks. (2019). Retrieved march 20, 2019, from Geeksforgeeks: https://www.geeksforgeeks.org/functions-in-plsql/

ORACLE. (2019). Retrieved march 20, 2019, from ORACLE: https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/packages.htm

ORACLE. (2019). *ORACLE*. Retrieved march 20, 2019, from ORACLE: https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6009.htm

Appendix

- **alters_10**

```
--@F:\db_group_10\alters_10.sql
--ALTER SCRIPT
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

/*--------------------------
---------PRIMARY KEYS------------
-----------------------------------*/

--primary key for festival_natures table
ALTER TABLE festival_natures
ADD CONSTRAINT pk_festival_natures
PRIMARY KEY (festival_nature_id);

--primary key for locations table
ALTER TABLE locations
ADD CONSTRAINT pk_locations
PRIMARY KEY (location_no);


--primary key for staff table
ALTER TABLE staff
ADD CONSTRAINT pk_staff
PRIMARY KEY (staff_id);


--primary key for festival_staff table
ALTER TABLE festival_staff
ADD CONSTRAINT pk_festival_staff
PRIMARY KEY (staff_id,location_no,festival_nature_id);


--primary key for festivals table
ALTER TABLE festivals
ADD CONSTRAINT pk_festivals
PRIMARY KEY (festival_nature_id,location_no);


/*--------------------------
---------FOREIGN KEYS------------
-----------------------------------*/

--foreign keys of festivals table
```

```
ALTER TABLE festivals
ADD CONSTRAINT fk_f_locations
FOREIGN KEY (location_no)
REFERENCES locations(location_no);

ALTER TABLE festivals
ADD CONSTRAINT fk_f_festival_natures
FOREIGN KEY (festival_nature_id)
REFERENCES festival_natures(festival_nature_id);


--foreign key of staff table
ALTER TABLE staff
ADD CONSTRAINT fk_s_staff
FOREIGN KEY (reports_to)
REFERENCES staff(staff_id);


--foreign key of festival_staff table
ALTER TABLE festival_staff
ADD CONSTRAINT fk_fs_staff
FOREIGN KEY (staff_id)
REFERENCES staff(staff_id);


ALTER TABLE festival_staff
ADD CONSTRAINT fk_fs_festivals
FOREIGN KEY (location_no, festival_nature_id)
REFERENCES festivals(location_no, festival_nature_id);


/* ---------------------
---------CHECK CONSTRAINTS---
-------------- -------------*/

--gender of the staff shoul be m, f, o
ALTER TABLE staff
ADD CONSTRAINT ck_staff_gender
CHECK (staff_gender IN ('M','F','O'));

--festival name should be in upper case
ALTER TABLE festivals
ADD CONSTRAINT ck_festivals
CHECK (festival_name=UPPER(festival_name));


--staff name should be in upper case
ALTER TABLE staff
```

```
ADD CONSTRAINT ck_firstname
CHECK (firstname=UPPER(firstname));

ALTER TABLE staff
ADD CONSTRAINT ck_lastname
CHECK (lastname=UPPER(lastname));


/* ---------------------
---------UNIQUE CONSTRAINTS---
-------------- -------------*/

--email of the staff should not repeat
ALTER TABLE staff
ADD CONSTRAINT uk_staff_email
UNIQUE (staff_email);
```

- **create_10**

```
--@F:\db_group_10\create_10.sql
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

/*-------------------
----------OBEJCTS-------------
--------------------*/
--creating the address_type
CREATE OR REPLACE TYPE address_type AS OBJECT (
        street VARCHAR2(50),
        city VARCHAR2(50),
        state VARCHAR2(50),
        country VARCHAR2(40))
/
SHOW ERRORS


--creating object table 'addresses' using address_type
CREATE TABLE addresses OF address_type;
SHOW ERRORS


--creating the activity_type
CREATE OR REPLACE TYPE activity_type AS OBJECT (
        activity_name VARCHAR2(50),
        category VARCHAR2(50))
/
```

```
SHOW ERRORS


--creating varray of activity_type
CREATE TYPE activity_varray_type AS VARRAY(10) OF activity_type;
/
SHOW ERRORS


--creating the contact_type
CREATE OR REPLACE TYPE contact_type AS OBJECT (
        medium VARCHAR2(25),
        contact_number VARCHAR2(25))
/
SHOW ERRORS


--creating the nested table named as contact_table_type from contact_type
CREATE TYPE contact_table_type AS TABLE OF contact_type;
/
SHOW ERRORS



/*-----------------------------------------------
---------- REALTIONAL TABLES-------------
----------------------------------**/

--creating relational table named festival_natures
CREATE TABLE festival_natures (
        festival_nature_id NUMBER(7),
        nature_name VARCHAR2(25) NOT NULL
);


--creating the relational table named festival_natures referencing the address_type
CREATE TABLE locations (
        location_no NUMBER(7),
        address REF address_type SCOPE IS addresses  NOT NULL
);


--creating relational table named festivals of activities_varray_type
CREATE TABLE festivals(
        festival_nature_id NUMBER(7) NOT NULL,
        location_no NUMBER(7) NOT NULL,
        festival_name VARCHAR2(50) NOT NULL,
        activities activity_varray_type,
```

```
        festival_start_date DATE,
        festival_end_date DATE
);


--creating relational table named staffs
CREATE TABLE staffs(
        staff_id NUMBER(7),
        reports_to NUMBER(7),
        firstname VARCHAR2(20) NOT NULL,
        lastname VARCHAR2(20) NOT NULL,
        address address_type,
        staff_gender CHAR DEFAULT 'M',
        staff_email VARCHAR2(35) NOT NULL,
        staff_employed_date DATE NOT NULL,
        username VARCHAR2(30),
        password VARCHAR2(30),
        salary NUMBER(10,2) NOT NULL,
        dob DATE NOT NULL,
        contact contact_table_type)
        NESTED TABLE contact STORE AS contact_table;


--creating the table named as festival_staffs
CREATE TABLE festival_staff(
        staff_id NUMBER(7),
        location_no NUMBER(7),
        festival_nature_id NUMBER(7)
);


/* SEQUENCES */

--creating sequences for festival_natures
CREATE SEQUENCE seq_festival_natures
INCREMENT BY 1
START WITH 0000001;

--creating sequence for locations
CREATE SEQUENCE seq_locations
INCREMENT BY 1
START WITH 0000001;

--creating seuquence for the staff table
CREATE SEQUENCE seq_staff
INCREMENT BY 1
START WITH 0000001;
```

```
--renaming
RENAME staffs TO staff;

--setting defualt vvalue to staff table
ALTER TABLE staff
MODIFY  (staff_employed_date DEFAULT SYSDATE);


--creating table names 'activity_logs' to store the user login date and time
CREATE TABLE activity_logs
(user_logged VARCHAR2(30),
        date_time DATE );
```

- **create_user_10**

```
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--creating userspace
CREATE USER group10 IDENTIFIED BY 18406499;
GRANT CREATE SESSION TO group10;
GRANT CREATE table TO group10;
GRANT CREATE view TO group10;
GRANT CREATE sequence TO group10;
GRANT CREATE synonym TO group10;
GRANT CREATE procedure TO group10;
GRANT CREATE trigger TO group10;
GRANT CREATE cluster TO group10;
GRANT CREATE type TO group10;
GRANT unlimited tablespace TO group10;
GRANT ALL PRIVILEGES TO group10;
ALTER USER group10 quota unlimited ON system;

--DROPPING USERSPACE
DROP USER group10 CASCADE;
```

- **cursor_10**

```
--@F:\db_group_10\cursor_10.sql
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/
```

```
--showing output on screen
SET SERVEROUTPUT ON

--using implicit cursor to update the firstname in the staff table
CREATE OR REPLACE PROCEDURE proc_imp_cursor(in_fname
staff.firstname%TYPE,in_lname staff.lastname%TYPE, in_nname
staff.firstname%TYPE) IS
vc_fname VARCHAR2(15);
vc_lname VARCHAR2(15);
vc_newname VARCHAR2(15);
exp1 EXCEPTION;
BEGIN
        vc_newname := in_nname;
        vc_fname := in_fname;
        vc_lname := in_lname;
        IF in_nname is NULL THEN RAISE exp1;
        ELSE
                UPDATE staff SET firstname=in_nname
                                WHERE vc_fname =in_fname AND lastname =
in_lname;
                IF SQL%FOUND THEN
                        DBMS_OUTPUT.PUT_LINE(vc_fname || ' UPDATED ');
                ELSE
                        DBMS_OUTPUT.PUT_LINE('USER DOES NOT EXIST');
                END IF;
        END IF;
--using exception handling
EXCEPTION
        WHEN exp1 THEN
                DBMS_OUTPUT.PUT_LINE('NEW NAME NOT INSERTED');
END proc_imp_cursor;
/
SHOW ERRORS


--using explicit cursor with for loop to retrieve the firstname, lastname, and salaray of
staff whose salary is greater than salary provided
CREATE OR REPLACE PROCEDURE proc_exp_cursor(in_salary
staff.salary%TYPE)IS
vn_rowcount NUMBER(2):=0;
CURSOR cur_salary IS
        SELECT firstname,lastname,salary
        FROM staff
        WHERE salary>in_salary;

BEGIN
                DBMS_OUTPUT.PUT_LINE('S.No'||'       '||'First Name'||'       '||'Last
Name'||'        '||'Salary');
```

```
		FOR rec_cur_salary IN cur_salary LOOP
			DBMS_OUTPUT.PUT_LINE(cur_salary%ROWCOUNT||'
'||rec_cur_salary.firstname||'	'|| rec_cur_salary.lastname||'
'||rec_cur_salary.salary);
			vn_rowcount := cur_salary%ROWCOUNT;
		END LOOP;

		DBMS_OUTPUT.PUT_LINE('THERE ARE '|| vn_rowcount ||'
STAFF. ');

END proc_exp_cursor;
/
SHOW ERRORS


--cursor using while loop to retrieve the staff id, firstname,lastname, and salary where
staff salary is greater than provided salary
--doesn't returns the data if the salary is greater than the salary in the database
CREATE OR REPLACE PROCEDURE proc_ckSal(in_salary staff.salary%TYPE)IS
	CURSOR cur_salary IS
		SELECT staff_id,firstname,lastname,salary
		FROM staff
		WHERE salary>in_salary;
	rec_cur_salary cur_salary%ROWTYPE;

BEGIN
DBMS_OUTPUT.PUT_LINE('S.No'||'	'||'First Name'||'	'||'Last Name'||'
'||'Salary');
OPEN cur_salary;
FETCH cur_salary INTO rec_cur_salary;
	IF cur_salary%NOTFOUND THEN
		DBMS_OUTPUT.PUT_LINE('SALARY RANGE TOO HIGH');

	ELSE
	WHILE cur_salary%FOUND LOOP
			DBMS_OUTPUT.PUT_LINE(rec_cur_salary.staff_id ||'  ||
rec_cur_salary.firstname||'  ' || rec_cur_salary.lastname||'  ' || rec_cur_salary.salary);
			FETCH cur_salary INTO rec_cur_salary;
	END LOOP;

	END IF;
CLOSE cur_salary;
END proc_ckSal;
/
SHOW ERRORS
```

