ASSIGNMENT, TERM II

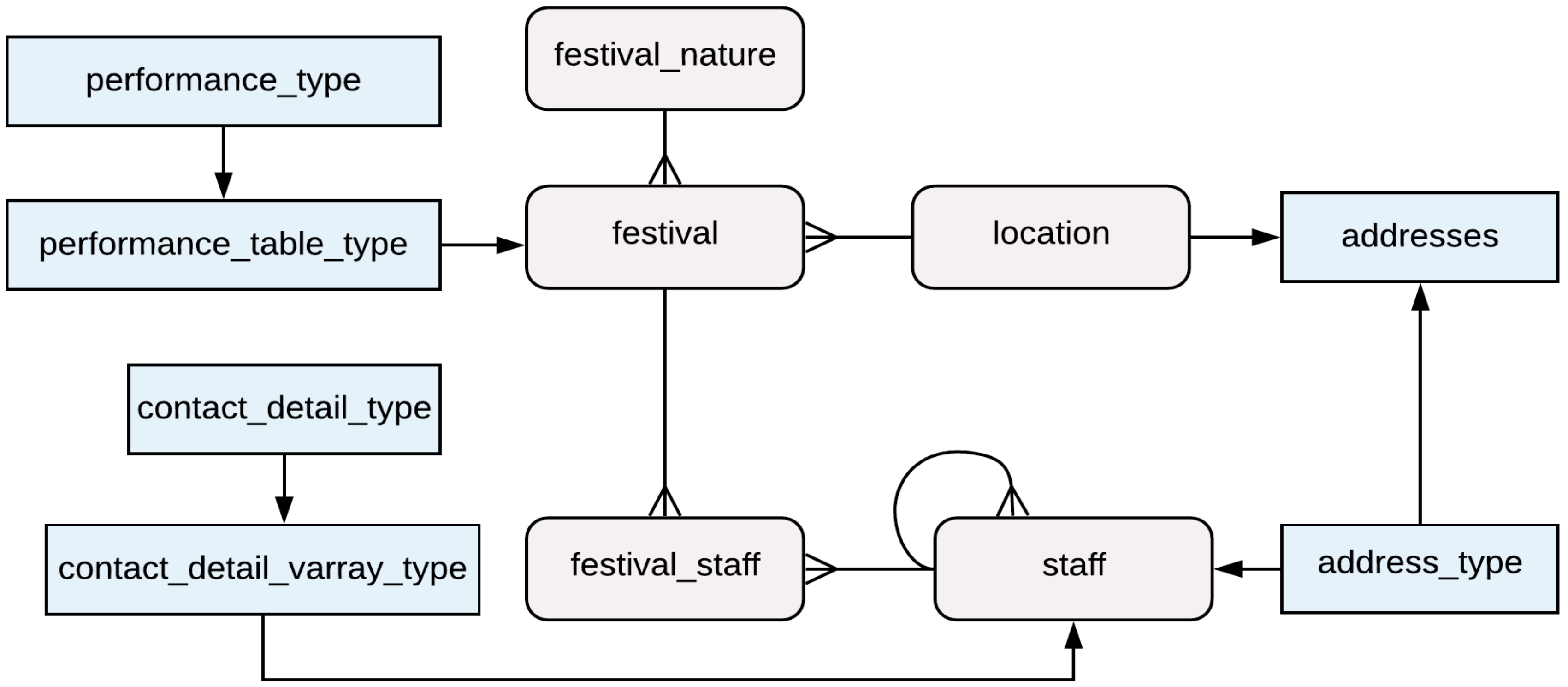
Database II

JANUARY 1, 2019

GROUP 4
Bhuwan Khatiwada, Bishownath Dhakal, Diwas Lamsal, Sandip Kumar Subba

Contents

Schema



performance_type
festival_nature
performance_table_type
festival
location
addresses
contact_detail_type
contact_detail_varray_type
festival_staff
staff
address_type

TABLE SPECIFICATION

Table Specification for Object Types

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT
address_type			
street	VARCHAR2(30)		
city	VARCHAR2(30)		
country	VARCHAR2(30)		

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT
contact_detail_type			
contact_number	VARCHAR2(15)		
number_type	VARCHAR2(30)		‘LANDLINE’

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT
performance_type			
name	VARCHAR2(30)		
artirst	VARCHAR2(30)		
genre	VARCHAR2(30)		

Table Specification for Relational Tables

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
staff				
staff_id	NUMBER(6)	pk_staff		seq_staff_id
firstname	VARCHAR2(30)	UPPER		
lastname	VARCHAR2(30)	UPPER		
gender	CHECK IN (M, F)		‘M’	
contact	contact_detail_varray_type			
current_address	address_type			
permanent_address	address_type			
email	VARCHAR2(60)			
*leader	NUMBER(6)	fk_s_staff		
salary	NUMBER(10,2)			

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
festival_natures				
festival_nature_id	NUMBER(6)	pk_festival_natures		seq_festival_nature_id

name	VARCHAR2(30)	UPPER		
target_audience	VARCHAR2(30)	UPPER	‘ADULT’	

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
locations				
location_id	NUMBER(6)	pk_locations		seq_location_id
address	REF(addresses)			
capacity	NUMBER(5)			
price	NUMBER(10,2)			

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
festivals				
*festival_nature_id	NUMBER(6)	pk_festivals, fk_f_festival_natures		
*location_id	NUMBER(6)	pk_festivals, fk_f_locations		
festival_name	VARCHAR2(30)	UPPER		
performance	performance_table_type			

ATTRIBUTE	DATATYPE	CONSTRAINT	DEFAULT	SEQUENCE
festival_staff				
festival_staff_id	NUMBER(6)	pk_festival_		seq_festival_

		staff		staff_id
*staff_id	NUMBER(6)	fk_fs_staff		
*festival_nature_id	NUMBER(6)	fk_fs_festivals, NOT NULL		
*location_id	NUMBER(6)	fk_fs_festivals, NOT NULL		

Entity description

Entity	**Description**
staff	Staff details
festival_nature	Details of festival’s type
festival	Details of festival
festival_staff	Details of staff and festival
location	Location

Automation Strategies

Automation Strategies for Functions and Procedures

S.N.	Procedural Element Type	Name	Parameter (Type)	Purpose	Applied Table	Return Type
1	Function	func_count_staff	-	-To count all data inserted into staff table	staff	NUMBER
2	Function	func_count_staff_salary_	in_salary NUMBER (Type = IN)	-To count all staffs' salaries for staff whose salary is greater than prescribed salary. (i.e. salary > in_salary)	staff	NUMBER
3	Function	func_count_festivals	-	-To count all data inserted into festivals table	festivals	NUMBER
4	Function	func_exp_location	-	-To select maximum price for location	locations	NUMBER

				available in locations table.		
5	Function	func_chp_location	-	-To select minimum price for location available in locations table.	locations	NUMBER
6	Function	func_increase_salary	-in_current_salary NUMBER (Type = IN) -in_percent NUMBER (Type = IN)	-To update salary of staff by given percentage.	staff	NUMBER
7	Function	func_discount_location	-in_current_price NUMBER (Type = IN) -in_percent NUMBER (Type = IN)	-To discount price of concerned location by given percentage.	locations	NUMBER
8	Function	func_username	in_staff_id staff.staff_id%Type (Type = IN)	-To provide username for staff stored in staff table. Staff's username is provided by concatenating different letter for their names.	staff	VARCHAR 2
9	Function	func_password	staff.staff_id%Type (Type = IN)	-To provide password for staff stored in staff table.	staff	VARCHAR 2

				Along with letter from their names, date has also been concatenated to create password.		
10	Function	func_check_string_num	in_string VARCHAR2 (Type = IN)	-To check if string passed through function contains numeric value or not.	-	NUMBER
11	Procedur e	proc_insert_addresses	-in_street VARCHAR2:=NULL (Type = IN) -in_city VARCHAR2:=NULL (Type = IN) -in_country VARCHAR2:=NULL (Type = IN)	-Procedure designed to insert data into object table “addresses”.	addresses (object table)	-
12	Procedur e	proc_insert_festival_natur es	-in_name festival_natures.name%TYPE (Type = IN) -in_targeted_audience festival_natures.targeted_audienc e %TYPE (Type = IN)	-Procedure designed to insert data into table “festival_nature s”.	festival_natur es	-

13	Procedure	proc_insert_locations	-in_capacity locations.capacity%TYPE (Type = IN) -in_price locations.price%TYPE (Type = IN) -in_address_ref VARCHAR2 (Type = IN)	-Procedure designed to insert data into table "locations".	locations	-
14	Procedure	proc_insert_staff	-in_firstname VARCHAR2 (Type = IN) -in_lastname VARCHAR2 (Type = IN) -in_gender CHAR (Type = IN) -in_num_t1VARCHAR2 (Type = IN) -in_type_t1 VARCHAR2 (Type = IN) -in_num_t2 VARCHAR2 (Type = IN) -in_type_t2 VARCHAR2 (Type = IN)	-Procedure designed to insert complete data of staff into table "staffs".	staffs	-

			-in_street_c VARCHAR2 (Type = IN) -in_city_c VARCHAR2 (Type = IN) -in_country_c VARCHAR2 (Type = IN) -in_street_p VARCHAR2 (Type = IN) -in_city_p VARCHAR2 (Type = IN) -in_country_p VARCHAR2 (Type = IN) -in_email VARCHAR2 (TYPE = IN) -in_leader VARCHAR2 (Type = IN) -in_salary NUMBER (Type = IN)			
15	Procedur e	proc_insert_festivals	-in_festival_nature_id NUMBER (Type = IN) -in_location_id NUMBER (Type = IN)	-Procedure designed to insert complete data of festival	festivals	-

			-in_name VARCHAR2 (Type = IN) -in_p1_name VARCHAR2 (Type = IN) -in_p1_artist VARCHAR2 (Type = IN) -in_p1_genre VARCHAR2 (Type = IN) -in_p2_name VARCHAR2 (Type = IN) -in_p2_artist VARCHAR2 (Type = IN) -in_p2_genre VARCHAR2 (Type = IN)	into table “festivals”.		
16	Procedure	proc_insert_festival_staff	-in_staff_id festival_staff.staff_id%TYPE (TYPE = IN) -in_festival_nature_id festival_staff.festival_nature_id%TYPE (Type = IN)	-Procedure designed to insert complete data of festival_staff into table “festival_staff”.	festival_staff	-

			-in_location_id festival_staff.location_id%TYPE (Type = IN)			
17	Procedur e	proc_reset_seq	p_seq_name VARCHAR2 (TYPE = IN)	-Procedure created to reset prescribed sequence to start value (beginning value)	-	-
18	Procedur e	proc_count_staff	-	-Procedure to count all data in staff table. -This procedure also execute function that counts data in staff table	staff	-
19	Procedur e	proc_count_staff_salary	in_salary NUMBER (Type = IN)	-Procedure to count all staffs whose salary is greater than prescribed value. -This procedure execute function that counts staff having salary greater than	staff	-

				prescribed value.		
20	Procedure	proc_count_festivals	-	-Procedure to count all data currently available in festivals table. -This procedure will also execute function that counts all data available in festivals table.	festivals	-
21	Procedure	proc_exp_location	-	-Procedure to execute function that return most expensive location in accordance to costs.	locations	-
22	Procedure	proc_chp_location	-	-Procedure to execute function that return cheapest location in accordance to costs.	locations	-
23	Procedure	proc_festival_detail	in_festival_name IN festivals.festival_name%TYPE (TYPE = IN)	-Procedure designed to display complete	festivals	-

				detail of festival whose name is passed in parameter.		
24	Procedure	proc_show_staff_address	in_staff_id staff.staff_id%TYPE (Type = IN)	-Procedure to display both permanent and current addresses of staff.	staff	-
25	Procedure	proc_increase_salary	-in_staff_id staff.staff_id%TYPE (Type = IN) -in_percent NUMBER (Type = IN)	-Procedure to execute function that return updated salary of staff after increasing it by percentage prescribed.	staff	-
26	Procedure	proc_discount_location	-in_location_id locations.location_id%TYPE (Type = IN) -in_precent NUMBER (Type = IN)	-Procedure to execute function that return price of location after making a discount of percentage prescribed.	locations	-
27	Procedure	proc_username_password	in_staff_id staff.staff_id%TYPE (Type = IN)	-Procedure to execute functions that return both username and	staff	-

				password for prescribed staff		
28	Procedure	proc_location_capacity_ck	-in_location_id NUMBER (Type = IN) -in_group_size NUMBER (Type = IN)	-Procedure to check for available capacity for people to fit in the location. -This procedur calculated both remaining number of seats after adding group of people in capacity available. -If group cannot fit in the capacity, this procedure also displayed number of additional seats required for all members of group to be fitted in the location.	locations	-

29	Procedure	proc_staff_firstname	in_staff_id NUMBER (Type = IN)	-Procedure to display staff first name in prescribed order. Example = RAM Output = R A M	staff	-
30	Procedure	proc_staff_surname	in_staff_id NUMBER (Type = IN)	-Procedure to display staff last name in prescribed order. Example = DHAKAL Output = D H A K A L	staff	-
31	Procedure	proc_ck_func_check_string_num	-	-Procedure to check if any string contains any numeric value or not.	-	-

Automation Strategies for Trigger

S.N.	Procedural Element Type	Name	Error/Messages Generated	Purpose	Applied Table	Firing Granularity
1	Trigger	trig_check_address	RAISE_APPLICATION_ERROR (-20001,'ERROR! THE PROVIDED ADDRESS ALREADY EXISTS')	-Trigger created to check if same address has been entered again or not.	addresses (object table)	Before
2	Trigger	trig_staff_email_ck	RAISE_APPLICATION_ERROR (-20001,'ERROR! INVALID EMAIL FORMAT')	-Trigger to check if correct format of email is being inserted into staffs table. -Trigger check '@' and '.' Sign in value to be entered.	staff	Before
3	Trigger	trig_staff_name_ck	*(For first name)* WHEN 0 - DBMS_OUTPUT.PUT_LINE ('VALID FIRSTNAME') WHEN 1- RAISE_APPLICATION_ERROR (-20001,'ERROR! INVALID	-Trigger to check if both first name and last name of staff is correct. If found incorrect, error message is generated.	staff	Before

			FIRSTNAME! NAME CANNOT CONTAIN NUMBERS') *(For last name)* WHEN 0 - DBMS_OUTPUT.PUT_LINE ('VALID LASTNAME') WHEN 1 – RAISE_APPLICATION_ERROR (-20001,'ERROR! INVALID LASTNAME! NAME CANNOT CONTAIN NUMBERS')			
4	Trigger	trig_del_staff	DBMS_OUTPUT.PUT_LINE ('YOU DELETED THE STAFF '||:OLD.firstname || ' ' || :OLD.lastname)	-Trigger that displays message when any member from staff table is deleted.	staff	After
5	Trigger	trig_ck_location	RAISE_APPLICATION_ERROR (-20001,'ERROR! CANNOT HAVE VALUES LESS THAN ZERO')	-Trigger to check if capacity or price of location is being entered in negative value. -If entered in negative value, error message is generated.	locations	Before

6	Trigger	trig_festival_natures	*WHILE INSERTING* DBMS_OUTPUT.PUT_LINE ('YOU INSERTED THE FESTIVAL NATURE '||:NEW.name || ' WITH FESTIVAL_NATURE_ID: '|| :NEW.festival_nature_id) *WHILE UPDATING* DBMS_OUTPUT.PUT_LINE ('YOU UPDATED THE FESTIVAL NATURE '||:NEW.name || ' WITH FESTIVAL_NATURE_ID: '|| :OLD.festival_nature_id) *WHILE DELETING* DBMS_OUTPUT.PUT_LINE ('YOU DELETED THE FESTIVAL NATURE '||:OLD.name || ' WITH FESTIVAL_NATURE_ID: '|| :OLD.festival_nature_id)	-Trigger designed to fire message when inserting, updating and deleting in festival_natures table.	festival_natures	After
7	Trigger	trig_festival_staff	*WHILE INSERTING* DBMS_OUTPUT.PUT_LINE ('YOU INSERTED A RECORD. FESTIVAL_STAFF_ID: '|| :NEW.festival_staff_id)	-Trigger designed to fire message when inserting, updating and	festival_staff	After

			WHILE UPDATING DBMS_OUTPUT.PUT_LINE ('YOU UPDATED A RECORD. FESTIVAL_STAFF_ID: '|| :OLD.festival_staff_id) *WHILE DELETING* DBMS_OUTPUT.PUT_LINE ('YOU DELETED A RECORD. FESTIVAL_STAFF_ID: '|| :OLD.festival_staff_id)	deleting in festival_staff table.		