- **drop_10**

```
--@F:\db_group_10\drop_10.sql
```

```
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

/*----------------------------------
------------ All the drop --------------
-----------------------------------------**/

--dropping triggers
DROP TRIGGER trig_check_age_del;
DROP TRIGGER trig_stop_pk_update_loc;
DROP TRIGGER trig_stop_pk_update;
DROP TRIGGER trig_check_festival_dates;
DROP TRIGGER trig_stop_pk_update_staff;
DROP TRIGGER trig_logon;
DROP TRIGGER trig_limit_login;


--only delete rows from all tables
DELETE FROM festival_staff;
DELETE FROM festivals;
DELETE FROM staff;
DELETE FROM locations;
DELETE FROM festival_natures;
DELETE FROM activity_logs;



--dropping other check constraints
ALTER TABLE staff
DROP CONSTRAINT uk_staff_email;

ALTER TABLE staff
DROP CONSTRAINT ck_staff_gender;

ALTER TABLE staff
DROP CONSTRAINT ck_firstname;

ALTER TABLE staff
DROP CONSTRAINT ck_lastname;

--dropping all the foreign keys
ALTER TABLE festival_staff
DROP CONSTRAINT fk_fs_staff;

ALTER TABLE festival_staff
```

```
DROP CONSTRAINT fk_fs_festivals;

ALTER TABLE festivals
DROP CONSTRAINT fk_f_festival_natures;

ALTER TABLE festivals
DROP CONSTRAINT fk_f_locations;

ALTER TABLE staff
DROP CONSTRAINT fk_s_staff;

--dropping all the primary keys

ALTER TABLE festival_staff
DROP CONSTRAINT pk_festival_staff;

ALTER TABLE festivals
DROP CONSTRAINT pk_festivals CASCADE;

ALTER TABLE festival_natures
DROP CONSTRAINT pk_festival_natures;

ALTER TABLE locations
DROP CONSTRAINT pk_locations;

ALTER TABLE staff
DROP CONSTRAINT pk_staff CASCADE;


--dropping tables
DROP TABLE festival_staff;
DROP TABLE festivals;
DROP TABLE staff;
DROP TABLE locations;
DROP TABLE festival_natures;
DROP TABLE activity_logs;

--dropping object table addresses
DROP TABLE addresses;

--dropping object types

DROP TYPE activity_varray_type;
DROP TYPE address_type;
DROP TYPE activity_type;
DROP TYPE contact_table_type;
DROP TYPE contact_type;
```

```
--dropping all the sequences
DROP SEQUENCE seq_festival_natures;
DROP SEQUENCE seq_locations;
DROP SEQUENCE seq_staff;


--dropping cursors
DROP PROCEDURE proc_imp_cursor;
DROP PROCEDURE proc_exp_cursor;
DROP PROCEDURE proc_ckSal;

--dropping procedures only
DROP PROCEDURE proc_add_addresses;
DROP PROCEDURE proc_delete_addresses_street;
DROP PROCEDURE proc_delete_addresses_city;
DROP PROCEDURE proc_delete_addresses_state;
DROP PROCEDURE proc_delete_addresses_country;
DROP PROCEDURE proc_add_location;
DROP PROCEDURE proc_delete_location;
DROP PROCEDURE proc_add_festival_natures;
DROP PROCEDURE proc_delete_festival_nature;
DROP PROCEDURE proc_add_festival;
DROP PROCEDURE proc_delete_festival;
DROP PROCEDURE proc_add_staff;
DROP PROCEDURE proc_delete_staff;
DROP PROCEDURE proc_add_festival_staff;
DROP PROCEDURE proc_delete_festival_staff;



--dropping procedures of functions
DROP PROCEDURE proc_count_staff;
DROP PROCEDURE proc_generate_age_staff;
DROP PROCEDURE proc_duration_of_festival;
DROP PROCEDURE proc_update_username_password;


--dropping functions
DROP FUNCTION func_count_staff;
DROP FUNCTION func_generate_age_staff;
DROP FUNCTION func_duration_of_festival;
DROP FUNCTION func_generate_staff_username;
DROP FUNCTION func_generate_staff_password;

--dropping package
DROP PACKAGE c_package;

--to purge all the objects from the database
```

```
PURGE RECYCLEBIN;
```

- **extras_10**

```
--@F:\db_group_10\extras_10.sql
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--extra scripts
/*-------------------------------------------------------------
-----creating package named 'c_package' which adds , deletes and lists the data of
festival_natures table
----------------------------------------**/
CREATE OR REPLACE PACKAGE c_package AS
  -- Adds a customer
  PROCEDURE proc_add_fn(in_id  festival_natures.festival_nature_id%type,
in_name festival_natures.nature_name%type);

  -- Removes a customer
  PROCEDURE proc_del_fn(in_id  festival_natures.festival_nature_id%type);
  --Lists all festival_natures
  PROCEDURE proc_list_fn;

END c_package;
/
--created


CREATE OR REPLACE PACKAGE BODY c_package AS
  PROCEDURE proc_add_fn(in_id  festival_natures.festival_nature_id%type,
in_name festival_natures.nature_name%type) IS
  BEGIN
    INSERT INTO festival_natures (festival_nature_id, nature_name)
      VALUES(in_id, in_name);
  END proc_add_fn;

  PROCEDURE proc_del_fn(in_id  festival_natures.festival_nature_id%type) IS
  BEGIN
    DELETE FROM festival_natures
    WHERE festival_nature_id = in_id;
  END proc_del_fn;


  PROCEDURE proc_list_fn IS
  CURSOR c_festival_natures is
    SELECT  nature_name FROM festival_natures;
```

```
  TYPE c_list is TABLE OF festival_natures.nature_name%type;
  name_list c_list := c_list();
  counter integer :=0;
  BEGIN
    FOR n IN c_festival_natures LOOP
    counter := counter +1;
    name_list.extend;
    name_list(counter) := n.nature_name;
    dbms_output.put_line('FESTIVAL NATURE(' ||counter|| '): '||name_list(counter));
    END LOOP;
         END proc_list_fn;

END c_package;
/
SHOW ERRORS
--created without errors


--executing the package to add to festival_natures
DECLARE
        id festival_natures.festival_nature_id%type:=8;
BEGIN
        c_package.proc_add_fn(100,'TEST');
END;
/


--executing the package to delete from festival_natures
DECLARE
        id festival_natures.festival_nature_id%type:=8;
BEGIN
        c_package.proc_del_fn(99);
END;
/


--executing the package to list  festival_natures
DECLARE
        id festival_natures.festival_nature_id%type:=8;
BEGIN
        c_package.proc_list_fn;
END;
/


--USING BULK BIND
/*-------------------------------------------------------
```

```
-----creating unnamed block which updates salary of the any two staff whose id's are
as given
--if id's do not match, a message is displayed
----------------------------------------**/

--testing before executing
SELECT staff_id, salary FROM staff;

--block to increase salary
DECLARE
  TYPE NumList IS TABLE OF staff.staff_id%TYPE;
  depts NumList := NumList(1,2);
BEGIN
  FORALL i IN depts.FIRST..depts.LAST
    UPDATE staff SET salary = salary + 500 WHERE staff_id = depts(i);
  IF SQL%BULK_ROWCOUNT(2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE('IDS DO NOT MATCH WITH ANY STAFF');
  ELSE
        DBMS_OUTPUT.PUT_LINE('SALARIES INCREMENTED');
  END IF;
END;
/
--testing if the changes are made
SELECT staff_id, salary FROM staff;
--salaries updated
```

- Functions_10

```
--@F:\db_group_10\functions_10.sql
--output display
SET SERVEROUTPUT ON


/*------------------FUNCTIONS WITHOUT PARAMETERS-----------**/
--function name 'func_count_staff 'that returns the number of staff in the staff table
CREATE OR REPLACE FUNCTION func_count_staff RETURN NUMBER AS
vn_count NUMBER(3);

BEGIN
        SELECT COUNT(staff_id)
        INTO vn_count
        FROM staff;

        RETURN vn_count;

END func_count_staff;
/
SHOW ERRORS
```

```
--procedure named'proc_count_staff' to execute the function 'func_count_staff'
CREATE OR REPLACE PROCEDURE proc_count_staff AS
vn_count NUMBER(3);
BEGIN
	vn_count := func_count_staff;
	DBMS_OUTPUT.PUT_LINE('THERE ARE '||vn_count||' STAFF
RECORDS.');
END proc_count_staff;
/
SHOW ERRORS

/*-------------------FUNCTIONS WITH PARAMETERS-----------**/
--creating a function named func_generate_age_staff which calculates age by given id
CREATE OR REPLACE FUNCTION func_generate_age_staff(in_date_of_birth
DATE) RETURN NUMBER IS
	vd_dob DATE;
	vn_calc_age NUMBER(3);
BEGIN

	vn_calc_age := FLOOR(MONTHS_BETWEEN(SYSDATE,
in_date_of_birth)/12);
	RETURN vn_calc_age;

END func_generate_age_staff;
/
SHOW ERRORS

--creating procedure named proc_generate_age_staff to call func_generate_age_staff
CREATE OR REPLACE PROCEDURE proc_generate_age_staff(in_id
staff.staff_id%TYPE)IS
	vn_calc_age NUMBER(3);
	vc_firstname staff.firstname%TYPE;
	vd_dob DATE;
BEGIN
	SELECT firstname, dob INTO vc_firstname, vd_dob FROM staff   WHERE
staff_id= in_id;
	vn_calc_age := func_generate_age_staff(vd_dob);
	DBMS_OUTPUT.PUT_LINE('AGE OF ' || vc_firstname ||' IS ' || vn_calc_age||
' YEARS OLD.');
END proc_generate_age_staff;
/
SHOW ERRORS

----------------------------------------------------------------
----------------------------------------------------------------
```

```
--creating parameterised function to calculate the duration of the festival by passing
festival's start and end dates
CREATE OR REPLACE FUNCTION func_duration_of_festival(in_start_date
DATE, in_end_date DATE) RETURN NUMBER IS
        vn_calc_duration NUMBER(4);
BEGIN
        vn_calc_duration:= in_end_date - in_start_date;
        RETURN vn_calc_duration;
END func_duration_of_festival;
/

SHOW ERRORS

--creating procedure named proc_duration_of_fetival to call
func_duration_of_festival
CREATE OR REPLACE PROCEDURE proc_duration_of_festival(in_fes_name
festivals.festival_name%TYPE)IS
        vn_duration NUMBER(3);
        vd_start DATE;
        vd_end DATE;
BEGIN
        SELECT festival_start_date,festival_end_date  INTO vd_start,vd_end FROM
festivals WHERE festival_name LIKE in_fes_name||'%';
        --SELECT festival_end_date INTO vd_end FROM festivals WHERE
festival_name = in_fes_name;
        vn_duration := func_duration_of_festival(vd_start, vd_end);
        DBMS_OUTPUT.PUT_LINE('THE DURATION OF FESTIVAL IS ' ||
vn_duration || ' DAYS.');

END proc_duration_of_festival;
/
SHOW ERRORS

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

--creating a function name func_generate_staff_username that returns username
CREATE OR REPLACE FUNCTION func_generate_staff_username(in_staff_id
staff.staff_id%TYPE) RETURN VARCHAR2 IS
  vc_username VARCHAR2(20);

BEGIN
        SELECT CONCAT(SUBSTR(firstname,1,3), SUBSTR(lastname,1,4))
        INTO vc_username
        FROM staff
        WHERE staff_id = in_staff_id;

        RETURN vc_username;
```

```
END func_generate_staff_username;
/
SHOW ERRORS

--creating a function named func_generate_staff_password that generates the
password using username, date of birth and firstname
CREATE OR REPLACE FUNCTION func_generate_staff_password(in_staff_id
staff.staff_id%TYPE) RETURN VARCHAR2 IS
  vc_password VARCHAR2(20);

BEGIN
        SELECT
CONCAT(CONCAT(UPPER(SUBSTR(username,1,4)),REPLACE(SUBSTR(dob,1,5
),'-')), SUBSTR(firstname,1,4))
        INTO vc_password
        FROM staff
        WHERE staff_id = in_staff_id;

        RETURN vc_password;

END func_generate_staff_password;
/
SHOW ERRORS

--creating a procedure named proc_insert_username_password
CREATE OR REPLACE PROCEDURE
proc_update_username_password(in_staff_id staff.staff_id%TYPE) IS
 vc_username VARCHAR2(20);
 vc_password VARCHAR2(20);
        BEGIN
        vc_username := func_generate_staff_username(in_staff_id);
         vc_password :=func_generate_staff_password(in_staff_id);

        UPDATE staff SET
                username = vc_username, password=vc_password
                WHERE staff_id = in_staff_id;


        END proc_update_username_password;
/
SHOW ERRORS
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

- **inserts_10**

```
--@F:\db_group_10\inserts_10.sql
```

```
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

/*
        Insert Queries

*/
--insert into addresses  object table
INSERT INTO addresses(street, city,state,country) VALUES ('SOUTH
SCIOTO','CIRCLEVILLE','OHIO','USA');
INSERT INTO addresses(city,state,country) VALUES ('HAWAII','HAWAII','USA');
INSERT INTO addresses(city,state,country) VALUES
('HOLLAND','MICHIGAN','USA');
INSERT INTO addresses(city,state,country) VALUES
('PITTSBURGH','PENNSYLVANIA','USA');
INSERT INTO addresses VALUES ('801 E. CESAR E. CHAVEZ BLVD','SAN
ANTONIO','TEXAS','USA');
INSERT INTO addresses VALUES ('FITCH
STREET','NORWALK','CONNECTICUT','USA');
INSERT INTO addresses VALUES ('MAIN STREET','VERMILION','OHIO','USA');
INSERT INTO addresses VALUES ('SPARTANBURG','SPARTANBURG','SOUTH
CAROLINA','USA');
INSERT INTO addresses VALUES ('THE
HIGHLANDS','LOUISVILLE','KENTUCKY','USA');
INSERT INTO addresses VALUES ('WESTERN GATEWAY PARK','DES
MOINES','IOWA','USA');
INSERT INTO addresses VALUES ('7250 STATEAVE','KANSAS
CITY','KANSAS','USA');
INSERT INTO addresses VALUES ('13590 N. 47th
AVE.','PHOENIX','ARIZONA','USA');
INSERT INTO addresses VALUES ('1901 CONVENTION CENTER DR','MIAMI
BEACH','FLORIDA','USA');
INSERT INTO addresses VALUES ('3800 NAPOLEON LN','IOWA
CITY','IOWA','USA');
INSERT INTO addresses VALUES ('100 COLUMBUS BLVD, HARTFORD, CT
06103','HARTFORD','CONNECTICUT','USA');

--insert into festival_nature using seq_festival_naturesuence
INSERT INTO festival_natures (festival_nature_id, nature_name)
VALUES(seq_festival_natures.NEXTVAL, 'CULTURAL');

INSERT INTO festival_natures (festival_nature_id, nature_name)
VALUES(seq_festival_natures.NEXTVAL, 'RELIGIOUS');

INSERT INTO festival_natures (festival_nature_id, nature_name)
```

```
VALUES(seq_festival_natures.NEXTVAL, 'FOOD');

INSERT INTO festival_natures (festival_nature_id, nature_name)
VALUES(seq_festival_natures.NEXTVAL, 'MUSIC');

INSERT INTO festival_natures (festival_nature_id, nature_name)
VALUES(seq_festival_natures.NEXTVAL, 'GAMING');


--inserting data into locations table
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='CIRCLEVILLE';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='HAWAII';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='HOLLAND';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='PITTSBURGH';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='SAN ANTONIO';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='NORWALK';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='VERMILION';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='SPARTANBURG';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='LOUISVILLE';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='DES MOINES';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='KANSAS CITY';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='PHOENIX';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='MIAMI BEACH';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='IOWA CITY';
INSERT INTO locations(location_no, address) SELECT seq_locations.NEXTVAL,
REF(a) FROM addresses a WHERE city='HARTFORD';


--inserting into festivals table
INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
```

```
VALUES(2,2,'ALOHA FESTIVAL',activity_varray_type(activity_type('FLORAL
PARADE','OUTDOOR'),activity_type('PRESENTATION OF ROYAL
COURT','OUTDOOR')),'14-SEP-2019','21-SEP-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(3,2,'TULIP TIME
FESTIVAL',activity_varray_type(activity_type('PARADES','OUTDOOR'),activity_ty
pe('FIREWORKS','OUTDOOR')),'6-MAY-2019','7-MAY-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(4,2,'PITTSBURGH FOLK
FESTIVAL',activity_varray_type(activity_type('FOLKDANCE','INDOOR'),activity_t
ype('MUSIC','INDOOR')),'6-SEP-2019','7-SEP-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(5,2,'TEXAS FOLKLIFE
FESTIVAL',activity_varray_type(activity_type('FOLKDANCE','INDOOR'),activity_t
ype('BALLET','INDOOR')),'7-SEP-2019','10-SEP-2019');

----------For Food festivals

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(1,4,'CIRCLEVILLE PUMPKIN
SHOW',activity_varray_type(activity_type('WEIGHING
PUMPKIN','OUTDOOR'),activity_type('MAKING PUMPKIN PIE','INDOOR')),'18-
OCT-2019','20-OCT-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(6,4,'NORWALK OYSTER
FESTIVAL',activity_varray_type(activity_type('FEAST','INDOOR')),'8-SEP-
2019','20-SEP-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(7,4,'FESTIVAL OF THE FISH',activity_varray_type(activity_type('FISH
FRYING','OUTDOOR'),activity_type('LOCAL BAND
CONCERT','OUTDOOR')),'5-JUN-2019','8-JUN-2019');
```