Automation Strategies for Cursor

S.N.	Procedural Element Type	Name	Parameter (Type)	Purpose	Applied Table	Cursor Type
1	Cursor	proc_del_address_cursor	in_street VARCHAR2 (Type = IN)	-Cursor created to delete all data from object table "addresses" having same street name as one passed in parameter.	addresses (object table)	Implicit Cursor
2	Cursor	proc_num_locations_price	in_price NUMBER (Type = IN)	This cursor adds 50 to the price of all the locations found below the provided price	locations	Implicit Cursor
3	Cursor	proc_view_addresses_cursor	out_num_records NUMBER (Type = OUT)	-Cursor to display all detail of addresses data inserted in addresses table.	locations	Explicit Cursor
4	Cursor	proc_view_festival_natures_cursor	out_num_records NUMBER (Type = OUT)	-Cursor to display all details of	festival_natures	Explicit Cursor

				festival nature from festival_nature table.		
5	Cursor	proc_view_staff_contact_cursor	-	-Cursor to display all details of staff from staff table.	staff	Explicit Cursor
6	Cursor	proc_view_staff_salary_weekly_cursor	in_staff_id NUMBER (Type = IN)	-Cursor to display weekly salary of staff in staff table.	staff	Explicit Cursor

Test Plan

Given below is complete test plan for script written. Test for different automation strategies has been carried out in different, yet suitable ways.

Table Test Plan

S.N.	Test Case	Tester	Supervision	Test Duration	Testing Condition
1	Creation of table	-Bishownath Dhakal -Sandip Subba	-Bhuwan Khatiwada -Diwas Lamsal		-Queried all table before creating new table. -Queried all table again after table creation.
2	Addition of Constraints to Table	-Bhuwan Khatiwada -Sandip Subba	-Bishownath Dhakal -Diwas Lamsal		-Inserted data using wrong value on column with constraints. -Inserted data using correct value on column with constraints.
3	-Inserting, into Table -Deleting data from Table -Updating data present in Table.	-Diwas Lamsal -Bishownath Dhakal	-Bhuwan Khatiwada -Sandip Subba		-Inserting data with default value left blank. -Inserting data using correct value. -Tried to insert into table using data of mis matched datatype.
4	Dropping Constraints	-Bishownath Dhakal	-Bhuwan Khatiwada		-All constraints are queried to ensure they are present in system. -All constraints are dropped and then constraints are queried to ensure their drop.

5	Dropping Table	-Sandip Subba	-Diwas Lamsal		-All Tables are queried to ensure they are present in system. -All Tables are dropped and then Tables are queried to ensure their drop.

Function, Cursors and Procedural Test Plan

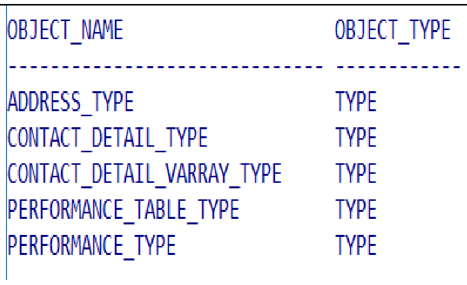
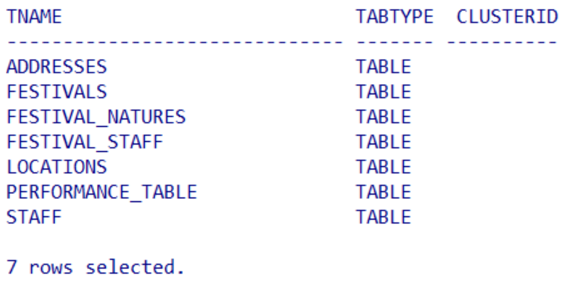
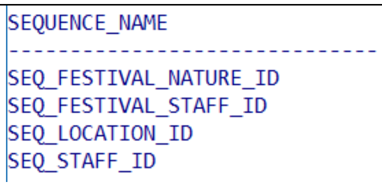
S.N.	Test Case	Tester	Supervision	Test Duration	Testing Condition
1	Creation of Functions, Cursors and Procedures	-Sandip Subba	-Diwas Lamsal		-All functions and procedure are created one by one. -Show Error was used to check for error contained in function and procedure.
2	Executing Procedures, Cursors and Functions.	-Sandip Subba	-Diwas Lamsal		-Queried carried out to check for condition before executing procedures. -Procedure are executed by passing both correct and invalid parameters.
3	Dropping all Procedures, Cursors and Functions.	-Bhuwan Khatiwada	-Diwas lamsal		-Queried carried out to check all available cursors, functions, and procedures. -Functions are dropped and procedure are executed so ensure if function is deleted or not. -Dropping both procedure and function in defined order. (Cursors are also defined n procedures)

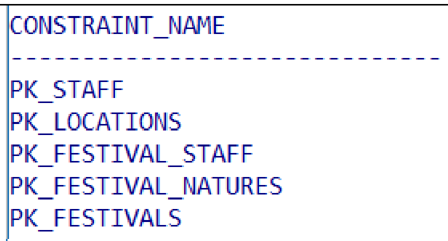
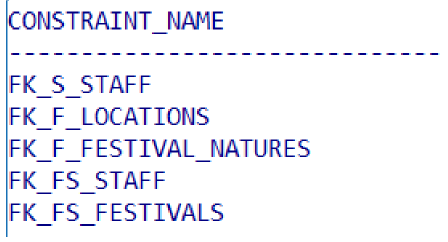
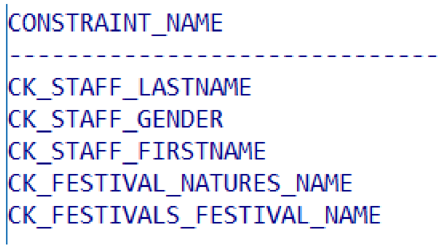
Trigger Test Plan

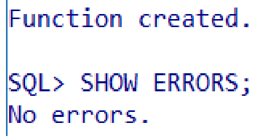
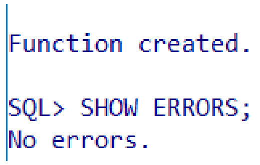
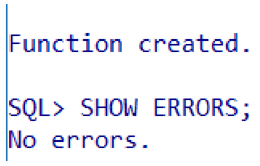
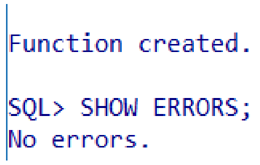
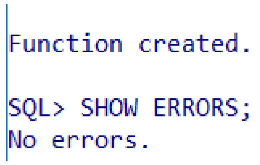
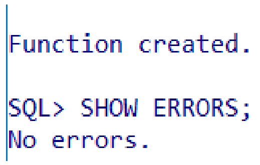
S.N.	Test Case	Tester	Supervision	Test Duration	Testing Condition
1	Creation of Triggers	-Sandip Subba	-Diwas Lamsal		-All triggers are created one by one. -Show Error was used to check for error contained in triggers.
2	Executing of Triggers	-Diwas Lamsal	-Bhuwan Khatiwada		Mainly four test cases have been tested for triggers. -Condition where trigger had to be fired and if trigger is being fired or not. -Condition where trigger was not to be fired and if trigger is being fired or not. -if trigger is being fired and changes is being made in system or not. -if no trigger is fired and yet changes are being carried out, even if those change was not meant to be carried out.

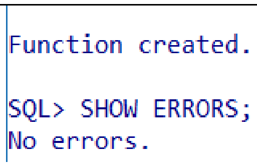
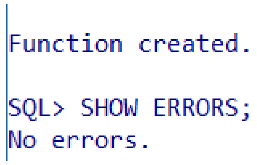
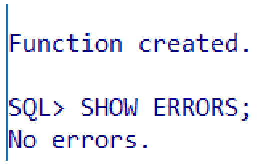
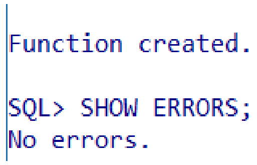
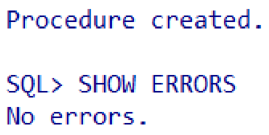
Test Cases

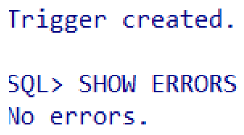
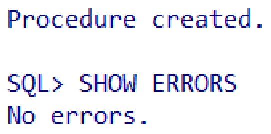
Test for all database syntax has been carried out on basis of test plan mentioned above. Table below shows complete test plan for this assignment.

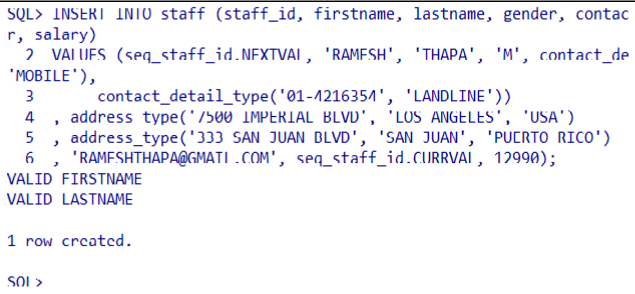
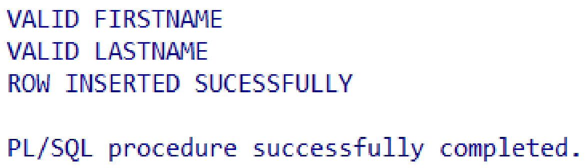
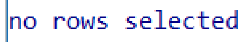
S. N.	DESCRIPTION	Expected Output	Actual Output	Remarks
1	All Type Creation. - Querying created type. COLUMN object_name FORMAT A30; COLUMN object_type FORMAT A12; SELECT object_name, object_type FROM user_objects WHERE object_type = 'TYPE';	5 rows selected.	 OBJECT_NAME OBJECT_TYPE ------------------------------ ------------ ADDRESS_TYPE TYPE CONTACT_DETAIL_TYPE TYPE CONTACT_DETAIL_VARRAY_TYPE TYPE PERFORMANCE_TABLE_TYPE TYPE PERFORMANCE_TYPE TYPE	√
2	All table Creation. - Querying created table. SELECT tname FROM tab;	7 rows selected.	 TNAME TABTYPE CLUSTERID ------------------------------ ------- ---------- ADDRESSES TABLE FESTIVALS TABLE FESTIVAL_NATURES TABLE FESTIVAL_STAFF TABLE LOCATIONS TABLE PERFORMANCE_TABLE TABLE STAFF TABLE 7 rows selected.	√
3	All sequence creation. - Querying created Sequence SELECT sequence_name FROM user_sequences;	4 rows selected.	 SEQUENCE_NAME ------------------------------ SEQ_FESTIVAL_NATURE_ID SEQ_FESTIVAL_STAFF_ID SEQ_LOCATION_ID SEQ_STAFF_ID	√

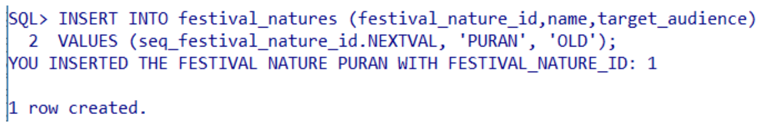
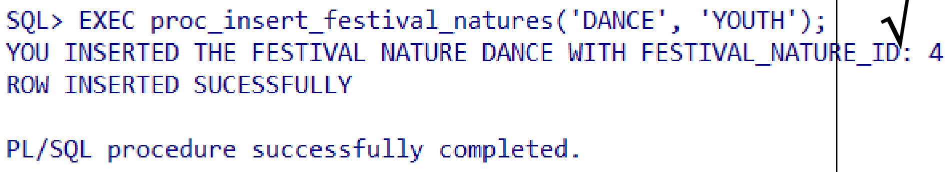
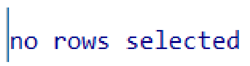
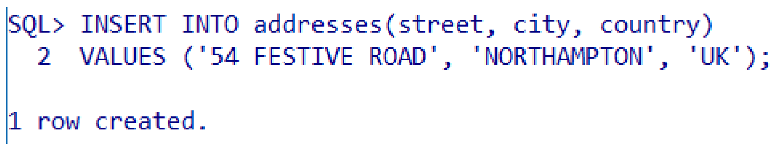
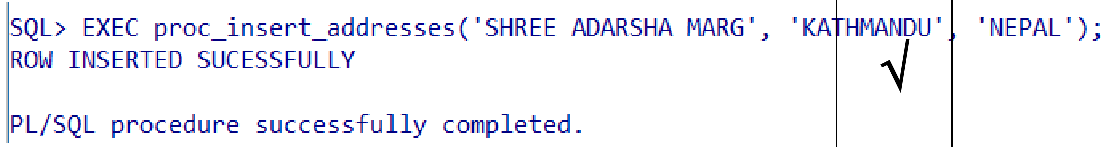
4	Addition of primary key, foreign key and check constraint on tables created above. - Querying all primary key created. SELECT constraint_name FROM user_constraints WHERE constraint_name LIKE 'PK%'; - Querying all foreign key created. SELECT constraint_name FROM user_constraints WHERE constraint_name LIKE 'FK%'; - Querying all check constraints created. SELECT constraint_name FROM user_constraints WHERE constraint_name LIKE 'CK%';	5 rows selected. 5 rows selected.	 CONSTRAINT_NAME ------------------------------ PK_STAFF PK_LOCATIONS PK_FESTIVAL_STAFF PK_FESTIVAL_NATURES PK_FESTIVALS  CONSTRAINT_NAME ------------------------------ FK_S_STAFF FK_F_LOCATIONS FK_F_FESTIVAL_NATURES FK_FS_STAFF FK_FS_FESTIVALS  CONSTRAINT_NAME ------------------------------ CK_STAFF_LASTNAME CK_STAFF_GENDER CK_STAFF_FIRSTNAME CK_FESTIVAL_NATURES_NAME CK_FESTIVALS_FESTIVAL_NAME	√ √ √

5	Creating all functions - func_count_staff	All function created with no compilation errors.	 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_count_staff_salary		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_count_festivals		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_exp_location		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_chp_location		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_increase_salary		 Function created. SQL> SHOW ERRORS; No errors.	√

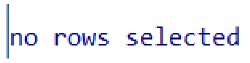
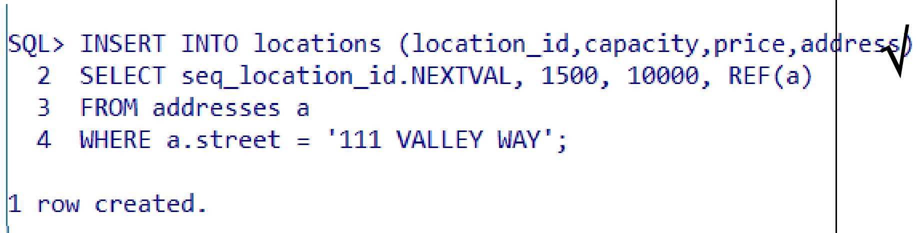
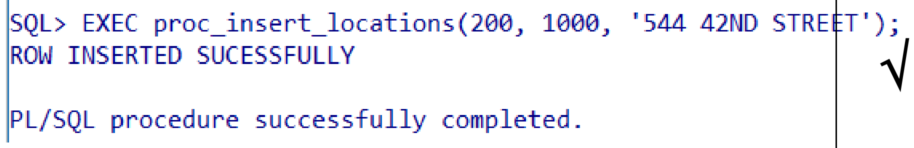
	- func_discount_location		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_username		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_password		 Function created. SQL> SHOW ERRORS; No errors.	√
	- func_check_string_num		 Function created. SQL> SHOW ERRORS; No errors.	√
6	Creating all procedures - proc_insert_addresses - proc_insert_festival_natures - proc_insert_locations - proc_insert_staff - proc_insert_festivals - proc_insert_festival_staff - proc_reset_seq	All procedures created with no compilation errors.	 Procedure created. SQL> SHOW ERRORS No errors. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output.	

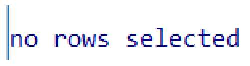
	- proc_count_staff - proc_count_staff_salary - proc_count_festivals - proc_exp_location - proc_chp_location - proc_festival_detail - proc_show_staff_addres s - proc_increase_salary - proc_discount_location - proc_username_passwor d - proc_location_capacity_ ck - proc_staff_firstname - proc_staff_surname		Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output. Same as Expected output.	√
7	Creating all triggers - trig_check_address - trig_staff_email_ck - trig_staff_name_ck - trig_del_staff - trig_ck_location - trig_festival_natures - trig_festival_staff	All triggers created with no complication errors.	 Trigger created. SQL> SHOW ERRORS No errors. Same as expected output. Same as expected output. Same as expected output. Same as expected output. Same as expected output. Same as expected output.	√
8	Creating all procedures that contains cursors.	All procedures with cursors created with no	 Procedure created. SQL> SHOW ERRORS No errors.	