```
----------FOR MUSIC FESTIVAL
INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(8,5,'MASTERWORKS
FESTIVAL',activity_varray_type(activity_type('INTENSE ARTISTIC
TRAINING','INDOOR'),activity_type('DEEP SPIRITUAL
GROWTH','INDOOR')),'18-OCT-2019','20-OCT-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(9,5,'FORECASTLE
FESTIVAL',activity_varray_type(activity_type('MUSIC','OUTDOOR'),activity_type(
'ART','OUTDOOR')),'8-SEP-2019','8-SEP-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(10,5,'80/35 MUSIC
FESTIVAL',activity_varray_type(activity_type('ART','OUTDOOR')),'5-JUN-
2019','8-JUN-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(11,5,'ELECTRONIC MUSIC
MIDWEST',activity_varray_type(activity_type('LOCAL BAND
CONCERT','OUTDOOR')),'7-SEP-2019','10-SEP-2019');


-------GAMING FESTIVAL
INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(12,6,'DEVFARE 2019',activity_varray_type(activity_type('MOBILE
GAMING','INDOOR'),activity_type('PC GAMING','INDOOR')),'23-MAR-2019','27-
MAR-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(13,6,'FLORIDA
SUPERCON',activity_varray_type(activity_type('CONCERT','INDOOR'),activity_ty
pe('PC GAMING','INDOOR')),'8-SEP-2019','10-SEP-2019');
```

```
INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(14,6,'RDX AUSTIN
2019',activity_varray_type(activity_type('ART','INDOOR'),activity_type('MOBILE
GAMING','INDOOR')),'6-SEP-2019','11-SEP-2019');

INSERT INTO
festivals(location_no,festival_nature_id,festival_name,activities,festival_start_date,fes
tival_end_date)
VALUES(15,6,'CONNECTICON 2019',activity_varray_type(activity_type('COMIC
STALL','INDOOR'),activity_type('COSPLAY DISPLAY','INDOOR')),'7-SEP-
2019','11-SEP-2019');


--
--inserting into staff table
INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date, salary, dob)
VALUES (seq_staff.NEXTVAL, 'DAVID', 'DOE', address_type('1148  Ashton
Lane','LIBERTY HILL', 'TEXAS','USA'),
contact_table_type(contact_type('MOBILE','+1 192-554-0133')), 'M',
'DAVIDDOE@HOTMAIL.COM', '12-MAY-2017','13000', '11-JUN-1980');

INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date,salary, dob)
VALUES (seq_staff.NEXTVAL,  'SCOTT', 'WALLACE', address_type('2132
Randall Drive','HONOLULU','HAWAII','USA'),
contact_table_type(contact_type('MOBILE','+1 194-564-0133')), 'M',
'SCOTTWALLACE@GMAI;.COM', '19-MAY-2017',  '10000', '11-FEB-1989');

INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date, salary, dob)
VALUES (seq_staff.NEXTVAL,  'ELVENA ', 'JONES', address_type('775 ROCKY
ROAD','WAYNE','PENNSYLVANIA','USA'),
contact_table_type(contact_type('MOBILE','+1 272-775-0190')), 'F',
'ELVENAJONES@HOTMAIL.COM', '17-OCT-2018',  '8000', '11-APR-1984');

INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date,salary, dob)
VALUES (seq_staff.NEXTVAL,  'KEVIA', 'PAYNE', address_type('4652 SUNSET
DRIVE','PINE BLUFF','ARKANSAS','USA'),
contact_table_type(contact_type('MOBILE','+1 102-875-0003')), 'F',
'KEVIAPAYNE@GMAIL.COM', '11-FEB-2018',  '9000', '11-MAY-1985');

INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date,salary, dob)
```

```
VALUES (seq_staff.NEXTVAL,  'MENAKA', 'MORRISON', address_type('4090
MORRIS STREET','FOWLERTON','TEXAS','USA'),
contact_table_type(contact_type('MOBILE','+1 198-123-3456')), 'F',
'MENAKA@HOTMAIL.COM', '18-JAN-2017',  '11000', '11-SEP-1981');

INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date,salary, dob)
VALUES (seq_staff.NEXTVAL,  'CONNOR', 'PAYNE', address_type('4478
TRAILS END ROAD','ALBANY','NEW YORK','USA'),
contact_table_type(contact_type('MOBILE','+1 129-123-0099')), 'M',
'CONNORPAYNE@GMAIL.COM', '12-JAN-2018', '14000', '11-NOV-1982');


--inserting into festival_staff table
INSERT INTO festival_staff(staff_id,location_no,festival_nature_id) VALUES
(1,2,2);
INSERT INTO festival_staff(staff_id,location_no,festival_nature_id) VALUES
(2,2,2);
INSERT INTO festival_staff(staff_id,location_no,festival_nature_id) VALUES
(3,2,2);
INSERT INTO festival_staff(staff_id,location_no,festival_nature_id) VALUES
(6,13,6);
INSERT INTO festival_staff(staff_id,location_no,festival_nature_id) VALUES
(4,1,4);
INSERT INTO festival_staff(staff_id,location_no,festival_nature_id) VALUES
(5,10,5);


--updating staff table
UPDATE staff SET reports_to = 1 WHERE staff_id =2;
UPDATE staff SET reports_to = 1 WHERE staff_id =3;
UPDATE staff SET reports_to = 4 WHERE staff_id =5;
UPDATE staff SET reports_to = 4 WHERE staff_id =6;
```

- **procedures_10**

```
--@F:\db_group_10\procedures_10.sql

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--displaying output on screen
SET SERVEROUTPUT ON

/*
```

```
        anonymous block to retrive the festival name whose festival_nature_id is
equal to 4 an location_no is 1
        if there is no such data then the output "NO DATA FOUND" will be displayed
*/
DECLARE
        vc_festival_name festivals.festival_name%TYPE;
BEGIN
        SELECT festival_name INTO vc_festival_name
        FROM festivals WHERE festival_nature_id=4 AND location_no=1;
        DBMS_OUTPUT.PUT_LINE(vc_festival_name);

  EXCEPTION
  WHEN no_data_found THEN
  DBMS_OUTPUT.PUT_LINE('------------------NO DATA FOUND------------------
');
END;
/

--creating procedure named proc_add_addresses to insert into addresses table whose
parameter is street, city, state, country
CREATE OR REPLACE PROCEDURE proc_add_addresses(in_street
addresses.street%TYPE,
        in_city addresses.city%TYPE,
        in_state addresses.state%TYPE,
        in_country addresses.country%TYPE) IS
        BEGIN
                --insert query
                INSERT INTO addresses(street, city, state, country)
                VALUES (in_street, in_city, in_state, in_country);
        END proc_add_addresses;
/
SHOW ERRORS
--no errors

--creating procedure named proc_delete_addresses_street to delete address column
from addresses table whose street is equal to parameter of procedure
CREATE OR REPLACE PROCEDURE proc_delete_addresses_street(in_street
addresses.street%TYPE) IS
        BEGIN
                DELETE FROM addresses
                WHERE street = in_street;

        END proc_delete_addresses_street;
/
SHOW ERRORS
--no errors
```

```
--creating procedure named proc_delete_addresses_street to delete address row from addresses table whose city is equal to parameter of procedure
CREATE OR REPLACE PROCEDURE proc_delete_addresses_city(in_city addresses.city%TYPE) IS
	BEGIN
		DELETE FROM addresses
		WHERE city = in_city;

	END proc_delete_addresses_city;
/
SHOW ERRORS
--no errors

--creating procedure named proc_delete_addresses_street to delete address row from addresses table whose state is equal to parameter of procedure
CREATE OR REPLACE PROCEDURE proc_delete_addresses_state(in_state addresses.state%TYPE) IS
	BEGIN
		DELETE FROM addresses
		WHERE state = in_state;

	END proc_delete_addresses_state;
/
SHOW ERRORS
--no errors


--creating procedure named proc_delete_addresses_street to delete address row from addresses table whose country is equal to parameter of procedure
CREATE OR REPLACE PROCEDURE proc_delete_addresses_country(in_country addresses.country%TYPE) IS
	BEGIN
		DELETE FROM addresses
		WHERE country = in_country;

	END proc_delete_addresses_country;
/
SHOW ERRORS
--no errors

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

--creating procedure named proc_add_location to insert values into locations table
CREATE OR REPLACE PROCEDURE proc_add_location(in_city addresses.city%TYPE) IS
	BEGIN
		--insert query
```

```
		INSERT INTO locations(location_no, address)
		SELECT seq_locations.NEXTVAL, REF(a) FROM addresses a
WHERE city = in_city;
	END proc_add_location;
/
SHOW ERRORS
--no errors

--creating procedure named proc_delete_location to delete the location column
sending the location number in parameters
CREATE OR REPLACE PROCEDURE proc_delete_location(in_location_no
locations.location_no%TYPE) IS
	BEGIN
		--delete query
		DELETE FROM locations WHERE location_no = in_location_no;

	END proc_delete_location;
/

SHOW ERRORS
--no errors

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

--creating procedure named proc_add_festival_natures to insert the values into
festival_nature table
CREATE OR REPLACE PROCEDURE proc_add_festival_natures(in_nature_name
festival_natures.nature_name%TYPE) IS
	BEGIN
		--insert query
		INSERT INTO festival_natures(festival_nature_id,nature_name)
		VALUES (seq_festival_natures.NEXTVAL, in_nature_name);

	END proc_add_festival_natures;
/
SHOW ERRORS
--no errors

--creating procedure named proc_delete_festival_nature to delete the row whose
festival_nature_id is equal to the in_festival_nature_id in parameter
CREATE OR REPLACE PROCEDURE
proc_delete_festival_nature(in_festival_nature_id
festival_natures.festival_nature_id%TYPE) IS
	BEGIN
		--delete query
		DELETE FROM festival_natures WHERE festival_nature_id =
in_festival_nature_id;
```

```
	END proc_delete_festival_nature;
/

SHOW ERRORS;

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

--creating procedure named proc_add_festival to insert values into festivals table
CREATE OR REPLACE PROCEDURE proc_add_festival(in_location_no
festivals.location_no%TYPE,
	in_festival_nature_id festivals.festival_nature_id%TYPE,
	in_festival_name festivals.festival_name%TYPE,
	in_activities festivals.activities%TYPE,
	in_festival_start_date festivals.festival_start_date%TYPE,
	in_festival_end_date festivals.festival_end_date%TYPE) IS
	BEGIN
		--insert query
		INSERT INTO festivals(location_no, festival_nature_id,
festival_name, activities, festival_start_date, festival_end_date)
		VALUES (in_location_no, in_festival_nature_id, in_festival_name,
in_activities, in_festival_start_date, in_festival_end_date);

	END proc_add_festival;
/
SHOW ERRORS;

--creating procedure named proc_delete_festival to delete the festival rows by sending
the festival_nature_id in parameters
CREATE OR REPLACE PROCEDURE proc_delete_festival(in_festival_nature_id
festivals.festival_nature_id%TYPE) IS
	BEGIN
		--delete query
		DELETE FROM festivals WHERE
festival_nature_id=in_festival_nature_id;

	END proc_delete_festival;
/

SHOW ERRORS
--no errors

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

--creating procedure named proc_add_staff to insert the values into staffs table
```

```
CREATE OR REPLACE PROCEDURE proc_add_staff(in_firstname
staff.firstname%TYPE,
	in_lastname staff.lastname%TYPE,
	in_address staff.address%TYPE,
	in_contact staff.contact%TYPE,
	in_staff_gender staff.staff_gender%TYPE,
	in_staff_email staff.staff_email%TYPE,
	in_staff_employed_date staff.staff_employed_date%TYPE,
	in_salary staff.salary%TYPE,
	in_dob staff.dob%TYPE
	) IS
	BEGIN
		--insert query
		INSERT INTO staff(staff_id, firstname, lastname, address, contact,
			staff_gender, staff_email, staff_employed_date,salary, dob)
		VALUES (seq_staff.NEXTVAL,in_firstname, in_lastname,
in_address, in_contact,
			in_staff_gender, in_staff_email, in_staff_employed_date,
in_salary, in_dob);
	END proc_add_staff;
/
SHOW ERRORS
--no errors

--creating procedure named proc_delete_staff to delete the staff rows by sending the
staff_id in parameters
CREATE OR REPLACE PROCEDURE proc_delete_staff(in_staff_id
staff.staff_id%TYPE) IS
	BEGIN
		--delete query
		DELETE FROM staff WHERE staff_id = in_staff_id;

	END proc_delete_staff;
/

SHOW ERRORS
--no errors

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

--creating procedure named proc_add_festival_staff to insert values into staffs table
CREATE OR REPLACE PROCEDURE proc_add_festival_staff(in_staff_id
festival_staff.staff_id%TYPE,
	in_location_no festival_staff.location_no%TYPE,
	in_festival_nature_id festival_staff.festival_nature_id%TYPE
	) IS
	BEGIN
```

```
                --insert query
                INSERT INTO festival_staff(staff_id, location_no, festival_nature_id)
                VALUES (in_staff_id,in_location_no, in_festival_nature_id);

        END proc_add_festival_staff;
/
SHOW ERRORS
--no errors

--creating procedure named proc_delete_festival_staff to delete the rows by sending
the staff_id in parameters
CREATE OR REPLACE PROCEDURE proc_delete_festival_staff(in_staff_id
festival_staff.staff_id%TYPE) IS
        BEGIN
                --delete query
                DELETE FROM festival_staff WHERE staff_id = in_staff_id;

        END proc_delete_festival_staff;
/
SHOW ERRORS
--no errors
```

- **query_10**

```
--@F:\db_group_10\qeury_10.sql

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

/*----------select Queries---------**/

--view all the tables in the userspace
SELECT * FROM TAB;

--viewing all the table structures
DESC festival_natures
DESC festivals
DESC locations
DESC staff
DESC festival_staff
DESC addresses



--all the column formatting
COLUMN street FORMAT A20
COLUMN city FORMAT A15
```

```
COLUMN state FORMAT A15
COLUMN country FORMAT A10

--column formatting
COLUMN street FORMAT A20
COLUMN city FORMAT A20
COLUMN state FORMAT A10
COLUMN country FORMAT A10

--column formatting for festivals table
COLUMN festival_name FORMAT A20
COLUMN activities FORMAT A15
COLUMN festival_start_date FORMAT A10
COLUMN festival_end_date FORMAT A10

--column formatting of staff table
COLUMN firstname FORMAT A10
COLUMN lastname FORMAT A10
COLUMN address FORMAT A10
COLUMN staff_email FORMAT A10
COLUMN username FORMAT A10
COLUMN password FORMAT A10
COLUMN contact FORMAT A10


--Viewing all the data from all the tables
SELECT * FROM festival_natures;
SELECT * FROM festivals;
SELECT * FROM locations;
SELECT * FROM staff;
SELECT * FROM festival_staff;
SELECT * FROM addresses;


/*----------querying all types----------**/
----querying sequence values
DESC user_sequences
SELECT sequence_name FROM user_sequences;
SELECT seq_locations.NEXTVAL FROM DUAL;
SELECT seq_locations.CURRVAL FROM DUAL;

--viewing user_objects table structure
DESC user_objects
COLUMN OBJECT_NAME FORMAT A25

--viewing all the objects
SELECT object_name, object_type FROM user_objects;
```

```
SELECT object_name, object_type FROM user_objects WHERE object_name NOT
LIKE 'S%';
SELECT object_name, object_type FROM user_objects WHERE object_type  LIKE
'TYP%';
SELECT object_name, object_type FROM user_objects WHERE object_type  LIKE
'TAB%';
SELECT object_name, object_type FROM user_objects WHERE object_type  LIKE
'PR%';
SELECT object_name, object_type FROM user_objects WHERE object_type  LIKE
'F%';


--querying triggers
--viewing trigger's table structure
DESC user_triggers
SELECT trigger_name, trigger_type FROM user_triggers;


--Viewing all the constraints
DESC user_constraints
SELECT constraint_name, constraint_type FROM user_constraints;
SELECT constraint_name, constraint_type FROM user_constraints WHERE
constraint_name NOT LIKE 'SYS%';
SELECT constraint_name, constraint_type FROM user_constraints WHERE
constraint_name  LIKE 'F%';
SELECT constraint_name, constraint_type FROM user_constraints WHERE
constraint_name  LIKE 'P%';


--Viewing all the sequence
DESC user_sequences
SELECT sequence_name FROM user_sequences;


--Query Festival Natures
SELECT * FROM festival_natures
WHERE festival_nature_id = 2;


--select query using OR
SELECT * FROM festival_natures
WHERE festival_nature_id = 3
OR festival_nature_id = 5;

--using projection
SELECT festival_nature_id, location_no,festival_name FROM festivals;
SELECT staff_id, firstname FROM staff;
```

```
--Query of festivals
SELECT festival_name FROM festivals
WHERE festival_name = 'DEVFARE 2019';

SELECT festival_name FROM festivals
WHERE festival_name LIKE 'A%';

SELECT festival_name, festival_start_date FROM festivals
WHERE festival_start_date = '14-SEP-2019';

SELECT festival_name, festival_start_date as "START", festival_end_date as "END"
FROM festivals
WHERE festival_end_date < '06-MAY-2019';

SELECT festival_name, festival_start_date as "START", festival_end_date as "END"
FROM festivals
WHERE festival_start_date = '06-SEP-2019' AND festival_end_date = '07-SEP-
2019';


--query using between
SELECT festival_name
FROM festivals
WHERE festival_start_date
BETWEEN '05-JUN-2019'
AND '07-SEP-2019';

--using union
SELECT l.address.country
FROM locations l
UNION
SELECT s.address.country
FROM staff s;

--minus | intersect |
SELECT l.address.city city
FROM locations l
INTERSECT
SELECT s.address.city city
FROM staff s;

SELECT l.address.city city
FROM locations l
MINUS
SELECT s.address.city city
FROM staff s;
```

```
--select query using IN
SELECT festival_name
FROM festivals
WHERE festival_start_date IN ('8-SEP-2019', '10-SEP-2019');

--select query using ALL
SELECT * FROM festivals
WHERE festival_start_date = ALL('8-SEP-2019', '10-SEP-2019');


--query joining the festival and location
SELECT fe.festival_name, fe.festival_start_date, fe.festival_end_date, l.address.street
street,l.address.city city
FROM locations l
INNER JOIN festivals fe
ON l.location_no = fe.location_no;


--LOCATIONS QUERY

--using poor method
SELECT * FROM locations
WHERE location_no = 0000001;

--using deference
SELECT location_no, DEREF(address) FROM locations;

--using best method (dot notation)
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;

--in descending order
SELECT location_no, l.address.street street,l.address.city city, l.address.state state
FROM locations l
ORDER BY state DESC;


/*--------------------------
----------STAFF QUERY----------
---------------------------------**/
SELECT firstname, username, password
FROM staff
WHERE staff_id = 0000001;