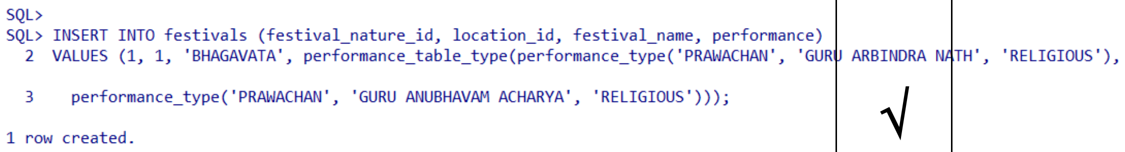
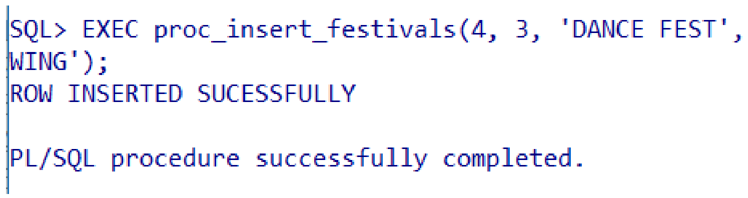
	- proc_del_address_cursor; - proc_num_locations_price; - proc_view_addresses_cursor; - proc_view_festival_natures_cursor; - proc_view_staff_contact_cursor; - proc_view_staff_salary_weekly_cursor	compilation errors.	Same as expected output. Same as expected output. Same as expected output. Same as expected output. Same as expected output.	√
9	Inserting into staff tables respectively. - Inserting in normal way. Select staff_id, firstname, lastname from staff; - Inserting using procedures. Example – executing procedure name -proc_insert_staff	Row inserted successfully. Row inserted successfully.	 SQL> INSERT INTO staff (staff_id, firstname, lastname, gender, contac r, salary) 2 VALUES (seq_staff_id.NEXTVAL, 'RAMESH', 'THAPA', 'M', contact_de 'MOBILE'), 3 contact_detail_type('01-4216354', 'LANDLINE')) 4 , address_type('7500 IMPERIAL BLVD', 'LOS ANGELES', 'USA') 5 , address_type('333 SAN JUAN BLVD', 'SAN JUAN', 'PUERTO RICO') 6 , 'RAMESHTHAPA@GMAIL.COM', seq_staff_id.CURRVAL, 12990); VALID FIRSTNAME VALID LASTNAME 1 row created. SQL>  VALID FIRSTNAME VALID LASTNAME ROW INSERTED SUCESSFULLY PL/SQL procedure successfully completed.	√ √
10	Inserting into festival_natures tables respectively. - Verifying empty table.		 no rows selected	√

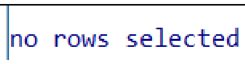
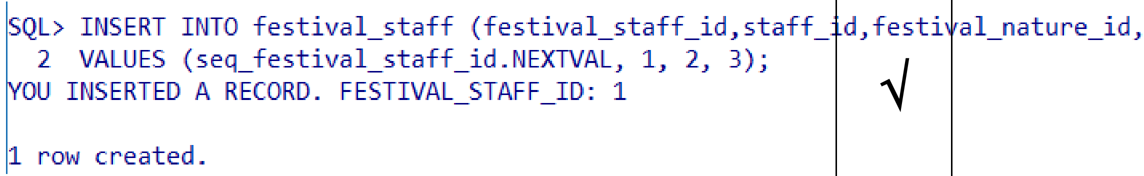
	SELECT * FROM festival_natures - Inserting in table in normal method. - Inserting using procedure. Example – executing procedure -EXEC proc_insert_festival_natures ('CONCERT', 'YOUTH')	No rows selected. Row inserted successfully. Row inserted successfully.	 SQL> INSERT INTO festival_natures (festival_nature_id,name,target_audience) 2 VALUES (seq_festival_nature_id.NEXTVAL, 'PURAN', 'OLD'); YOU INSERTED THE FESTIVAL NATURE PURAN WITH FESTIVAL_NATURE_ID: 1 1 row created.  SQL> EXEC proc_insert_festival_natures('DANCE', 'YOUTH'); YOU INSERTED THE FESTIVAL NATURE DANCE WITH FESTIVAL_NATURE_ID: 4 ROW INSERTED SUCESSFULLY PL/SQL procedure successfully completed.	√ √
11	Inserting into addresses object tables respectively. - Verifying empty table. SELECT street, city, country FROM addresses; - Inserting in table in normal method. - Inserting using procedure. Example – executing procedure	No rows selected. Row inserted successfully. Row inserted successfully.	 no rows selected  SQL> INSERT INTO addresses(street, city, country) 2 VALUES ('54 FESTIVE ROAD', 'NORTHAMPTON', 'UK'); 1 row created.  SQL> EXEC proc_insert_addresses('SHREE ADARSHA MARG', 'KATHMANDU', 'NEPAL'); ROW INSERTED SUCESSFULLY PL/SQL procedure successfully completed.	√ √ √

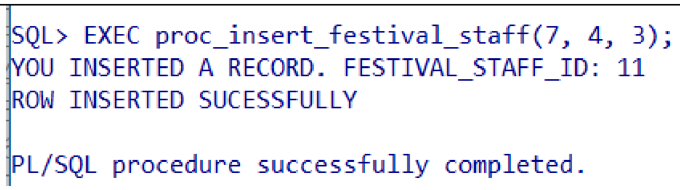
	- EXEC proc_insert_addresses ('SHREE ADARSHA MARG', 'KATHMANDU', 'NEPAL')			

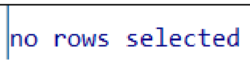
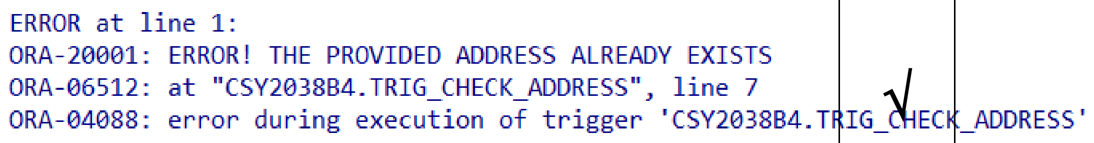
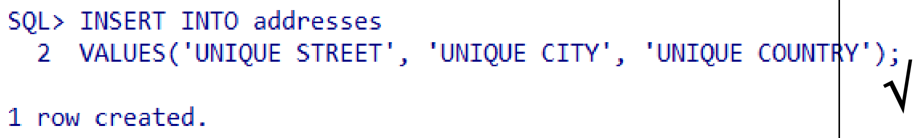
12	Inserting into locations tables respectively.
	Requires references of object table addresses.
	- Verifying empty table. SELECT street, city, country FROM locations;
	
	- Inserting in table in normal method.
	
	- Inserting using procedure. Example – executing procedure -EXEC proc_insert_locations (1200, 20500, ‘177 AIRPORT ROAD’);
		No rows selected.
		
		Row inserted successfully.
		Row inserted successfully.
			
			`no rows selected`
			
			
			`SQL> INSERT INTO locations (location_id,capacity,price,address)`
			`  SELECT seq_location_id.NEXTVAL, 1500, 10000, REF(a)`
			`  FROM addresses a`
			`  WHERE a.street = '111 VALLEY WAY';`
			
			`1 row created.`
			
			
			`SQL> EXEC proc_insert_locations(200, 1000, '544 42ND STREET');`
			`ROW INSERTED SUCESSFULLY`
			
			`PL/SQL procedure successfully completed.`
				√
				
				√
				
				√

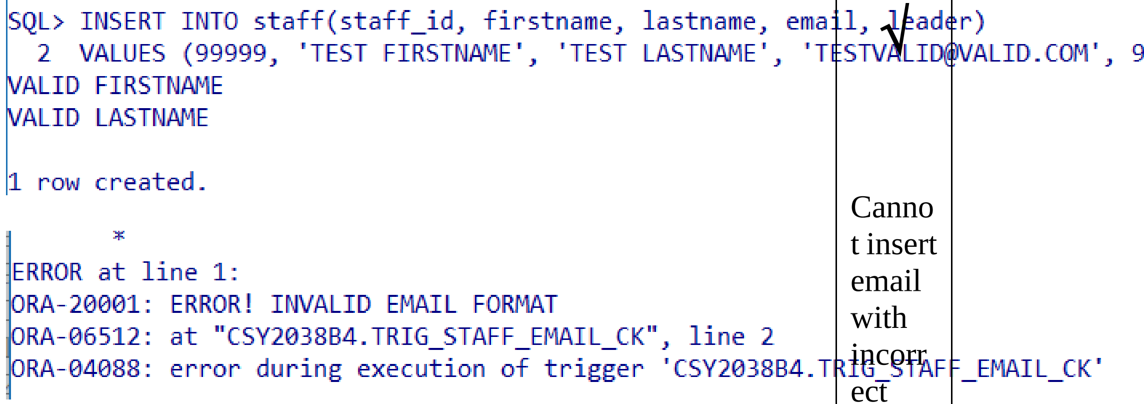
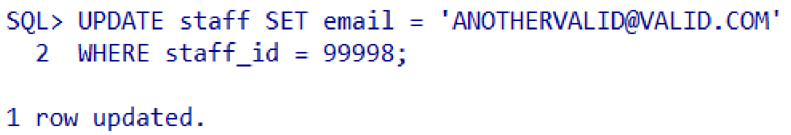
13	Inserting into festivals tables respectively. - Verifying empty table.	No rows selected.	 `no rows selected`	√

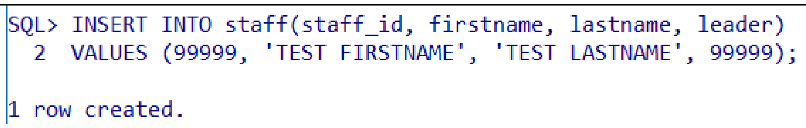
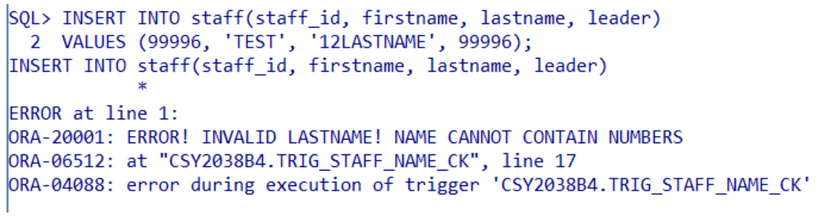
	SELECT * FROM festivals;
	
	- Inserting in table in normal method.
	
	- Inserting using procedure.
	Example – executing procedure
	-EXEC proc_insert_festivals (2, 3, 'SCREAM FEST', 'SINGING', 'NEPATHYA', 'ROCK', 'MUSIC', 'ANTIM GRAHAN', 'METAL');
		Row inserted successfully.
		
		Row inserted successfully.
			
			`SQL>`
			`SQL> INSERT INTO festivals (festival_nature_id, location_id, festival_name, performance)`
			`  2  VALUES (1, 1, 'BHAGAVATA', performance_table_type(performance_type('PRAWACHAN', 'GURU ARBINDRA NATH', 'RELIGIOUS'),`
			`  3    performance_type('PRAWACHAN', 'GURU ANUBHAVAM ACHARYA', 'RELIGIOUS')));`
			`1 row created.`
			
			
			`SQL> EXEC proc_insert_festivals(4, 3, 'DANCE FEST',`
			`WING');`
			`ROW INSERTED SUCESSFULLY`
			
			`PL/SQL procedure successfully completed.`
				√
				
				√

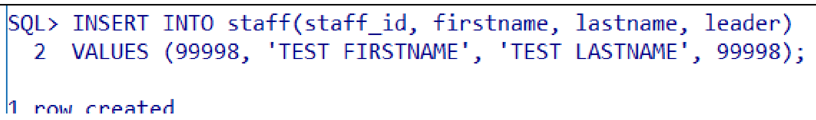
14	Inserting into festival_staff tables respectively.
	- Verifying empty table.
	SELECT * FROM festival_staff;
	
	- Inserting in table in normal method.
		No rows selected.
		
		Row inserted successfully.
			
			`no rows selected`
			
			
			`SQL> INSERT INTO festival_staff (festival_staff_id,staff_id,festival_nature_id,`
			`  2  VALUES (seq_festival_staff_id.NEXTVAL, 1, 2, 3);`
			`YOU INSERTED A RECORD. FESTIVAL_STAFF_ID: 1`
			
			`1 row created.`
				√
				
				√

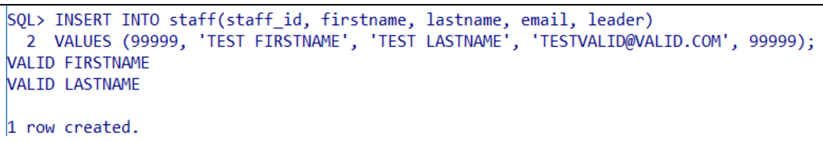
	- Inserting using procedure.
	Example – executing procedure
	-EXEC proc_insert_festival_staff(7, 4, 3)
			
			`SQL> EXEC proc_insert_festival_staff(7, 4, 3);`
			`YOU INSERTED A RECORD. FESTIVAL_STAFF_ID: 11`
			`ROW INSERTED SUCESSFULLY`
			
			`PL/SQL procedure successfully completed.`
				√

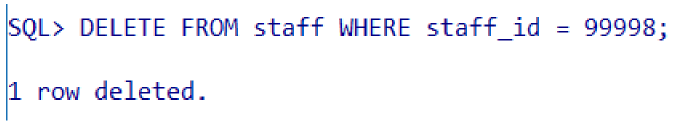
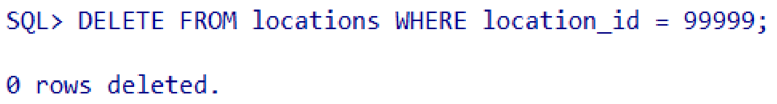
15	TEST trig_check_address
	- Inseting dummy data.
	
	- Fires where needed
	
	- Does not fire where not needed
		1 row created.
		
		Error is generated preventing the row from being inserted into the addressed table
		
		No errors are shown and the row is inserted into the addresses table
			
			`no rows selected`
			
			
			`ERROR at line 1:`
			`ORA-20001: ERROR! THE PROVIDED ADDRESS ALREADY EXISTS`
			`ORA-06512: at "CSY2038B4.TRIG_CHECK_ADDRESS", line 7`
			`ORA-04088: error during execution of trigger 'CSY2038B4.TRIG_CHECK_ADDRESS'`
			
			
			`SQL> INSERT INTO addresses`
			`  2  VALUES('UNIQUE STREET', 'UNIQUE CITY', 'UNIQUE COUNTRY');`
			
			`1 row created.`
				√
				
				√
				
				√

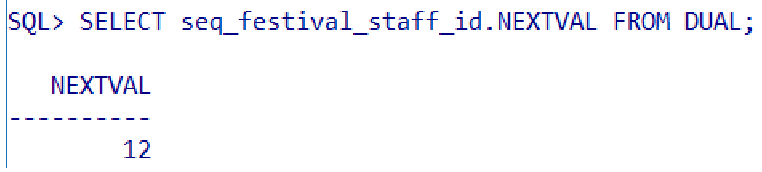
16	TEST trig_staff_email_ck
	- First Insert a valid row
	
	- 1 Fires where needed
	
	- Does not fire where not needed
		1 row created.
		
		for each insert or update attempt: Error message shown
			
			`SQL> INSERT INTO staff(staff_id, firstname, lastname, email, leader)`
			`  2  VALUES (99999, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTVALID@VALID.COM', 9`
			`VALID FIRSTNAME`
			`VALID LASTNAME`
			
			`1 row created.`
			
			`       *`
			`ERROR at line 1:`
			`ORA-20001: ERROR! INVALID EMAIL FORMAT`
			`ORA-06512: at "CSY2038B4.TRIG_STAFF_EMAIL_CK", line 2`
			`ORA-04088: error during execution of trigger 'CSY2038B4.TRIG_STAFF_EMAIL_CK'`
			
			
			`SQL> UPDATE staff SET email = 'ANOTHERVALID@VALID.COM'`
			`  2  WHERE staff_id = 99998;`
			
			`1 row updated.`
				√
				
				Cannot insert email with incorrect format.
				