SELECT firstname, lastname, contact, staff_email,staff_gender ,username
FROM staff
WHERE staff_gender = 'M';
```

```
--querying staff ID , firstname and lastname whose firstname has 'a' in it ;
SELECT staff_id, firstname ,lastname
FROM staff
WHERE firstname  LIKE '%A%';

--querying staff ID, firstname and lastname by matching staff first name that start
exactly with D and ends with E;
SELECT staff_id, firstname, lastname
FROM staff
WHERE lastname LIKE '_O_';


/*-----------------------------------------------------
----------------------NESTED QUERY--------------**/
--using EXISTS, querying the staff table to show staff whose salaries are greater than
their mentor
SELECT i.staff_id, i.firstname, i.salary, i.reports_to
FROM staff i
WHERE EXISTS(
        SELECT staff_id
        FROM staff m
        WHERE i.reports_to IS NOT NULL
        AND i.reports_to = m.staff_id
        AND i.salary > m.salary);

--using NOT IN, showing the cities from the addresses table which are not in staff
table
SELECT city
FROM addresses
WHERE city NOT IN(
        SELECT s.address.city
        FROM staff s);

--using IN,viewing all the festival name in festivals in 'HOLLAND' city
SELECT festival_name
FROM festivals
WHERE location_no IN(
        SELECT location_no
        FROM locations l
        WHERE l.address.city = 'HOLLAND');


--using =, viewing the staff's firstname whose salary is the highest
SELECT DISTINCT(firstname), salary
FROM staff
WHERE salary =
        (SELECT MAX(salary) FROM staff);
```

```
--IS NULL
SELECT staff_id, firstname, username
FROM staff s
WHERE s.username IS NULL;

--CROSS JOINS DATA(CARTESIAN PRODUCT)
SELECT f.festival_name, s.firstname
FROM festivals f, staff s;

--JOIN using table alias
SELECT l.location_no, f.festival_nature_id
FROM locations l JOIN festivals f
ON l.location_no = f.location_no;

-- INNER JOIN BY APPLYING CONDITION
SELECT s.address,fs.location_no
FROM staff s JOIN festival_staff fs
ON s.staff_id = fs.staff_id
WHERE s.staff_gender = 'M';

--LEFT, RIGHT AND OUTER JOINS
SELECT fs.nature_name,f.festival_name
FROM festival_natures fs LEFT JOIN festivals f
ON fs.festival_nature_id = f.festival_nature_id;

SELECT fs.nature_name,f.festival_name
FROM festival_natures fs RIGHT JOIN festivals f
ON fs.nature_name = 'CULTURAL';

SELECT fs.nature_name,f.festival_name AS fest_name
FROM festival_natures fs FULL OUTER JOIN festivals f
ON fs.festival_nature_id = f.festival_nature_id;




/*-----------------------------------------------------
----------BUILT IN FUNCTIONS--------------------
-----------------------------------**/
--retrieving specific data using DISTINCT
SELECT DISTINCT(s.firstname),fs.staff_id
FROM staff s JOIN festival_staff fs
ON s.staff_id = fs.staff_id;

--SUM
```

```
SELECT SUM(salary) AS "Total Salary"
FROM staff;

--COUNT function with GROUP BY
SELECT festival_nature_id,COUNT(festival_name) AS total_festivals
FROM festivals
GROUP BY festival_nature_id
ORDER BY festival_nature_id;

--MIN function
SELECT MIN(salary) AS "LOWEST SALARY"
FROM staff;

--MAX function
SELECT MAX(salary) AS "HIGHEST SALARY"
FROM staff;

--AVG FUNCTION
SELECT AVG(salary) AS "AVERAGE SALARY EXPENSE"
FROM staff;

--TRIM
SELECT TRIM('  HELLO !')  FROM DUAL;


--REPLACE
SELECT REPLACE(SYSDATE, '-')  FROM DUAL;


--RUNTIME PARAMETERS
--counting the locations where festivals of type given 'festival_nature_id' are held
SELECT COUNT(location_no) AS locations ,festival_nature_id
FROM festivals
GROUP BY festival_nature_id
HAVING festival_nature_id= '&id';


--using CEIL,FLOOR,ROUND and TRUNC in one statement to show values of
salary
SELECT salary,
                CEIL(salary) CEILED,
                FLOOR(salary) FLOORED,
                ROUND(salary) ROUNDED,
          TRUNC(salary) TRUNCATED
FROM staff;


/*-----------OBJECT QUERYING----------
```

```
----------------------------------------**/

--viewing contact_type
DESC contact_type

--using poor method
SELECT      UPPER(firstname), contact FROM staff;
SELECT      LOWER(firstname), contact FROM staff;

--using best method
SELECT s.firstname firstname, a.medium medium, a.contact_number contact
FROM staff s , TABLE(s.contact) a;

--querying only the nested table "contact_table_type"
SELECT VALUE(s)
FROM THE(
        SELECT contact
        FROM staff
        WHERE staff_id=5)s;

--querying varray, activity_varray_type
DESC activity_type

--using poor method
SELECT festival_name, activities FROM festivals;

--using best method
--COLUMN FORMATTING
COLUMN festival FORMAT A10
COLUMN activity FORMAT A10
COLUMN category FORMAT A10
SELECT f.festival_name festival, a.activity_name activity, a.category category
FROM festivals f , TABLE(f.activities) a;


--query using switch case
SELECT s.firstname,s.lastname, s.address.city city, s.address.country country
FROM staff s
ORDER BY (CASE
WHEN s.address.city IS NULL THEN s.address.country
ELSE s.address.city
END);


--checking user login log 'activity_logs' table
SELECT user_logged, TO_CHAR(date_time, 'MM/DD/YYYY HH24:MI:SS') AS "
DATE-TIME"  FROM  activity_logs ;
```

- **runnable_10**

```
--@F:\db_group_10\runnables_10.sql

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--run command

@F:\db_group_10\create_10.sql
@F:\db_group_10\alters_10.sql
@F:\db_group_10\inserts_10.sql
@F:\db_group_10\functions_10.sql
@F:\db_group_10\procedures_10.sql
@F:\db_group_10\cursor_10.sql
@F:\db_group_10\extras_10.sql
@F:\db_group_10\qeury_10.sql
@F:\db_group_10\triggers_10.sql
@F:\db_group_10\tests_10.sql
@F:\db_group_10\drop_10.sql

--to check whether all the commands worked or not
SELECT * FROM TAB;
```

- **test_func_10**

```
--@F:\db_group_10\test_func_10.sql

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--test script of all the functions and related procedures
--testing 'functions_10.sql'

--to see the output on screen
SET SERVEROUTPUT ON

--column formatting
COLUMN firstname FORMAT A10
COLUMN lastname FORMAT A10
COLUMN username FORMAT A10
COLUMN password FORMAT A20
COLUMN address FORMAT A10
COLUMN staff_email FORMAT A10

------------------------------------------------
```

```
---------for procedures 'proc_count_staff'---------
-------------------------------------------------
--checking the staffs
DESC staff;

--executing 'proc_count_staff'
EXECUTE proc_count_staff;
--Result: THERE ARE 6 STAFF RECORDS

SELECT * FROM staff;
--6 rows returned

-------------------------------------------------
---------for procedure 'proc_generate_age_staff'---
-------------------------------------------------
DESC staff;

--executing by putting the date value in parameter
EXECUTE proc_generate_age_staff(3);
--Result: AGE OF ELVENA  IS 35 YEARS OLD.

--runtime parameter
EXEC proc_generate_age_staff(&in_id);
--shows age


-------------------------------------------------
---------for 'proc_duration_of_festival'-----------
-------------------------------------------------
--viwing table structure
DESC festivals

--executing by putting the festival name in parameter to calculate the duration of
festival
EXEC proc_duration_of_festival('DEVFARE');
--Result: THE DURATION OF FESTIVAL IS 4 DAYS.

-------------------------------------------------
---------for 'proc_update_username_password'-------
-------------------------------------------------
--first checking data of the table staff
SELECT firstname, username, password FROM staff
WHERE staff_id = 2;
-- Result: username and password columns are empty

--executing the procedure by putting the staff_id to generate both username and
password of that user
EXEC proc_update_username_password(2);
```

```
--PL/SQL procedure successfully completed

--checking if the username and password is updated in the staff table
SELECT firstname, username, password FROM staff
WHERE staff_id = 2;
--Result: username and password updated
```

- **tests_10**

```
--@F:\db_group_10\tests_10.sql
/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--all the test script's run command

@F:\db_group_10\tests_trigggers.sql
@F:\db_group_10\tests_cur_10.sql
@F:\db_group_10\test_func_10.sql
@F:\db_group_10\tests_proc_10.sql
```

- **tests_cur_10**

```
--@F:\db_group_10\tests_cur_10.sql

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

--Testing Cursor

/*----------------------------------
---------for 'proc_imp_cursor'-------
----------------------------------*/
--shows all the columns of staff table
DESC staff;

--checking the total staff
SELECT firstname FROM staff;
--Result: 6 rows returned

INSERT INTO staff (staff_id,firstname,lastname,staff_email,salary,dob)
VALUES(8,'RAM','SHRESTHA','123@gmail.com','10000', '11-FEB-1989');

--executing procedure
```

```
EXEC proc_imp_cursor('DAVID','DOE','SHYAM');
--Result: FIRST NAME DAVID UPDATED TO SHYAM

SELECT firstname FROM staff;
--Result: 7 rows returned

--re executing with firstname not in sites table
EXEC proc_imp_cursor('TENZIN','SHERPA','');
--USER DOES NOT EXIST
EXEC proc_imp_cursor('SHYAM','','');
--USER DOES NOT EXIST
SELECT firstname FROM staff;