				√

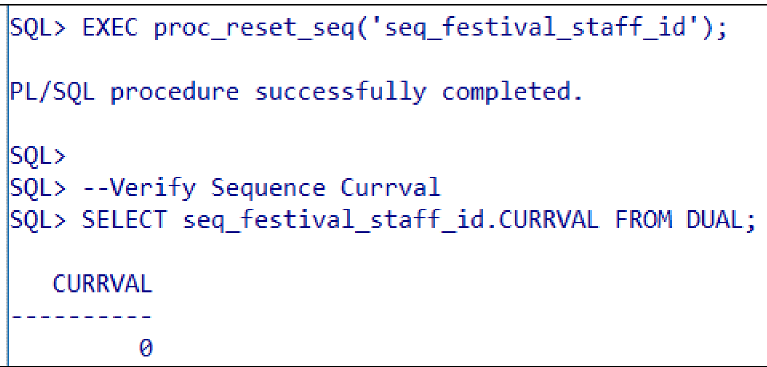
17	TEST trig_staff_name_ck
	- First Insert a valid row
	
	- Fires where needed
	
	- Does not fire where not needed
		1 row created.
		
		for each insert or update attempt: Error message shown saying invalid firstname or lastname
		
		The row is inserted and
			
			`SQL> INSERT INTO staff(staff_id, firstname, lastname, leader)`
			`  2  VALUES (99999, 'TEST FIRSTNAME', 'TEST LASTNAME', 99999);`
			
			`1 row created.`
			
			
			`SQL> INSERT INTO staff(staff_id, firstname, lastname, leader)`
			`  2  VALUES (99996, 'TEST', '12LASTNAME', 99996);`
			`INSERT INTO staff(staff_id, firstname, lastname, leader)`
			`            *`
			`ERROR at line 1:`
			`ORA-20001: ERROR! INVALID LASTNAME! NAME CANNOT CONTAIN NUMBERS`
			`ORA-06512: at "CSY2038B4.TRIG_STAFF_NAME_CK", line 17`
			`ORA-04088: error during execution of trigger 'CSY2038B4.TRIG_STAFF_NAME_CK'`
				√
				
				√

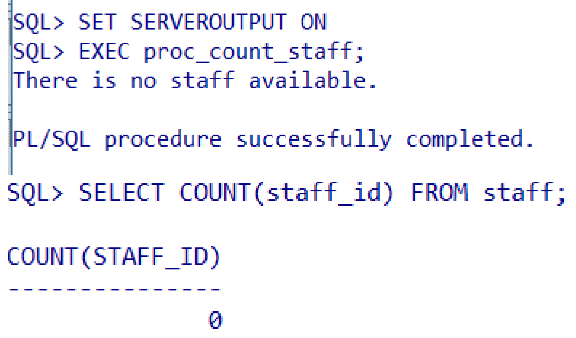
		updated succesfully, also messages showing VALID FIRSTNAME and VALID LASTNAME	 `SQL> INSERT INTO staff(staff_id, firstname, lastname, leader)`
			`  2  VALUES (99998, 'TEST FIRSTNAME', 'TEST LASTNAME', 99998);`
			
			`1 row created`
				√

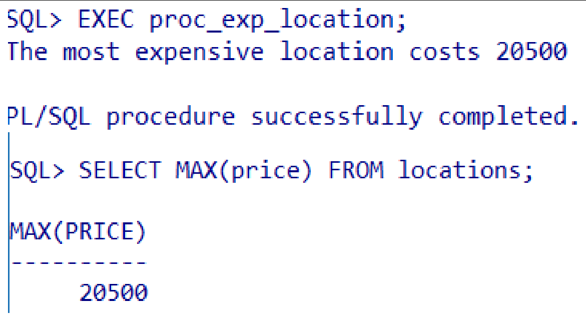
18	TEST trig_del_staff
	- First Insert a valid row
		1 row created.	 `SQL> INSERT INTO staff(staff_id, firstname, lastname, email, leader)`
			`  2  VALUES (99999, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTVALID@VALID.COM', 99999);`
			`VALID FIRSTNAME`
			`VALID LASTNAME`
			
			`1 row created.`
				√

	- Fires where needed	A message is displayed saying you deleted a staff along with the name of the staff	 `SQL> DELETE FROM staff WHERE staff_id = 99998;` `1 row deleted.`	√
	- Does not fire where not needed Similarly, same test is conducted for other three triggers and no errors were found during the test.	No extra message is displayed	 `SQL> DELETE FROM locations WHERE location_id = 99999;` `0 rows deleted.`	√

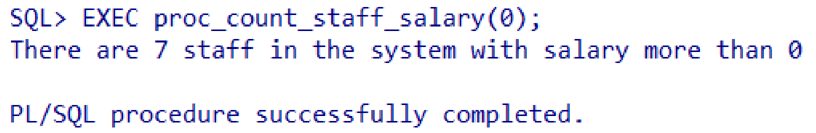
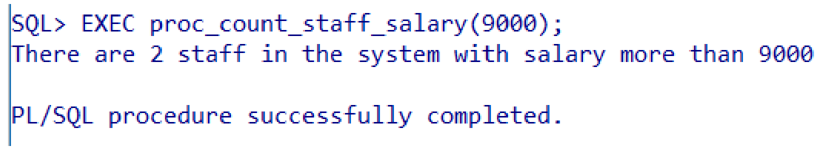
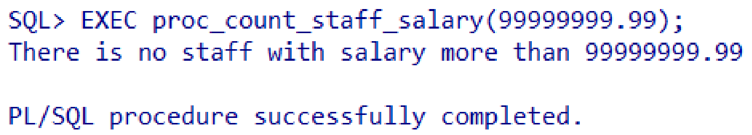
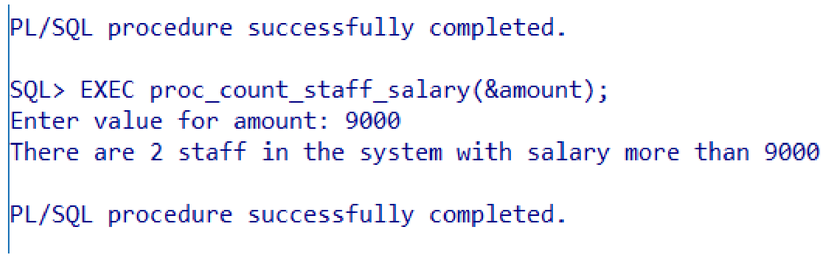
19	TEST PROC_RESET_SEQ
	- Increasing sequence value by 1.
	
	- Resettig the sequence
		New Sequence value displayed.	 `SQL> SELECT seq_festival_staff_id.NEXTVAL FROM DUAL;`
			
			`   NEXTVAL`
			`----------`
			`        12`
				√

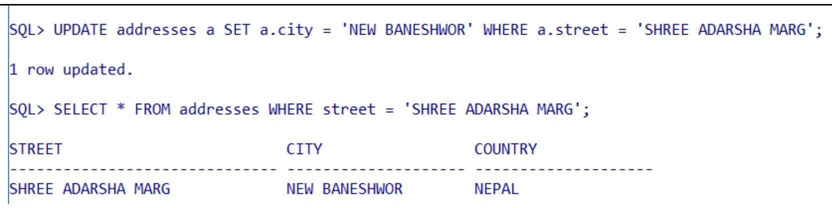
		Sequence reset back to its default value.	
			`SQL> EXEC proc_reset_seq('seq_festival_staff_id');`
			
			`PL/SQL procedure successfully completed.`
			
			`SQL>`
			`SQL> --Verify Sequence Currval`
			`SQL> SELECT seq_festival_staff_id.CURRVAL FROM DUAL;`
			
			`   CURRVAL`
			`----------`
			`         0`
				√

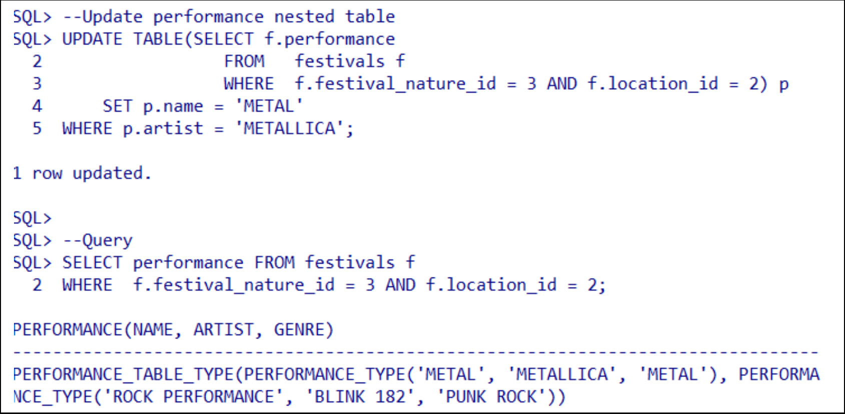
20	Test procedures that display count
	- Total number of staffs
	EXEC proc_count_staff;
	
	- Querying using actual query.
	
	In same way test is performed for procedures-
	proc_count_festivals
	where no errors were found.
		Total number of staffs gets displayed.	
			`SQL> SET SERVEROUTPUT ON`
			`SQL> EXEC proc_count_staff;`
			`There is no staff available.`
			
			`PL/SQL procedure successfully completed.`
			
			`SQL> SELECT COUNT(staff_id) FROM staff;`
			
			`COUNT(STAFF_ID)`
			`---------------`
			`              0`
				√

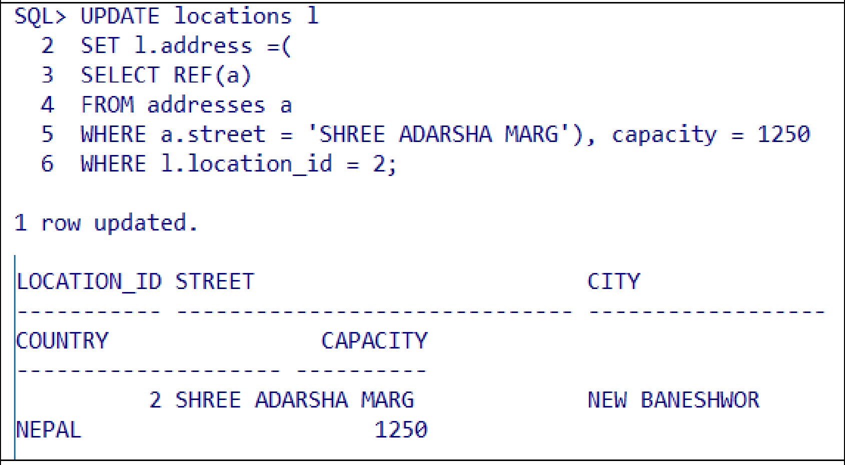
21	Test to display highest price of location
	- Using procedure created.
	EXEC proc_exp_location;
	
	- Verifying using actual query.
	
	Similarly, test has been carried out for displaying minimum cost
		Highest price of location gets displayed.
		
		Same output.
			
			`SQL> EXEC proc_exp_location;`
			`The most expensive location costs 20500`
			
			`PL/SQL procedure successfully completed.`
			
			`SQL> SELECT MAX(price) FROM locations;`
			
			`MAX(PRICE)`
			`----------`
			`     20500`
				√

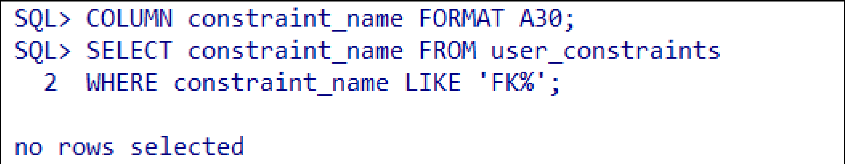
	of location, and no errors were encountered.			

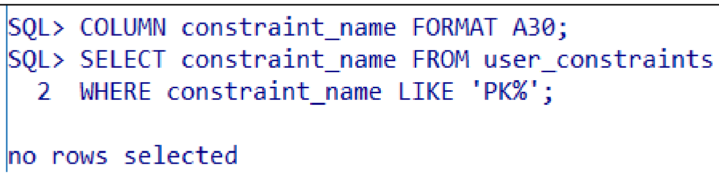
22	test proc_count_staff_salary Bound checking.
	- Minimum range
	EXEC proc_count_staff_salary(0)
	
	- Medium range
	EXEC proc_count_staff_salary(9000)
	
	- Maximum range
	EXEC proc_count_staff_salary(99999999.99)
	
	- Entering amount for parameter according to our will.
	EXEC proc_count_staff_salary(&amount);
		Displays name of all staffs whose salary is more than that of value passed through parameter.	
			`SQL> EXEC proc_count_staff_salary(0);`
			`There are 7 staff in the system with salary more than 0`
			
			`PL/SQL procedure successfully completed.`
			
			
			`SQL> EXEC proc_count_staff_salary(9000);`
			`There are 2 staff in the system with salary more than 9000`
			
			`PL/SQL procedure successfully completed.`
			
			
			`SQL> EXEC proc_count_staff_salary(99999999.99);`
			`There is no staff with salary more than 99999999.99`
			
			`PL/SQL procedure successfully completed.`
			
			
			`PL/SQL procedure successfully completed.`
			
			`SQL> EXEC proc_count_staff_salary(&amount);`
			`Enter value for amount: 9000`
			`There are 2 staff in the system with salary more than 9000`
			
			`PL/SQL procedure successfully completed.`
				√
				
				√
				
				√
				
				√

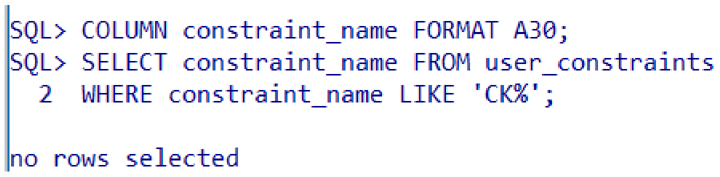
23	Updating address table addresses
	- Updated city = 'NEW BANESHWOR' WHERE street = SHREE ADARSHA MARG'
	
	Querying updated result.
		City successfully updated to new city.	
			`SQL> UPDATE addresses a SET a.city = 'NEW BANESHWOR' WHERE a.street = 'SHREE ADARSHA MARG';`
			
			`1 row updated.`
			
			`SQL> SELECT * FROM addresses WHERE street = 'SHREE ADARSHA MARG';`
			
			`STREET                         CITY                 COUNTRY`
			`------------------------------ -------------------- --------------------`
			`SHREE ADARSHA MARG             NEW BANESHWOR        NEPAL`
				√

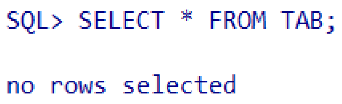
24	Update performance nested table
	Update query to update performance name for performance table where primary keys were- f.festival_nature_id = 3 AND f.location_id = 2
		Performance name updated successfully.	
			`SQL> --Update performance nested table`
			`SQL> UPDATE TABLE(SELECT f.performance`
			`                FROM   festivals f`
			`                WHERE  f.festival_nature_id = 3 AND f.location_id = 2) p`
			`        SET p.name = 'METAL'`
			`  WHERE p.artist = 'METALLICA';`
			
			`1 row updated.`
			
			`SQL>`
			`SQL> --Query`
			`SQL> SELECT performance FROM festivals f`
			`  WHERE  f.festival_nature_id = 3 AND f.location_id = 2;`
			
			`PERFORMANCE(NAME, ARTIST, GENRE)`
			`--------------------------------------------------------------------------------`
			`PERFORMANCE_TABLE_TYPE(PERFORMANCE_TYPE('METAL', 'METALLICA', 'METAL'), PERFORMA`
			`NCE_TYPE('ROCK PERFORMANCE', 'BLINK 182', 'PUNK ROCK'))`
				√

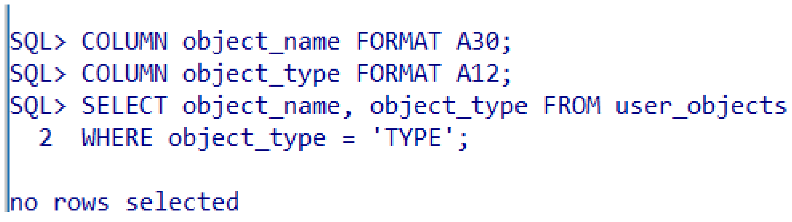
25	Update address ref and capacity together
	Selecting address from addresses table and capacity updated to 1250 where
	Location_id = 2
		New capacity and location set successfully.	
			`SQL> UPDATE locations l`
			`  SET l.address =(`
			`  SELECT REF(a)`
			`  FROM addresses a`
			`  WHERE a.street = 'SHREE ADARSHA MARG'), capacity = 1250`
			`  WHERE l.location_id = 2;`
			
			`1 row updated.`
			
			`LOCATION_ID STREET                         CITY`
			`----------- ------------------------------ ------------------`
			`COUNTRY              CAPACITY`
			`-------------------- ----------`
			`          2 SHREE ADARSHA MARG             NEW BANESHWOR`
			`NEPAL                      1250`
				√

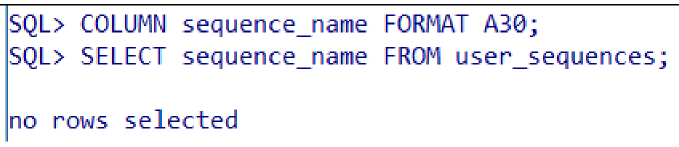
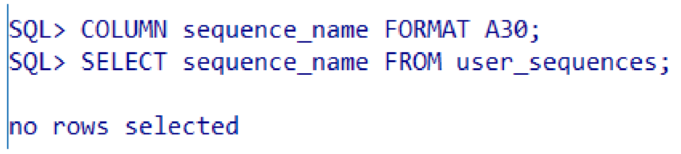
26	DROP TEST
	- Dropping all foreign key.
	Querying all foreign keys.
	COLUMN constraint_name FORMAT A30;
	SELECT constraint_name FROM user_constraints WHERE constraint_name LIKE 'FK%';
		All forgien key gets dropped.
		No rows selected.
			
			`SQL> COLUMN constraint_name FORMAT A30;`
			`SQL> SELECT constraint_name FROM user_constraints`
			`  WHERE constraint_name LIKE 'FK%';`
			
			`no rows selected`
				√

	- Dropping all primay key.
	Querying all primary keys.
	COLUMN constraint_name FORMAT A30;
	SELECT constraint_name FROM user_constraints WHERE constraint_name LIKE 'PK%';
		All primary key gets dropped.
		No rows selected.
			
			`SQL> COLUMN constraint_name FORMAT A30;`
			`SQL> SELECT constraint_name FROM user_constraints`
			`  2  WHERE constraint_name LIKE 'PK%';`
			
			`no rows selected`
				√

	- Dropping all check constraints
	Querying all check constraintss.
	COLUMN constraint_name FORMAT A30;
	SELECT constraint_name FROM user_constraints WHERE constraint_name LIKE 'CK%';
		All check constraints get dropped.
		No rows selected.
			
			`SQL> COLUMN constraint_name FORMAT A30;`
			`SQL> SELECT constraint_name FROM user_constraints`
			`  2  WHERE constraint_name LIKE 'CK%';`
			
			`no rows selected`
				√

	- Dropping all table.
	Querying all table that exists.
	COLUMN tname FORMAT A30;
	SELECT * FROM TAB;
		No rows selected.	
			`SQL> SELECT * FROM TAB;`
			
			`no rows selected`
				√

	- Dropping all types.
	COLUMN object_name FORMAT A30;
	COLUMN object_type FORMAT A12;
		No rows selected.	
			`SQL> COLUMN object_name FORMAT A30;`
			`SQL> COLUMN object_type FORMAT A12;`
			`SQL> SELECT object_name, object_type FROM user_objects`
			`  2  WHERE object_type = 'TYPE';`
			
			`no rows selected`
				√

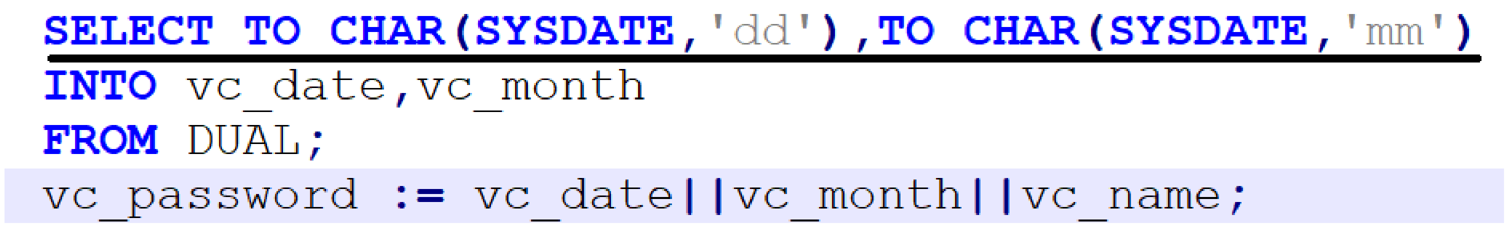
	SELECT object_name, object_type FROM user_objects WHERE object_type = 'TYPE';
	
	- Dropping all seuqences.
	COLUMN sequence_name FORMAT A30;
	SELECT sequence_name FROM user_sequences;
		No rows selected.	
			```
			SQL> COLUMN sequence_name FORMAT A30;
			SQL> SELECT sequence_name FROM user_sequences;

			no rows selected
			```
			
			```
			SQL> COLUMN sequence_name FORMAT A30;
			SQL> SELECT sequence_name FROM user_sequences;

			no rows selected
			```

Additional Research Evidence

Additional research has been carried out to support assignment. These additional researches were for creating schema level trigger, concatenating date for creating password, exception handling for procedures and functions, creating cursors using while loop, out parameter in procedures etc. all these additional researches has been explained in detail below.

- **Date Concatenation**

Like string date can also be concatenated in database. Unlike CONCAT function for sting, date concatenation cannot be carried out in similar manner. To_char function is used to concatenate date in oracle. By concatenating date, password for staff can be generated. Password of staff contains combination of letter from their name and date in its concatenated format. Given below provided screenshot of date concatenation in system.



```
SELECT TO_CHAR(SYSDATE,'dd'),TO_CHAR(SYSDATE,'mm')
INTO vc_date,vc_month
FROM DUAL;
vc_password := vc_date||vc_month||vc_name;
```

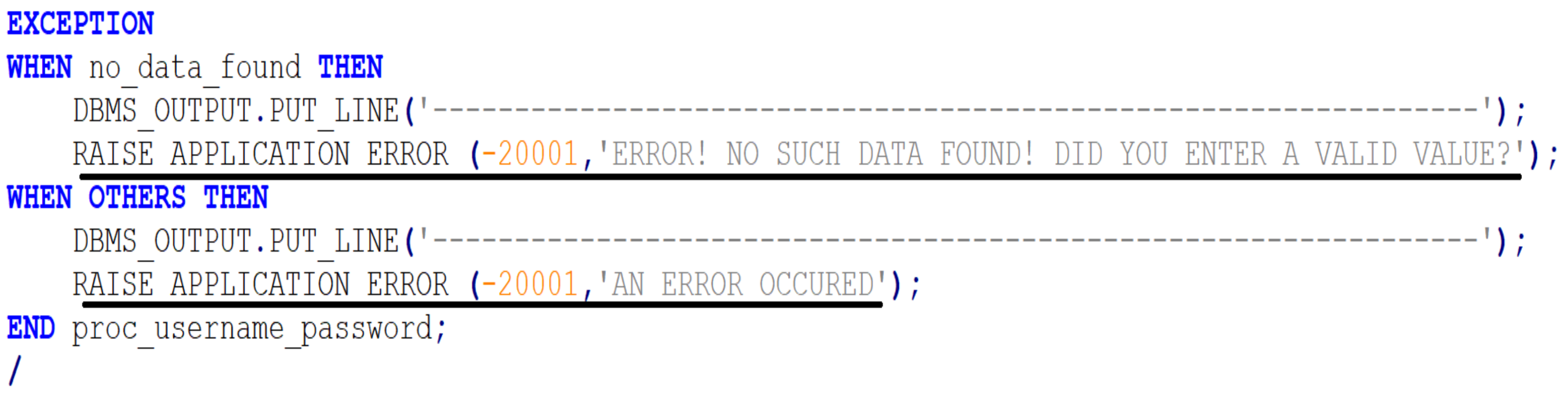
Fig I – database syntax to concatenation of date.

- **Throwing Exception**

Appropriate error handling has been carried out throughout assignment. Error handling has been carried out suing exception, this can be found mainly in procedures and functions. Example of error handling used in function is discussed below.

If procedure to display both username and password for staff, if no data passed through parameter of procedure is found, error would be generated. This error is then handled using error handling. When no data is found, message stating, "ERROR! NO SUCH DATA FOUND! DID YOU ENTER A VALID VALUE?" is displayed. If error is occurred due to other various reason another message stating, "AN ERROR OCCURRED" is displayed.

Screenshot of exception handling for procedure that prints both username and password for staff is shown in screenshot below.

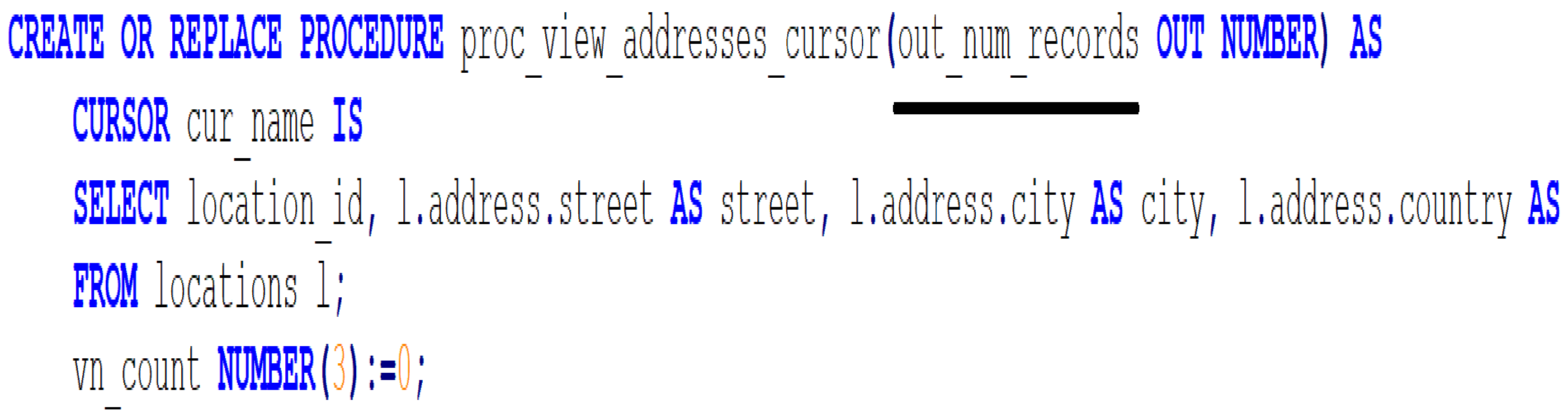


```
EXCEPTION
WHEN no_data_found THEN
    DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------');
    RAISE_APPLICATION_ERROR (-20001,'ERROR! NO SUCH DATA FOUND! DID YOU ENTER A VALID VALUE?');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------');
    RAISE_APPLICATION_ERROR (-20001,'AN ERROR OCCURED');
END proc_username_password;
/
```

Fig ii – database syntax for error handling using exception.

- **Use of "OUT" parameter**

Out parameter has also been used in this assignment. Out parameter is assigned while defining cursors. For example, out parameter has been using in procedure named "proc_view_addresses_cursor" and its parameter is "out_num_records". This parameter in this procedure return the total number of found rows(locations) found. So out parameter can be useful when cursors are defined in procedures.

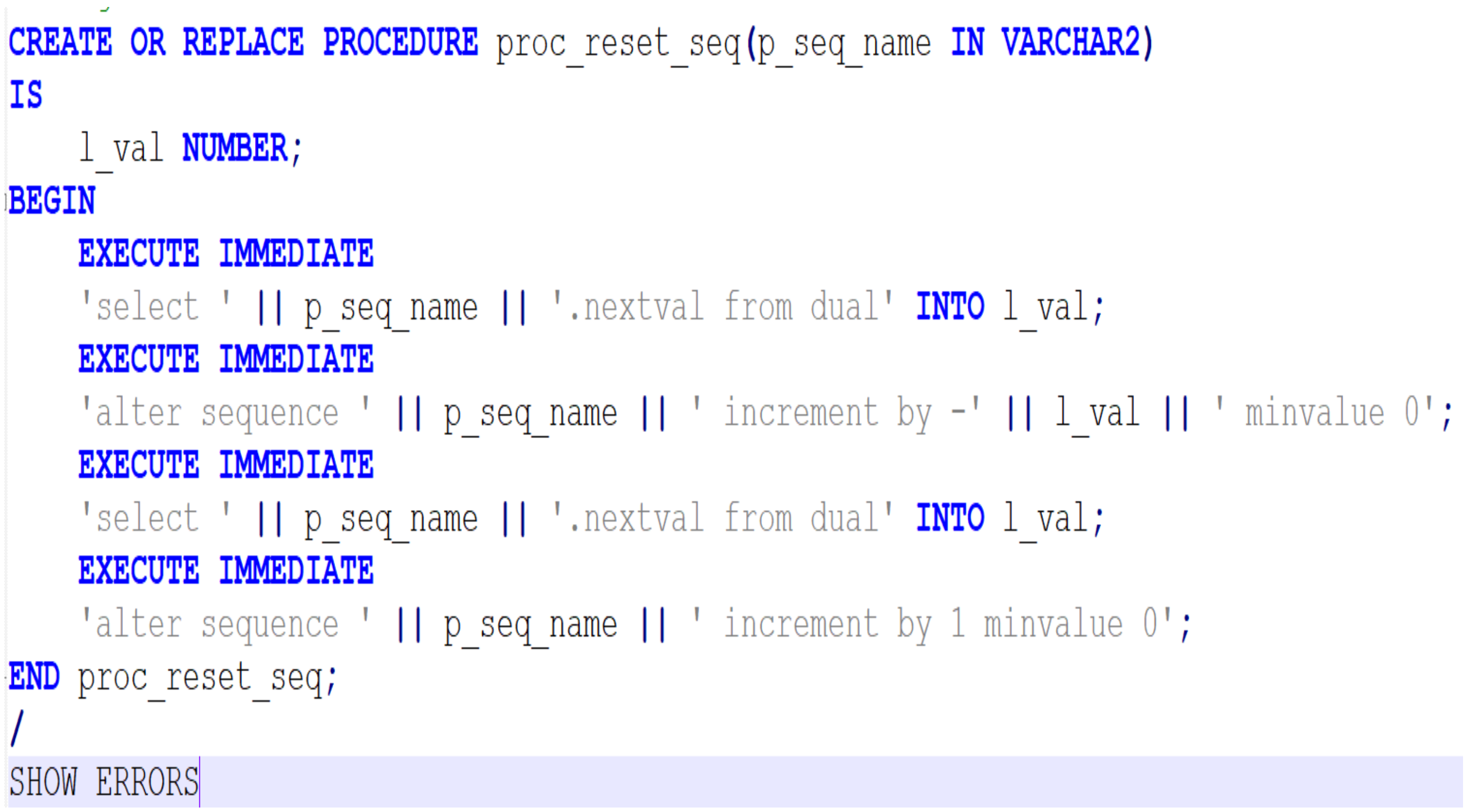


```
CREATE OR REPLACE PROCEDURE proc_view_addresses_cursor(out_num_records OUT NUMBER) AS
    CURSOR cur_name IS
    SELECT location_id, l.address.street AS street, l.address.city AS city, l.address.country AS
    FROM locations l;
    vn_count NUMBER(3):=0;
```

Fig iii – using out parameter in procedure named proc_view_addresses_cursor

- **Resetting Sequence back to its default value.**

A procedure has been created to reset all sequence back to its default value. Procedure is named as "proc_reset_seq". when name of sequence is passed through parameter of this procedure, it gets reset back to its default value. Currently four sequence has been created in this assignment which are used for different table as their primary key. Whenever these sequences had to be reset, they can simply be passed through procedure mentioned above.