/*---------------------------------------
---------for 'proc_exp_cursor'-------
---------------------------------------*/
--Testing of explicit cursor

--checking all the rows from staff table
SELECT salary FROM staff;
--8 rows selected

--executing cursor with value 10000 in parameter
EXEC proc_exp_cursor(10000);
--Result: THERE ARE 3 STAFF

--executing cursor with value 6000 in parameter
EXEC proc_exp_cursor(6000);
--Result: THERE ARE 8 STAFF

--executing cursor with value 99000 in parameter
EXEC proc_exp_cursor(99000);
--Result: THERE ARE 0 STAFF



/*---------------------------------------
---------for 'proc_ckSal'---------
---------------------------------------*/
--checking all the rows from staff table
SELECT salary FROM staff;
--8 rows selected

--executing procedure with value 6000
EXEC proc_ckSal(6000);
--Result: SHOWS TABLE with 8 rows
```

```
--executing procedure with value 10000
EXEC proc_ckSal(10000);
--Result: SHOWS TABLE with 3 rows

--executing procedure with value 100000
EXEC proc_ckSal(100000);
--Result: SALARY RANGE TOO HIGH
```

- **tests_proc_10**

```
--@F:\db_group_10\tests_proc_10.sql

--TESTING

/*-----------------------------------------------------
        ------for procedure 'proc_add_addresses'----
-----------------------------------------------------*/

--first checking data in the addresses object table
SELECT * FROM addresses;
--14 rows returned

--executing the procedure with correct data
EXEC proc_add_addresses('SOUTH SCIOTO','CIRCLEVILLE','OHIO','USA');

--checking if the procedure added data in the table
SELECT * FROM addresses;
--15 rows returned

--executing the procedure with incorrect data
EXEC proc_add_addressess('CIRCLEVILLE','OHIO','USA');

--checking if the procedure added data in the table
SELECT * FROM addresses;
--no row added and returns the 15 rows returned

/*-----------------------------------------------------
-----for procedure 'proc_delete_addressess_street'-------
-----------------------------------------------------*/

--first checking data in the addresses object table
SELECT * FROM addresses;
--15 rows returned

--executing the procedure with correct data
EXEC proc_delete_addresses_street('SOUTH SCIOTO');

--checking if the procedure added data in the table
SELECT * FROM addresses;
--14 rows returned
```

```
--executing the procedure with incorrect data
EXEC proc_delete_addressess_street(1);

--checking if the procedure added data in the table
SELECT * FROM addresses;
--no row added

/*-----------------------------------------------------
-----for procedure 'proc_delete_addressess_city'---------
-----------------------------------------------------*/

--first checking data in the addresses object table
SELECT * FROM addresses;
-- 15 rows returned

--executing the procedure with correct data
EXEC proc_delete_addresses_city('CIRCLEVILLE');

--checking if the procedure deleted data in the table
SELECT * FROM addresses;
--14 rows returned

--executing the procedure with incorrect data
EXEC proc_delete_addresses_city(01);

--checking if the procedure deleted data in the table
SELECT * FROM addresses;
--no row added



/*-------------------------------------------------------------
---------for procedure 'proc_delete_addressess_state'----------
-------------------------------------------------------------------------*/

--first checking data in the addresses object table
SELECT * FROM addresses;
-- 15 rows returned

--executing the procedure with correct data
EXEC proc_delete_addresses_state('OHIO');

--checking if the procedure deleted data in the table
SELECT * FROM addresses;
--13 rows returned

--executing the procedure with incorrect data
```

```
EXEC proc_delete_addresses_state(01);

--checking if the procedure added data in the table
SELECT * FROM addresses;
--no row added

-----------------------------------------------------
-----for procedure 'proc_delete_addresses_country'-----
-----------------------------------------------------
--first checking data in the addresses object table
SELECT * FROM addresses;
-- 13 rows returned

--executing the procedure with correct data
EXEC proc_delete_addresses_country('USA');

--checking if the procedure deleted data in the table
SELECT * FROM addresses;
--13 rows returned

--executing the procedure with incorrect data
EXEC proc_delete_addresses_country(01);

--checking if the procedure added data in the table
SELECT * FROM addresses;
--no row added

-----------------------------------------------------
-------for procedure 'proc_add_location'--------------
-----------------------------------------------------
--first checking data in the location table
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;
-- 15 rows returned

--executing the procedure with correct data
EXEC proc_add_location('TEST');

--checking if the procedure added data in the table
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;
--16 rows returned

--executing the procedure with incorrect data
EXEC proc_add_location(01);
```

```
--checking if the procedure added data in the table
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;
--no row added

-----------------------------------------------------
-------for procedure 'proc_delete_location'------------
-----------------------------------------------------
--first checking data in the location table
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;
-- 15 rows returned

--executing the procedure with correct data
EXEC proc_delete_location(16);

--checking if the procedure deleted data in the table
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;
--16 rows returned

--executing the procedure with incorrect data
EXEC proc_delete_location('TEST');

--checking if the procedure deleted data in the table
SELECT location_no, l.address.street street,l.address.city city, l.address.state state,
l.address.country country
FROM locations l;


-----------------------------------------------------
----for procedure 'proc_add_festival_natures' ---------
-----------------------------------------------------
--first checking data in the festival_natures table
SELECT * FROM festival_natures;
--5  rows returned

--executing the procedure with correct data
EXEC proc_add_festival_natures('TEST');

--checking if the procedure added data in the table
SELECT * FROM festival_natures;
--6 rows returned

--executing the procedure with incorrect data
```

```
EXEC proc_add_festival_natures('');

--checking if the procedure added data in the table
SELECT * FROM festival_natures;
--no row added

-----------------------------------------------------
-----for procedure 'proc_delete_festival_nature' ------
-----------------------------------------------------
--first checking data in the festival_natures table
SELECT * FROM festival_natures;
--6  rows returned

--executing the procedure with correct data
EXEC proc_delete_festival_nature(7);

--checking if the procedure deleted data in the table
SELECT * FROM festival_natures;
--5 rows returned

--executing the procedure with incorrect data
EXEC proc_add_festival_natures('');

--checking if the procedure added data in the table
SELECT * FROM festival_natures;
--no row added

-----------------------------------------------------
-----for procedure 'proc_add_festival' -------------
-----------------------------------------------------
--first checking data in the festivals table
SELECT * FROM festivals;
--15  rows returned

--executing the procedure with correct data
EXEC proc_add_festival(12, 2,'TULIP ME
FESTIVAL',activity_varray_type(activity_type('PARADS','OUTDOOR'),activity_typ
e('FIREWORKS','OUTDOOR')),'6-MAY-2019','7-MAY-2019');

--checking if the procedure deleted data in the table
SELECT * FROM festivals;
--16 rows returned

--executing the procedure with incorrect data
EXEC proc_add_festival('');
-- procedure didn't execute

--checking if the procedure added data in the table
```

```
SELECT * FROM festivals;
--no row added

-----------------------------------------------------
---------for procedure 'proc_add_festival' ----------
-----------------------------------------------------
--first checking data in the festivals table
SELECT * FROM festivals;


--executing the procedure with correct data
EXEC proc_delete_festival(2);

--checking if the procedure deleted data in the table
SELECT * FROM festivals;
--11 rows returned

--executing the procedure with incorrect data
EXEC proc_delete_festival('');
-- procedure didn't execute

--checking if the procedure added data in the table
SELECT * FROM festivals;
--no row added

-----------------------------------------------------
-------------for procedure 'proc_add_staff' -----------
-----------------------------------------------------
--first checking data in the staff table
SELECT * FROM staff;

--executing the procedure with correct data
EXEC proc_add_staff('KHANDO', 'MORRISON', address_type('4090 MORRIS
STREET','FOWLERTON','TEXAS','USA'),
contact_table_type(contact_type('MOBILE','+1 198-123-3456')), 'F',
'KHANDO@HOTMAIL.COM', '18-JAN-2018',  '11001', '11-SEP-1981');

--checking if the procedure deleted data in the table
SELECT * FROM staff;


--executing the procedure with incorrect data
EXEC proc_add_staff('');
-- procedure didn't execute

--checking if the procedure added data in the table
SELECT * FROM staff;
--no row added
```

```
-----------------------------------------------------
--------for procedure 'proc_delete_staff'--------------
-----------------------------------------------------
--first checking data in the staff table
SELECT * FROM staff;
--7  rows returned

--executing the procedure with correct data
EXEC proc_delete_staff(7);

--checking if the procedure deleted data in the table
SELECT * FROM staff;
--6 rows returned

--executing the procedure with incorrect data
EXEC proc_add_staff('');
-- procedure didn't execute

--checking if the procedure added data in the table
SELECT * FROM staff;
--no row added

-----------------------------------------------------
--------for procedure 'proc_add_festival_staff' -------
-----------------------------------------------------
--first checking data in the festival_staff table
SELECT * FROM festival_staff;
--6 rows returned

--executing the procedure with correct data
EXEC proc_add_festival_staff(2,2,2);

--checking if the procedure deleted data in the table
SELECT * FROM festival_staff;
--7 rows returned

--executing the procedure with incorrect data
EXEC proc_add_staff('');
-- procedure didn't execute

--checking if the procedure added data in the table
SELECT * FROM festival_staff;
--no row added

-----------------------------------------------------
------for procedure 'proc_delete_festival_staff' ------
-----------------------------------------------------
```

```
--first checking data in the festival_staff table
SELECT staff_id FROM festival_staff;
--7 rows returned