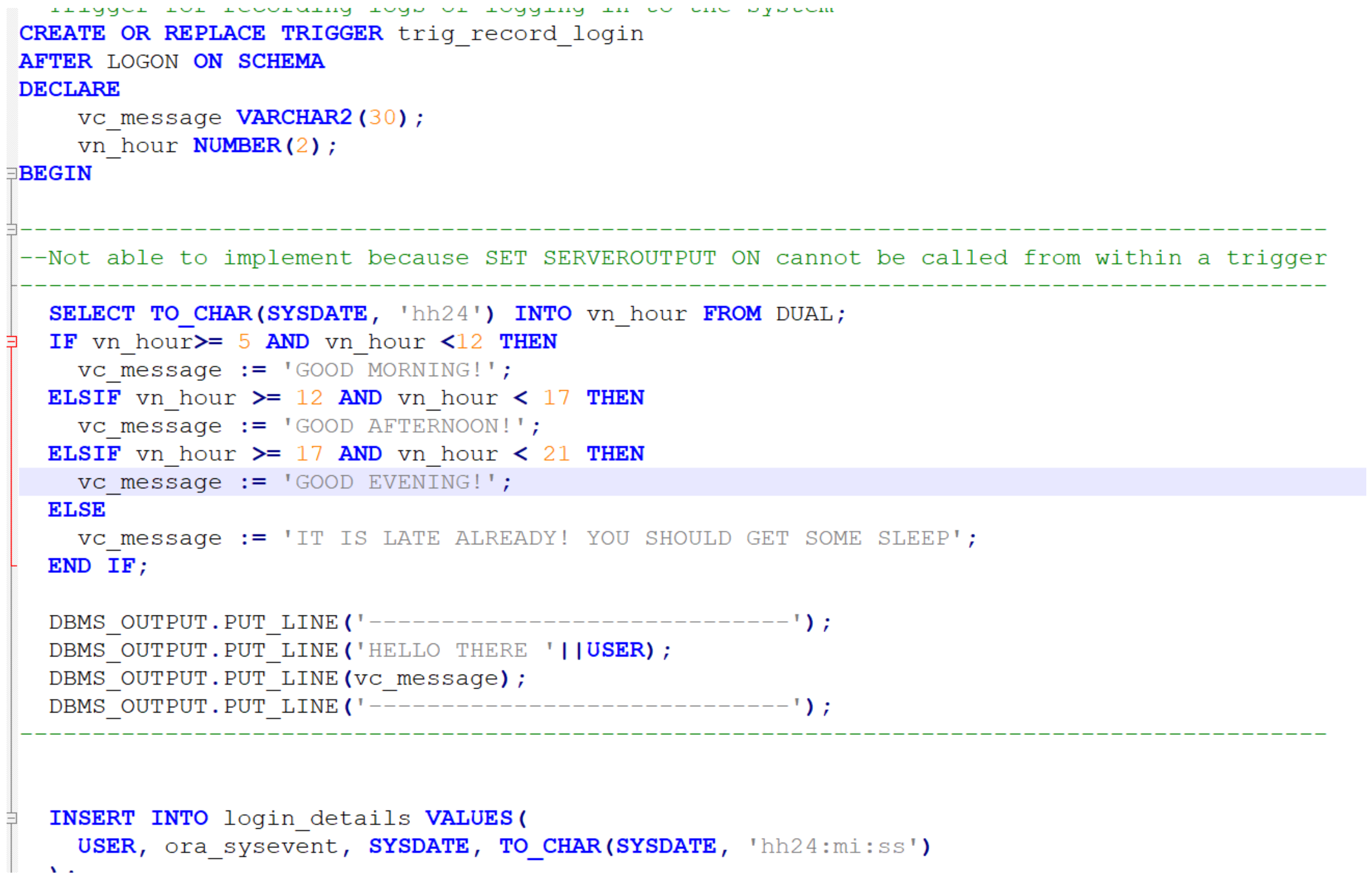


```
CREATE OR REPLACE PROCEDURE proc_reset_seq(p_seq_name IN VARCHAR2)
IS
    l_val NUMBER;
BEGIN
    EXECUTE IMMEDIATE
    'select ' || p_seq_name || '.nextval from dual' INTO l_val;
    EXECUTE IMMEDIATE
    'alter sequence ' || p_seq_name || ' increment by -' || l_val || ' minvalue 0';
    EXECUTE IMMEDIATE
    'select ' || p_seq_name || '.nextval from dual' INTO l_val;
    EXECUTE IMMEDIATE
    'alter sequence ' || p_seq_name || ' increment by 1 minvalue 0';
END proc_reset_seq;
/
SHOW ERRORS
```

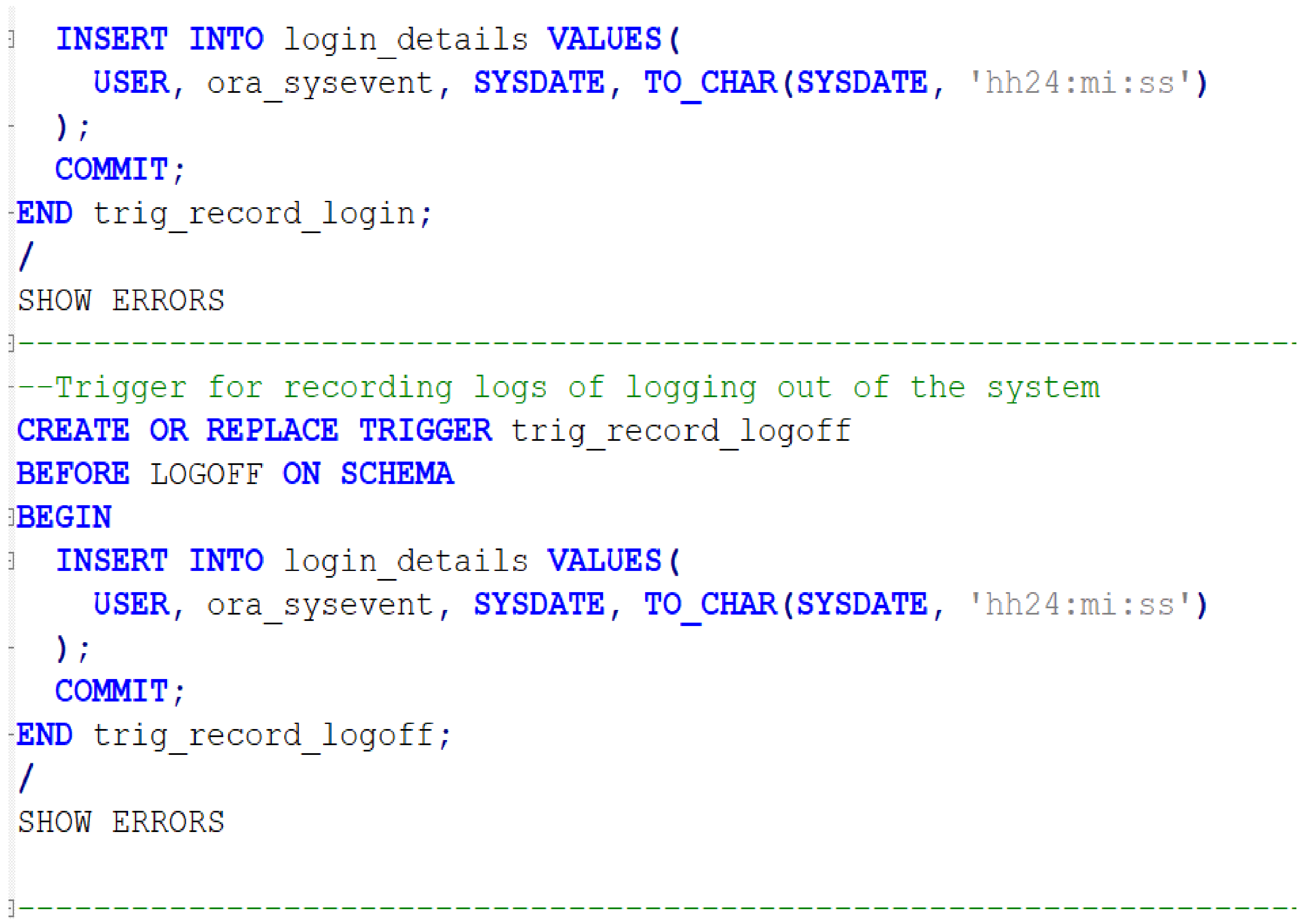
Fig iv – procedure to reset all sequence back to its default value.

- **Schema Level Trigger**

Along with triggers for datatype, schema level trigger has been created in the system. Schema level trigger is created for keeping activity log of all users in database. Not only does log contains, login in information, it also contains code that displays message stating time of the day i.e. “Good Morning”, “Good Afternoon” or “Good Evening”. If user tries to log in late in the day, message stating “IT IS LATE ALREADY! YOU SHOULD GET SOME SLEEP” will be displayed. Given below is screenshot of syntax for creation of schema level trigger. However, these messages are not displayed as SET SERVEROUTPUT ON cannot be executed from within a trigger. After this has been manually entered to oracle files, the trigger will be able to output those messages.



```
  Trigger for recording logs of logging in to the system
CREATE OR REPLACE TRIGGER trig_record_login
AFTER LOGON ON SCHEMA
DECLARE
    vc_message VARCHAR2(30);
    vn_hour NUMBER(2);
BEGIN

-------------------------------------------------------------------------------------------
--Not able to implement because SET SERVEROUTPUT ON cannot be called from within a trigger
-------------------------------------------------------------------------------------------
  SELECT TO_CHAR(SYSDATE, 'hh24') INTO vn_hour FROM DUAL;
  IF vn_hour>= 5 AND vn_hour <12 THEN
    vc_message := 'GOOD MORNING!';
  ELSIF vn_hour >= 12 AND vn_hour < 17 THEN
    vc_message := 'GOOD AFTERNOON!';
  ELSIF vn_hour >= 17 AND vn_hour < 21 THEN
    vc_message := 'GOOD EVENING!';
  ELSE
    vc_message := 'IT IS LATE ALREADY! YOU SHOULD GET SOME SLEEP';
  END IF;

  DBMS_OUTPUT.PUT_LINE('-----------------------------');
  DBMS_OUTPUT.PUT_LINE('HELLO THERE '||USER);
  DBMS_OUTPUT.PUT_LINE(vc_message);
  DBMS_OUTPUT.PUT_LINE('-----------------------------');
-------------------------------------------------------------------------------------------


  INSERT INTO login_details VALUES(
    USER, ora_sysevent, SYSDATE, TO_CHAR(SYSDATE, 'hh24:mi:ss')
  );
```

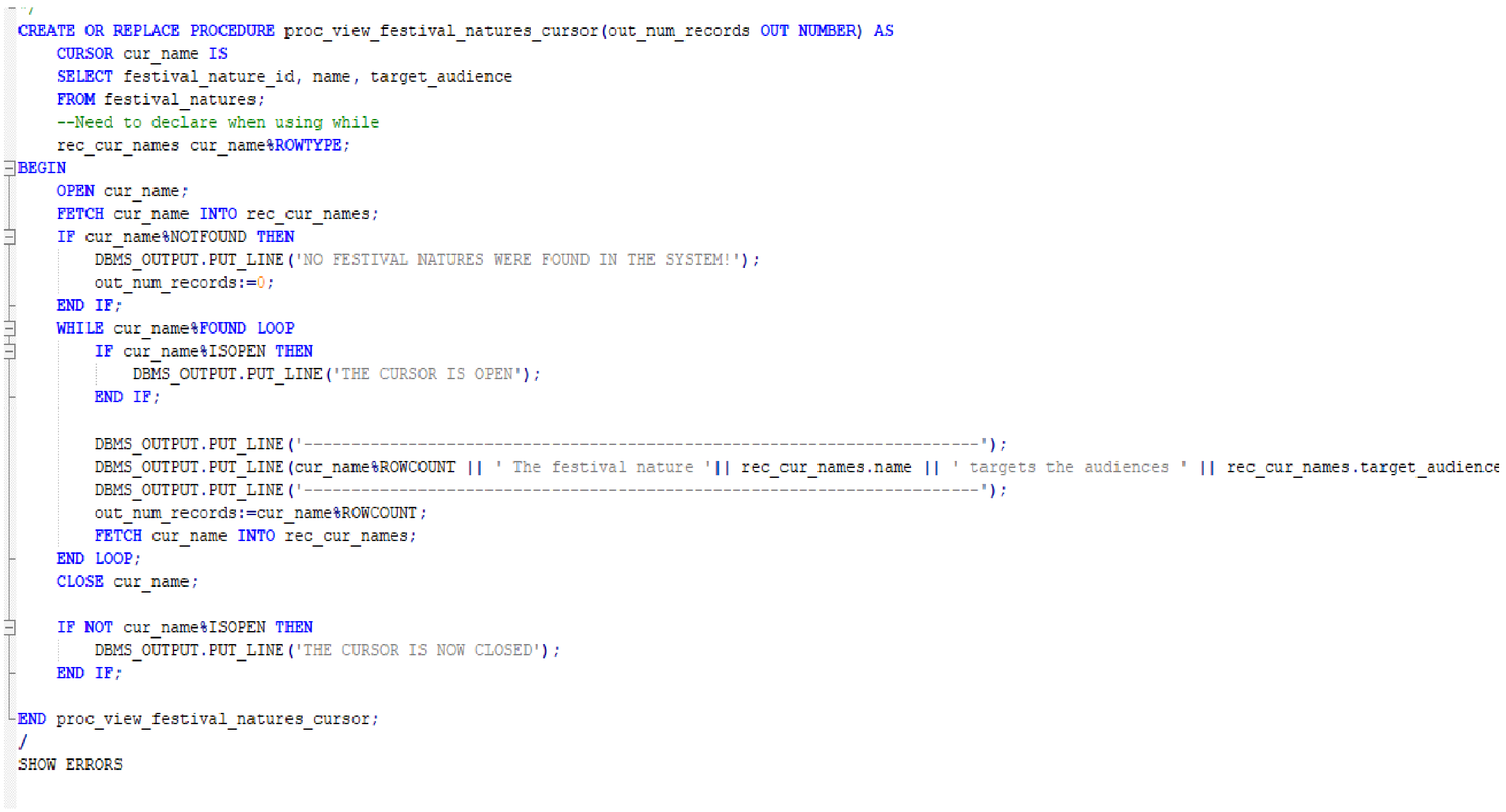


```
  INSERT INTO login_details VALUES(
    USER, ora_sysevent, SYSDATE, TO_CHAR(SYSDATE, 'hh24:mi:ss')
  );
  COMMIT;
END trig_record_login;
/
SHOW ERRORS
-------------------------------------------------------------------
--Trigger for recording logs of logging out of the system
CREATE OR REPLACE TRIGGER trig_record_logoff
BEFORE LOGOFF ON SCHEMA
BEGIN
  INSERT INTO login_details VALUES(
    USER, ora_sysevent, SYSDATE, TO_CHAR(SYSDATE, 'hh24:mi:ss')
  );
  COMMIT;
END trig_record_logoff;
/
SHOW ERRORS

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

Fig v – database syntax for creating log for user login.

- **Defining Cursor using WHILE loop**

Cursor has been created using WHILE loop as well. Unlike FOR loop, under WHILE loop cursor has to be closed. Given below is screenshot of syntax of database code where cursor has been defined using WHILE loop.



```
CREATE OR REPLACE PROCEDURE proc_view_festival_natures_cursor(out_num_records OUT NUMBER) AS
    CURSOR cur_name IS
    SELECT festival_nature_id, name, target_audience
    FROM festival_natures;
    --Need to declare when using while
    rec_cur_names cur_name%ROWTYPE;
BEGIN
    OPEN cur_name;
    FETCH cur_name INTO rec_cur_names;
    IF cur_name%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO FESTIVAL NATURES WERE FOUND IN THE SYSTEM!');
        out_num_records:=0;
    END IF;
    WHILE cur_name%FOUND LOOP
        IF cur_name%ISOPEN THEN
            DBMS_OUTPUT.PUT_LINE('THE CURSOR IS OPEN');
        END IF;

        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE(cur_name%ROWCOUNT || ' The festival nature '|| rec_cur_names.name || ' targets the audiences ' || rec_cur_names.target_audience
        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------------------');
        out_num_records:=cur_name%ROWCOUNT;
        FETCH cur_name INTO rec_cur_names;
    END LOOP;
    CLOSE cur_name;

    IF NOT cur_name%ISOPEN THEN
        DBMS_OUTPUT.PUT_LINE('THE CURSOR IS NOW CLOSED');
    END IF;

END proc_view_festival_natures_cursor;
/
SHOW ERRORS
```

Fig vi – defining cursor using WHILE loop in database.

Reference

Given below are list of websites which were taken as reference for completion of this assignment.

- RebellionRider. (2019). *Schema Level Database LOGON Trigger In PL/SQL | RebellionRider*. [online] Available at: http://www.rebellionrider.com/schema-level-database-logon-trigger-in-pl-sql/ [Accessed 6 Apr. 2019].
- Matthias Hoys. (2012). *Extracting hour and minute from a TIMESTAMP and DATE variable*. [online] Available at: https://matthiashoys.wordpress.com/2012/07/02/extracting-hour-and-minute-from-date-variable/ [Accessed 29 Mar. 2019].
- Techonthenet.com. (2019). *Oracle / PLSQL: || Operator*. [online] Available at: https://www.techonthenet.com/oracle/functions/concat2.php [Accessed 13 Mar. 2019].
- Plsql-tutorial.com. (2019). *PL/SQL Tutorial - PL/SQL - passing parameters in procedures and functions.*. [online] Available at: https://plsql-tutorial.com/plsql-passing-parameters-procedure-function.htm [Accessed 17 Mar. 2019].
- Techonthenet.com. (2019). *Oracle / PLSQL: WHEN OTHERS Clause*. [online] Available at: https://www.techonthenet.com/oracle/exceptions/when_others.php [Accessed 1 Apr. 2019].
- H., Porter, D. and Heller, J. (2019). *How do I reset a sequence in Oracle?* [online] Stack Overflow. Available at: https://stackoverflow.com/questions/51470/how-do-i-reset-a-sequence-in-oracle [Accessed 7 Apr. 2019].
- Mistapink (2015). *Oracle 12c create user* [online] Stack Exchange. Available at: https://dba.stackexchange.com/questions/60806/oracle-12c-create-user [Accessed 13 Mar. 2019]

- Tutorialspoint (2019). *PL/SQL – CASE Statement* [online] Tutorialspoint. Available at: https://www.tutorialspoint.com/plsql/plsql_case_statement.htm [Accessed 10 Apr. 2019]

- Oracle Docs (2019). PL/SQL Control Structrues [online] Oracle Docs. Available at: https://docs.oracle.com/cd/A97630_01/appdev.920/a96624/04_struc.htm [Accessed 10 Apr. 2019]

- Domscheit, W. (2014). How do I update a single value inside a nested table type in Oracle11g? [online] Stackoverflow. Available at: https://stackoverflow.com/questions/25730999/how-do-i-update-a-single-value-inside-a-nested-table-type-in-oracle11g [Accessed 11 Apr. 2019]

Appendix

All script files are listed below-

1. Createuser.txt

```
/*

	-Create User CSY2038B4 for Group 4

	Mistapink (2015)
	-alter session set "_ORACLE_SCRIPT"=true; --FOR 12C

	CREATE USER FILE
		-Contains commands for creating the user

	GROUP 4

		18406547 - Diwas Lamsal
		18406498 - Bhuwan Khatiwada
		18413710 - Sandip Kumar Subba
		18413644 - Bishownath Dhakal
```

```
*/



--@D:\Database_Assignment_II\script\createuser.txt

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

CREATE USER csy2038B4 IDENTIFIED BY group4;
GRANT CREATE SESSION TO CSY2038B4;
GRANT CREATE TABLE TO CSY2038B4;
GRANT CREATE VIEW TO CSY2038B4;
GRANT CREATE SEQUENCE TO CSY2038B4;
GRANT CREATE SYNONYM TO CSY2038B4;
GRANT CREATE PROCEDURE TO CSY2038B4;
GRANT CREATE TRIGGER TO CSY2038B4;
GRANT CREATE CLUSTER TO CSY2038B4;
GRANT CREATE TYPE TO CSY2038B4;
GRANT UNLIMITED TABLESPACE TO CSY2038B4;

ALTER USER CSY2038B4 QUOTA UNLIMITED ON SYSTEM;
```

2. Create_4.txt

```
/*

        CREATE FILE

                -Contains create commands for creating types, sequences and tables

                -Creates are in order


        GROUP 4


                18406547 - Diwas Lamsal

                18406498 - Bhuwan Khatiwada

                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal


*/



--@D:\Database_Assignment_II\script\create_4.txt
```

```
--Display dbms output
SET SERVEROUTPUT ON



------------------------------------------TYPES-----------------------------------------------

--This command can be used to verify what has been created in the schema
SELECT object_name FROM user_objects;



--address_type
CREATE OR REPLACE TYPE address_type AS OBJECT(
	street VARCHAR2(30),
	city VARCHAR2(30),
	country VARCHAR2(30)
);
/

--addresses table made of address_type
CREATE TABLE addresses OF address_type;
```

```
--contact_detail_type

CREATE OR REPLACE TYPE contact_detail_type AS OBJECT(

        contact_number VARCHAR2(15),

        number_type VARCHAR2(30)

);

/


--contact_detail_varray_type made of contact_detail_type

CREATE TYPE contact_detail_varray_type AS VARRAY(2) OF contact_detail_type;

/


--performance_type

CREATE OR REPLACE TYPE performance_type AS OBJECT(

        name VARCHAR2(30),

        artist VARCHAR2(30),

        genre VARCHAR2(30)

);

/


--performance_table_type made of performance_type
```

```
CREATE TYPE performance_table_type AS TABLE OF performance_type;

/



---------------------------------------------TABLES--------------------------------------------------



--Staff table and the sequence

CREATE TABLE staff(

            staff_id NUMBER(6),

            firstname VARCHAR2(20),

            lastname VARCHAR2(20),

            gender CHAR DEFAULT 'M',

            contact contact_detail_varray_type,

            current_address address_type,

            permanent_address address_type,

            email VARCHAR2(60),

            leader NUMBER(6) NOT NULL,

            salary NUMBER(10, 2)

);
```

```
CREATE SEQUENCE seq_staff_id

INCREMENT BY 1

START WITH 1

MINVALUE 0

MAXVALUE 999999;



--Festival_natures table and the sequence

CREATE TABLE festival_natures(

	festival_nature_id NUMBER(6),

	name VARCHAR2(30),

	target_audience VARCHAR2(30)

);


CREATE SEQUENCE seq_festival_nature_id

INCREMENT BY 1

START WITH 1

MINVALUE 0

MAXVALUE 999999;
```

```
--Locations table and the sequence

CREATE TABLE locations(

            location_id NUMBER(6),

            address REF address_type SCOPE IS addresses,

            capacity NUMBER(5),

            price NUMBER(10,2)

);


CREATE SEQUENCE seq_location_id

INCREMENT BY 1

START WITH 1

MINVALUE 0

MAXVALUE 999999;



--Festivals table

CREATE TABLE festivals(

            festival_nature_id NUMBER(6) NOT NULL,

            location_id NUMBER(6) NOT NULL,

            festival_name VARCHAR2(30),

            performance performance_table_type
)
NESTED TABLE performance STORE AS performance_table;



--Festival_staff table and the sequence
CREATE TABLE festival_staff(
        festival_staff_id NUMBER(6),
        staff_id NUMBER(6) NOT NULL,
        festival_nature_id NUMBER(6) NOT NULL,
        location_id NUMBER(6) NOT NULL
);

CREATE SEQUENCE seq_festival_staff_id
INCREMENT BY 1
START WITH 1
MINVALUE 0
MAXVALUE 999999;


--------------------------------------------VIEW-------------------------------------------------
```

```
--Confirm that all the creates have worked properly


--View objects
COLUMN object_name FORMAT A30;
COLUMN object_type FORMAT A12;
SELECT object_name, object_type FROM user_objects
WHERE object_type = 'TYPE';


--View tables
COLUMN tname FORMAT A30;
SELECT * FROM TAB;


--View Sequences
COLUMN sequence_name FORMAT A30;
SELECT sequence_name FROM user_sequences;
```

3. Create_test_4.txt

```
/*
```

```
	CREATE TEST FILE

		-Contains commands for testing the creates


	GROUP 4


		18406547 - Diwas Lamsal

		18406498 - Bhuwan Khatiwada

		18413710 - Sandip Kumar Subba

		18413644 - Bishownath Dhakal

*/


--@D:\Database_Assignment_II\script\create_test_4.txt



-------------------------------------------- CREATING TYPES -------------------------------------------------
```

```
-- TEST

--View Object TYPES

COLUMN object_name FORMAT A30;

COLUMN object_type FORMAT A12;

SELECT object_name, object_type FROM user_objects

WHERE object_type = 'TYPE';




--------------------------------- TEST RESULT--------------------------------


--------- BEFORE CREATING ALL TYPES ------------


--EXPECTED OUTPUT

-- NO ROWS SELECTED


-- ACTUAL OUTPUT

-- NO ROWS SELECTED


--------- AFTER CREATING ALL TYPES ------------
```

```
--EXPECTED OUTPUT

-- 5 ROWS SELECTED


-- ACTUAL OUTPUT

-- 5 ROWS SELECTED
```

```
-------------------------------------------- CREATING TABLES ------------------------------------------------


-- TEST

--View ALL TABLES

COLUMN tname FORMAT A30;

SELECT * FROM TAB;


--------------------------------- TEST RESULT--------------------------------
```

```
--------- BEFORE CREATING ALL TABLES ------------

--EXPECTED OUTPUT
-- NO ROWS SELECTED

-- ACTUAL OUTPUT
-- NO ROWS SELECTED

--------- AFTER CREATING ALL TABLES ------------

--EXPECTED OUTPUT
-- 7 ROWS SELECTED

-- ACTUAL OUTPUT
-- 7 ROWS SELECTED
```

```
-------------------------------------------- CREATING SEQUENCE -------------------------------------------------


-- TEST

--View SEQUENCE

COLUMN sequence_name FORMAT A30;

SELECT sequence_name FROM user_sequences;





---------------------------------- TEST RESULT--------------------------------


--------- BEFORE CREATING ALL SEQUENCE ------------


--EXPECTED OUTPUT

-- NO ROWS SELECTED


-- ACTUAL OUTPUT

-- NO ROWS SELECTED


--------- AFTER CREATING ALL SEQUENCE ------------
```

```
--EXPECTED OUTPUT
-- 4 ROWS SELECTED

-- ACTUAL OUTPUT
-- 4 ROWS SELECTED
```

4. Constraint_4.txt

```
/*

	CONSTRAINT FILE
		-Contains commands for creating constraints
		-Adds pk, fk, and other constraints to tables

	GROUP 4

		18406547 - Diwas Lamsal
		18406498 - Bhuwan Khatiwada
```

```
                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal

*/




--@D:\Database_Assignment_II\script\constraint_4.txt



--Display dbms output

SET SERVEROUTPUT ON



---------------------------------------------STAFF TABLE------------------------------------------------


--Staff table and constraints

ALTER TABLE staff

ADD CONSTRAINT pk_staff

PRIMARY KEY (staff_id);
```

```
ALTER TABLE staff

ADD CONSTRAINT ck_staff_gender

CHECK (gender IN('M', 'F'));


ALTER TABLE staff

ADD CONSTRAINT ck_staff_firstname

CHECK (firstname= UPPER(firstname));


ALTER TABLE staff

ADD CONSTRAINT ck_staff_lastname

CHECK (lastname= UPPER(lastname));


ALTER TABLE staff

ADD CONSTRAINT fk_s_staff

FOREIGN KEY (leader)

REFERENCES staff(staff_id);



-------------------------------------------FESTIVAL_NATURES TABLE-------------------------------------------------
```

```
--festival_natures table and the constraints

ALTER TABLE festival_natures

ADD CONSTRAINT pk_festival_natures

PRIMARY KEY (festival_nature_id);


ALTER TABLE festival_natures

ADD CONSTRAINT ck_festival_natures_name

CHECK (name= UPPER(name));


ALTER TABLE festival_natures

ADD CONSTRAINT ck_festival_natures_target_audience

CHECK (target_audience= UPPER(target_audience));




-------------------------------------------LOCATIONS------------------------------------------------



--locations table and the constraints

ALTER TABLE locations

ADD CONSTRAINT pk_locations
```

```
PRIMARY KEY (location_id);




-------------------------------------------FESTIVALS TABLE-----------------------------------------------


--festivals table and the constraints
ALTER TABLE festivals
ADD CONSTRAINT pk_festivals
PRIMARY KEY (festival_nature_id, location_id);

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

ALTER TABLE festivals
ADD CONSTRAINT fk_f_locations
FOREIGN KEY (location_id)
```

```
REFERENCES locations(location_id);


ALTER TABLE festivals

ADD CONSTRAINT ck_festivals_festival_name

CHECK (festival_name = UPPER(festival_name));




-------------------------------------------FESTIVAL_STAFF TABLE------------------------------------------------


--festival_staff table and the constraints

ALTER TABLE festival_staff

ADD CONSTRAINT pk_festival_staff

PRIMARY KEY (festival_staff_id);


ALTER TABLE festival_staff

ADD CONSTRAINT fk_fs_festivals

FOREIGN KEY (festival_nature_id, location_id)

REFERENCES festivals(festival_nature_id, location_id);


ALTER TABLE festival_staff
```

```
ADD CONSTRAINT fk_fs_staff

FOREIGN KEY (staff_id)

REFERENCES staff(staff_id);



------------------------------------------VIEW-----------------------------------------------



COLUMN constraint_name FORMAT A30;

SELECT constraint_name FROM user_constraints

WHERE constraint_name NOT LIKE 'SYS%';
```

5. Constraint_test_4.txt

```
/*

          CONSTRAINT TEST FILE

                    -Contains commands for testing the constraints


          GROUP 4
```

```
                    18406547 - Diwas Lamsal

                    18406498 - Bhuwan Khatiwada

                    18413710 - Sandip Kumar Subba

                    18413644 - Bishownath Dhakal

*/

--@D:\Database_Assignment_II\script\constraint_test_4.txt

-------------------------------------------STAFF TABLE------------------------------------------------

-----------------------------TEST FOR PRIMARY KEY CONSTRAINT------------------------------

-- BEFORE ASSIGNING PRIMARY KEY TO STAFF TABLE

-- CHECKING ALREADY INSERTED ROWS IN staff

SELECT staff_id,firstname FROM staff;
```

```
-- TRYING TO INSERT IN staff Table

INSERT INTO staff (staff_id,firstname)
VALUES (1, 'BISHOWNATH');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.


-- AFTER ASSIGNING PRIMARY KEY TO STAFF TABLE


-- CREATING PRIMARY KEY FOR STAFF TABLE

ALTER TABLE staff
ADD CONSTRAINT pk_staff
PRIMARY KEY (staff_id);

-- TRYING TO INSERT IN staff Table
```

```
INSERT INTO staff (staff_id,firstname)

VALUES (1, 'BISHOWNATH');



-- EXPECTED OUTPUT / ACTUAL OUTPUT

-- unique constraint (CSY2038B4.PK_STAFF) violated


-- REMOVING THE DATA INSERTED


DELETE FROM staff;


-- OUTPUT

-- 1 ROW DELETED


SELECT staff_id FROM staff;

--OUTPUT

-- NO ROW SELECTED.
```

```
-----------------------------TEST FOR CHECK CONSTRAINT ON GENDER COLUMN ('M/F')------------------------------


-- BEFORE ASSIGNING CHECK CONSTRAINT TO STAFF TABLE

-- CHECKING ALREADY INSERTED ROWS IN staff

SELECT staff_id,firstname FROM staff;


-- TRYING TO INSERT IN staff Table

INSERT INTO staff (staff_id,firstname,gender)
VALUES (1, 'BISHOWNATH','O');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.

-- REMOVING THE DATA INSERTED
```

```
DELETE FROM staff;

-- AFTER ASSIGNING CHECK CONSTRAINT TO STAFF TABLE

ALTER TABLE staff
ADD CONSTRAINT ck_staff_gender
CHECK (gender IN('M', 'F'));

-- TRYING TO INSERT IN staff Table

INSERT INTO staff (staff_id,firstname,gender)
VALUES (1, 'BISHOWNATH','O');


-- EXPECTED OUTPUT / ACTUAL OUTPUT
--check constraint (CSY2038B4.CK_STAFF_GENDER) violated


-- TRYING TO INSERT CORRECT DATA IN staff Table
DELETE FROM staff;
```

```
INSERT INTO staff (staff_id,firstname,gender)

VALUES (1, 'BISHOWNATH','M');



-- EXPECTED OUTPUT / ACTUAL OUTPUT

-- ROW INSERTED SUCESSFULLY.