--executing the procedure with correct data
EXEC proc_delete_festival_staff(2);

--checking if the procedure deleted data in the table
SELECT * FROM festival_staff;
--5 rows returned

--executing the procedure with incorrect data
EXEC proc_delete_festival_staff(' ');
-- procedure didn't execute

--checking if the procedure added data in the table
SELECT * FROM festival_staff;
--no row added
```

- **tests_triggers**

```
--@F:\db_group_10\tests_trigggers.sql
--testing triggers

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/

/*----------------------------
-----------TESTING trig_check_festival_dates------------
----------------------**/
--checking location_no , festival_nature_id
SELECT location_no FROM locations;
--16 rows selected

SELECT  festival_nature_id FROM festival_natures;
--        7 rows selected

--inserting in the festivals table with festival_start_date less than festival_end_date
INSERT INTO festivals(location_no,festival_nature_id,
festival_name,activities,festival_start_date,festival_end_date)
VALUES(7,2,'TEST FESTIVAL',
activity_varray_type(activity_type('PARADE','OUTDOOR'),
activity_type('PRESENTATION','OUTDOOR')),'10-NOV-2019','20-NOV-2019');
--Result: 1 rows created and no triggers generated

--inserting in the festivals table with festival_start_date greater than festival_end_date
```

```
INSERT INTO festivals(location_no, festival_nature_id,
festival_name, activities, festival_start_date,festival_end_date)
VALUES(8,2,'TEST FESTIVAL',
activity_varray_type(activity_type('PARADE','OUTDOOR'),
activity_type('PRESENTATION','OUTDOOR')),
'30-NOV-2020','20-NOV-2019');
--Displayed the error message.


/*------------------------------
-----------TESTING trig_check_insert_update_del-----
-------------------------**/
--testing the triger by updating the satff's age with age >18
UPDATE staff SET dob= '01-JAN-1990' WHERE staff_id =1;
--SUCCESSFUL DISPLAYED

--deleting
DELETE FROM staff WHERE staff_id=2;
--deleted

--testing with age>60
UPDATE staff SET dob= '01-JAN-1920' WHERE staff_id =1;
--trigger raised error

--testing with age<18
UPDATE staff SET dob= '01-JAN-2004' WHERE staff_id =1;
--trigger raised error


/*------------------------------
-----------TESTING trig_stop_pk_update------
---------------------------**/

SELECT festival_nature_id FROM festival_natures;

--inserting new dummy id
INSERT INTO festival_natures VALUES(200,'TEST');
--successfully inserted
SELECT festival_nature_id FROM festival_natures;
--7 rows selected
--testing by trying to change the festival_nature_id of 2to 9
UPDATE festival_natures SET festival_nature_id=222 WHERE
festival_nature_id=200;
--TRIGGER GENERATED


/*------------------------------
--------TESTING trig_stop_pk_update_loc----
```

```
------------**/
SELECT location_no FROM locations;
--16 rows selected
--inserting new dummy id
INSERT INTO locations(location_no, address) SELECT 200, REF(a) FROM
addresses a WHERE a.city='HARTFORD';
--successfully inserted
--testing by trying to change location_no of 1 to 40
UPDATE locations SET location_no =209 WHERE location_no=200;



/*------------------------------
--------TESTING trig_stop_pk_update_staff----
------------**/

--inserting dummy data
INSERT INTO staff (staff_id, firstname, lastname, address, contact, staff_gender,
staff_email, staff_employed_date,salary, dob)
VALUES (99,  'CONNOR', 'PAYNE', address_type('4478 TRAILS END
ROAD','ALBANY','NEW YORK','USA'),
contact_table_type(contact_type('MOBILE','+1 129-123-0099')), 'M',
'CONNOE@GMAIL.COM', '12-JAN-2018', '14000', '11-NOV-1982');

SELECT staff_id , firstname FROM staff;
--7 rows selected
--testing
UPDATE staff SET staff_id=1000 WHERE staff_id=100;
--



/*------------------------------
--------TESTING trig_logon----
------------**/
--cannot login  before 8am and after 6pm.
--CONNECT SYS AS SYSDBA

--CONNECT group10
```

- **triggers_10**

```
--@F:\db_group_10\triggers_10.sql

/*
        GROUP 10
        Name: Dipen Maharjan, Tenzin Dhundup, Sarina Acharya,Tsering Khando
        UoN: 18406500, 18406552, 18406480, 18406499
*/
```

```
/*-------------------------------------
---Time limit for the user to create alter or drop in the database.
--The trigger is fired when the user tries to login except between 8am to 6pm.
-----------------------------------------**/

CREATE OR REPLACE TRIGGER trig_limit_login
AFTER CREATE OR ALTER OR DROP ON group10.SCHEMA
DECLARE
        vn_hour NUMBER(2);

BEGIN
SELECT TO_NUMBER(TO_CHAR(SYSDATE,'HH24')) INTO vn_hour FROM
DUAL;
                IF vn_hour<13 OR vn_hour>23THEN
                        RAISE_APPLICATION_ERROR(-20001,'CANNOT MAKE
CHANGES IN DATABASE BEFORE 8PM AND AFTER 6PM.');
                END IF;
END;
/
SHOW ERRORS
--no errors

--trigger name 'trig_check_festival_dates' to check whether the festival start and end
dates are valid
--The trigger is fired when the user tries to enter the end date earlier than the  start
date.
CREATE OR REPLACE TRIGGER  trig_check_festival_dates
BEFORE INSERT OR UPDATE ON festivals
FOR EACH ROW
WHEN (NEW.festival_end_date < NEW.festival_start_date)

BEGIN
        RAISE_APPLICATION_ERROR(-20004, 'FESTIVAL END DATE IS
LOWER THAN START DATE!!!');
END trig_check_festival_dates;
/
SHOW ERRORS
--trigger created

--trigger named 'trig_check_age_del'
--The trigger is fired when the user tries to enter the age of the staff either below 18 or
above 60.
CREATE OR REPLACE TRIGGER trig_check_age_del
AFTER INSERT OR UPDATE OR DELETE OF dob ON staff
FOR EACH ROW
DECLARE
      vn_age NUMBER(3);
      vd_today DATE;
```

```
BEGIN
SELECT SYSDATE INTO vd_today FROM DUAL;
                vn_age:=FLOOR(MONTHS_BETWEEN(sysdate, :NEW.dob)/12);
IF INSERTING OR UPDATING THEN
  IF (vn_age<18) OR (vn_age>60) OR :NEW.dob>vd_today THEN
          RAISE_APPLICATION_ERROR(-20002,'THE AGE MUST BE BETWEEN
18 - 60 YEARS');
  ELSE
          DBMS_OUTPUT.PUT_LINE('SUCCESSFUL');
END IF;
ELSE
    DBMS_OUTPUT.PUT_LINE(' YOU ARE DELETING '|| :OLD.firstname);

END IF;
END trig_check_age_del;
/
SHOW ERRORS


-- The trigger is fired when updating primary key in any table
--FESTIVAL_NATURES TABLE
-- The trigger is fired when updating primary key in  the festival_natures
CREATE OR REPLACE TRIGGER trig_stop_pk_update
AFTER UPDATE OF festival_nature_id ON festival_natures
BEGIN
          RAISE_APPLICATION_ERROR(-20003,'You cannot update the primary
key');

END trig_stop_pk_update;
/
SHOW ERRORS

--LOCATIONS TABLE
-- The trigger is fired when updating primary key in the locations table
CREATE OR REPLACE TRIGGER trig_stop_pk_update_loc
AFTER UPDATE OF location_no ON locations
BEGIN
          RAISE_APPLICATION_ERROR(-20003,'You cannot update the primary
key');

END trig_stop_pk_update_loc;
/
SHOW ERRORS

--STAFF TABLE
-- The trigger is fired when updating primary key in the staff table
CREATE OR REPLACE TRIGGER trig_stop_pk_update_staff
```

```
AFTER UPDATE OF staff_id ON staff
BEGIN
        RAISE_APPLICATION_ERROR(-20003,'You cannot update the primary
key');

END trig_stop_pk_update_staff;
/
SHOW ERRORS


/*-----------  ---------------------------
----------DATABASE LEVEL TRIGGER-------------------
------------------------------------------ ----**/
--trigger named 'trig_logon' that inserts date and time when user logs on the system
into the 'activity_logs' table
CREATE OR REPLACE TRIGGER trig_logon
AFTER LOGON
ON DATABASE
BEGIN
     INSERT INTO activity_logs(user_logged, date_time)
     VALUES (USER, SYSDATE);
END;
  /
SHOW ERRORS
--trigger created with  no errors


/*---------------------------------
----------ENABLE | DISABLE TRIGGERS-------------
------------------------**/

------To disable the triggers
ALTER TRIGGER trig_limit_login DISABLE;
ALTER TRIGGER trig_check_festival_dates DISABLE;
ALTER TRIGGER trig_check_age_del DISABLE;
ALTER TRIGGER trig_stop_pk_update DISABLE;
ALTER TRIGGER trig_stop_pk_update_loc DISABLE;
ALTER TRIGGER trig_stop_pk_update_staff DISABLE;
ALTER TRIGGER trig_logon DISABLE;

-------To enable the triggers

ALTER TRIGGER trig_limit_login ENABLE;
ALTER TRIGGER trig_check_festival_dates ENABLE;
ALTER TRIGGER trig_check_age_del ENABLE;
ALTER TRIGGER trig_stop_pk_update ENABLE;
ALTER TRIGGER trig_stop_pk_update_loc ENABLE;
ALTER TRIGGER trig_stop_pk_update_staff ENABLE;
```

```
ALTER TRIGGER trig_logon ENABLE;
```