-- REMOVING THE DATA INSERTED


DELETE FROM staff;


-- OUTPUT

-- 1 ROW DELETED


SELECT staff_id FROM staff;

--OUTPUT

-- NO ROW SELECTED.
```

```
-----------------------------TEST FOR CHECK CONSTRAINT ON FIRSTNAME/LASTNAME COLUMN ('M/F')-------------------------------

-- BEFORE ASSIGNING CHECK CONSTRAINT TO STAFF TABLE

-- CHECKING ALREADY INSERTED ROWS IN staff

SELECT staff_id,firstname,lastname FROM staff;

-- TRYING TO INSERT IN staff Table

INSERT INTO staff (staff_id,firstname,lastname)
VALUES (1, 'BISHOWNATH','dhakal');

INSERT INTO staff (staff_id,firstname,lastname)
VALUES (2, 'sandeep','SHUBBA');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.

-- REMOVING THE DATA INSERTED

DELETE FROM staff;



-- AFTER ASSIGNING CHECK CONSTRAINT TO STAFF TABLE

-- CONSTRAINT FOR FIRSTNAME COLUMN
ALTER TABLE staff
ADD CONSTRAINT ck_staff_firstname
CHECK (firstname= UPPER(firstname));


--CONSTRAINT ON LASTNAME COLUMN
ALTER TABLE staff
ADD CONSTRAINT ck_staff_lastname
CHECK (lastname= UPPER(lastname));
```

```
INSERT INTO staff (staff_id,firstname,lastname)

VALUES (1, 'BISHOWNATH','dhakal');


INSERT INTO staff (staff_id,firstname,lastname)

VALUES (2, 'sandeep','SHUBBA');


-- EXPECTED OUTPUT / ACTUAL OUTPUT

-- check constraint (CSY2038B4.CK_STAFF_LASTNAME) violated

-- check constraint (CSY2038B4.CK_STAFF_FIRSTNAME) violated


-- INSERTING VALID DATA INTO THE TABLE STAFF



INSERT INTO staff (staff_id,firstname,lastname)

VALUES (1, 'BISHOWNATH','DHAKAL');


-- EXPECTED OUTPUT / ACTUAL OUTPUT

-- 1 ROW CREATED
-- REMOVING THE DATA INSERTED

DELETE FROM staff;

-- OUTPUT
-- 1 ROW DELETED

SELECT staff_id FROM staff;
--OUTPUT
-- NO ROW SELECTED.



------------------------------TEST FOR FOREIGN KEY CONSTRAINT-------------------------------

-- BEFORE ASSIGNING FOREIGN KEY TO STAFF TABLE

-- CHECKING ALREADY INSERTED ROWS IN staff
```

```
SELECT staff_id,firstname,leader FROM staff;

-- TRYING TO INSERT IN staff Table

INSERT INTO staff (staff_id,firstname,leader)
VALUES (1, 'BISHOWNATH',2);

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.

-- REMOVING THE DATA INSERTED

DELETE FROM staff;



-- AFTER ASSIGNING FOREIGN KEY TO STAFF TABLE

ALTER TABLE staff
ADD CONSTRAINT fk_s_staff
FOREIGN KEY (leader)
```

```
REFERENCES staff(staff_id);



-- TRYING TO INSERT IN staff Table


INSERT INTO staff (staff_id,firstname,leader)
VALUES (1, 'BISHOWNATH',2);


-- EXPECTED OUTPUT / ACTUAL OUTPUT
--ORA-02291: integrity constraint (CSY2038B4.FK_S_STAFF) violated - parent key



-- INSERTING VALID DATA IN staff table


INSERT INTO staff (staff_id,firstname,leader)
VALUES (1, 'BISHOWNATH',1);


-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.
-- REMOVING THE DATA INSERTED


DELETE FROM staff;


-- OUTPUT

-- 1 ROW DELETED


SELECT staff_id FROM staff;

--OUTPUT

-- NO ROW SELECTED.










-------------------------------------------------FESTIVAL_NATURES TABLE-------------------------------------------------------


---------------------------------TEST FOR PRIMARY KEY CONSTRAINT-----------------------------------
```

```
-- BEFORE ASSIGNING PRIMARY KEY TO FESTIVAL_NATURES TABLE

-- CHECKING ALREADY INSERTED ROWS IN FESTIVAL_NATURES

SELECT festival_nature_id,name FROM festival_natures;

-- TRYING TO INSERT IN FESTIVAL_NATURES Table

INSERT INTO festival_natures(festival_nature_id,name)
VALUES (1, 'CONCERT');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.

-- AFTER ASSIGNING PRIMARY KEY TO FESTIVAL_NATURES TABLE
-- CREATING PRIMARY KEY FOR FESTIVAL_NATURES TABLE

ALTER TABLE festival_natures
ADD CONSTRAINT pk_festival_natures
PRIMARY KEY (festival_nature_id);

-- TRYING TO INSERT IN FESTIVAL_NATURES Table

INSERT INTO festival_natures(festival_nature_id,name)
VALUES (1, 'CONCERT');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ORA-00001: unique constraint (CSY2038B4.PK_FESTIVAL_NATURES) violated


-- TRYING TO INSERT VALID DATA IN FESTIVAL_NATURES Table

INSERT INTO festival_natures(festival_nature_id,name)
VALUES (2, 'CHILDREN FAIR');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.

-- DELETING ALL INSERTED DATA FROM FESTIVAL_NATURES TABLE

DELETE FROM festival_natures;

SELECT festival_nature_id,name FROM festival_natures;

-- EXPECTED / ACUTAL OUTPUT
-- NO ROWS SELECTED




------------------------------TEST FOR CHECK CONSTRAINT ON DIFFERENT COLUMNS-------------------------------

-- BEFORE ASSIGNING CHECK CONSTRAINTS TO festival_natures table


-- CHECKING ALREADY INSERTED ROWS IN FESTIVAL_NATURES
```

```
SELECT festival_nature_id,name FROM festival_natures;

-- TRYING TO INSERT IN FESTIVAL_NATURES Table

INSERT INTO festival_natures(festival_nature_id,name,target_audience)
VALUES (1, 'CONCERT','adult');

INSERT INTO festival_natures(festival_nature_id,name,target_audience)
VALUES (2, 'children fair','CHILDREN');

-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.

-- DELETEING RECENTLY ADDED DATA

DELETE FROM festival_natures;

-- AFTER ASSIGNING CHECK CONSTRAINTS TO festival_natures table

-- CREATING CHECK CONSTRAINTS FOR FESTIVAL_NATURES TABLE
```

```
--CONSTRAINT FOR NAME COLUMN

ALTER TABLE festival_natures

ADD CONSTRAINT ck_festival_natures_name

CHECK (name= UPPER(name));


--CONSTRAINT FOR TARGET_AUDIENCE COLUMN

ALTER TABLE festival_natures

ADD CONSTRAINT ck_festival_natures_taudience

CHECK (target_audience= UPPER(target_audience));



-- TRYING TO INSERT IN FESTIVAL_NATURES Table


INSERT INTO festival_natures(festival_nature_id,name,target_audience)

VALUES (1, 'CONCERT','adult');


INSERT INTO festival_natures(festival_nature_id,name,target_audience)

VALUES (2, 'children fair','CHILDREN');


-- EXPECTED OUTPUT / ACTUAL OUTPUT

-- ORA-02290: check constraint (CSY2038B4.CK_FESTIVAL_NATURES_NAME) violated
```

```
--ORA-02290: check constraint (CSY2038B4.CK_FESTIVAL_NATURES_TAUDIENCE) violated


-- TRYING TO INSERT VALID DATA IN FESTIVAL_NATURES Table


INSERT INTO festival_natures(festival_nature_id,name,target_audience)
VALUES (1, 'CONCERT','ADULT');



-- EXPECTED OUTPUT / ACTUAL OUTPUT
-- ROW INSERTED SUCESSFULLY.


-- DELETING ALL INSERTED DATA FROM FESTIVAL_NATURES TABLE


DELETE FROM festival_natures;


SELECT festival_nature_id,name FROM festival_natures;


-- EXPECTED / ACUTAL OUTPUT
-- NO ROWS SELECTED
```

```
------------------------------------------LOCATIONS TABLE-----------------------------------------------

-----------------------------TEST FOR PRIMARY KEY CONSTRAINT-------------------------------

-- BEFORE ASSIGNING PRIMARY KEY CONSTRAINT TO locations table



-- CHECKING ALREADY INSERTED ROWS IN locations


SELECT location_id,capacity FROM locations;


-- EXPECTED / ACUTAL OUTPUT
-- NO ROWS SELECTED


-- TRYING TO INSERT IN locations Table
```

```
INSERT INTO locations (location_id,capacity)

VALUES (1,50000);


-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.


-- AFTER ASSIGNING PRIMARY KEY CONSTRAINT TO locations table


-- CREATING PRIMARY KEY FOR LOCATIONS TABLE


ALTER TABLE locations

ADD CONSTRAINT pk_locations

PRIMARY KEY (location_id);



-- TRYING TO INSERT IN locations Table


INSERT INTO locations (location_id,capacity)

VALUES (1,50000);


-- EXPECTED / ACUTAL OUTPUT
```

```
--ORA-00001: unique constraint (CSY2038B4.PK_LOCATIONS) violated


-- TRYING TO INSERT IN  VALID DATA locations Table


INSERT INTO locations (location_id,capacity)
VALUES (2,60000);

-- EXPECTED / ACUTAL OUTPUT
-- 1 row created.


-- DELETING ALL INSERTED DATA FROM LOCATIONS TABLE

DELETE FROM locations;

SELECT location_id,capacity FROM locations;

-- EXPECTED / ACUTAL OUTPUT
-- NO ROWS SELECTED
```

```
-------------------------------------------FESTIVALS TABLE------------------------------------------------

-----------------------------TEST FOR PRIMARY KEY AND FOREIGN KEY CONSTRAINT------------------------------


-- BEFORE ASSIGNING ANY KIND OF CONSTRAINTS IN festivals table.


-- CHECKING ALL COLUMN FOR festivals table
DESC festivals;


-- CHECKING ALL COLUMN FOR festival_natures table
DESC festival_natures;


-- QUERYING festival_nature_id FROM festival_natures table
```

```
SELECT festival_nature_id,name FROM festival_natures;


-- INSERTING INTO festival_natures table


INSERT INTO festival_natures(festival_nature_id,name)
VALUES (1, 'CONCERT');


-- EXPECTED / ACUTAL OUTPUT
-- 1 row created.



-- CHECKING ALL COLUMN FOR locations table
DESC locations;



-- TRYING TO INSERT IN locations Table


INSERT INTO locations (location_id,capacity)
VALUES (1,50000);


-- EXPECTED / ACUTAL OUTPUT
-- 1 row created.



-- CHECKING FOR PREVIOUSLY INSERTED DATA IN festivals table

SELECT festival_nature_id,location_id FROM festivals;


-- TRYING TO INSERT IN festivals Table

INSERT INTO festivals (festival_nature_id,location_id,festival_name)

VALUES (1,1,'CHRISTMAS');


-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.


-- AFTER CREATING BOTH PRIMARY KEY AND FOREIGN KEY CONSTRAINTS


-- CREATING PRIMARY KEY CONSTRAINTS


ALTER TABLE festivals

ADD CONSTRAINT pk_festivals

PRIMARY KEY (festival_nature_id, location_id);
```

```
-- CREATING FOREIGN KEY CONTRAINTS

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

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


-- TRYING TO INSERT IN festivals Table
INSERT INTO festivals (festival_nature_id,location_id,festival_name)
VALUES (1,1,'CHRISTMAS');

-- EXPECTED / ACUTAL OUTPUT
--ORA-00001: unique constraint (CSY2038B4.PK_FESTIVALS) violated
```

```
INSERT INTO festivals (festival_nature_id,location_id,festival_name)

VALUES (1,2,'CHRISTMAS');

-- EXPECTED / ACUTAL OUTPUT

--ORA-02291: integrity constraint (CSY2038B4.FK_F_LOCATIONS) violated - parent

--key not found




-- DELETING PREVIOUSLY INSERTED ROWS FROM festivals table


DELETE FROM festivals;



-- TRYING TO INSERT VALID DATA IN festivals table


INSERT INTO festivals (festival_nature_id,location_id,festival_name)

VALUES (1,1,'CHRISTMAS');


-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.
-- DELETING PREVIOUSLY INSERTED ROWS FROM festival_natures table

DELETE FROM festival_natures;


SELECT festival_nature_id,location_id FROM festivals;

-- EXPECTED / ACUTAL OUTPUT

-- NO ROWS SELECTED




-----------------------------TEST FOR CHECK CONSTRAINT------------------------------



-- BEFORE ASSIGNING CHECK CONSTRAINT IN festivals table.


-- CHECKING FOR PREVIOUSLY INSERTED DATA IN festivals table

SELECT festival_nature_id,location_id FROM festivals;


-- TRYING TO INSERT IN festivals Table

INSERT INTO festivals (festival_nature_id,location_id,festival_name)

VALUES (1,1,'christmas');
```

```
-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.


-- DELETING PREVIOUSLY INSERTED ROWS from festivals table

DELETE FROM festivals;


-- AFTER ASSIGNING CHECK CONSTRAINT IN festivals table.

-- CHECK CONSTRAINT FOR festival_name column


ALTER TABLE festivals

ADD CONSTRAINT ck_festivals_festival_name

CHECK (festival_name = UPPER(festival_name));


-- TRYING TO INSERT IN festivals Table

INSERT INTO festivals (festival_nature_id,location_id,festival_name)

VALUES (1,1,'christmas');


-- EXPECTED / ACUTAL OUTPUT

-- ORA-02290: check constraint (CSY2038B4.CK_FESTIVALS_FESTIVAL_NAME) violated
-- TRYING TO INSERT IN festivals Table

INSERT INTO festivals (festival_nature_id,location_id,festival_name)

VALUES (1,1,'CHRISTMAS');


-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.


-- DELETING PREVIOUSLY INSERTED ROWS FROM festivals table

DELETE FROM festivals;


-- DELETING PREVIOUSLY INSERTED ROWS FROM locations table

DELETE FROM locations;



-- DELETING PREVIOUSLY INSERTED ROWS FROM festival_natures table

DELETE FROM festival_natures;


SELECT festival_nature_id,location_id FROM festivals;

-- EXPECTED / ACUTAL OUTPUT

-- NO ROWS SELECTED
```

```
-------------------------------------------FESTIVAL_STAFF TABLE------------------------------------------------

------------------------------TEST FOR PRIMARY KEY AND FOREIGN KEY CONSTRAINT-------------------------------


-- BEFORE ASSIGNING ANY KIND OF CONSTRAINTS IN festival_staff table.


-- CHECKING ALL COLUMN FOR festival_staff table
DESC festivals;


-- CHECKING ALL COLUMN FOR festival_natures table
DESC festival_natures;


-- QUERYING festival_nature_id FROM festival_natures table
```

```
SELECT festival_nature_id,name FROM festival_natures;

-- INSERTING INTO festival_natures table

INSERT INTO festival_natures(festival_nature_id,name)
VALUES (1, 'CONCERT');

-- EXPECTED / ACUTAL OUTPUT
-- 1 row created.



-- CHECKING ALL COLUMN FOR locations table
DESC locations;



-- TRYING TO INSERT IN locations Table

INSERT INTO locations (location_id,capacity)
VALUES (1,50000);
```

```
-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.


-- CHECKING ALL COLUMN FOR staff table

DESC staff;


-- TRYING TO INSERT staff Table


INSERT INTO staff (staff_id,firstname)

VALUES (1,'BISHOWNATH');


-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.


-- CHECKING ALL COLUMN FOR festivals table

DESC festivals;


-- TRYING TO INSERT festivals Table


-- TRYING TO INSERT IN festivals Table

INSERT INTO festivals (festival_nature_id,location_id,festival_name)
```

```
VALUES (1,1,'CHRISTMAS');

-- EXPECTED / ACUTAL OUTPUT
-- 1 row created.



-- INSERTING INTO festival_staff table

-- CHECKING FOR PREVIOUSLY INSERTED DATA IN festival_name table

SELECT festival_staff_id,festival_nature_id,location_id,staff_id FROM festival_staff;
-- EXPECTED / ACUTAL OUTPUT
-- NO ROWS SELECTED

-- INSERTING DATA INTO festival_staff table

INSERT INTO festival_staff(festival_staff_id,festival_nature_id,location_id,staff_id)
VALUES (1,1,1,1);

-- EXPECTED / ACUTAL OUTPUT
-- 1 row created.
-- AFTER ASSIGNING ANY KIND OF CONSTRAINTS IN festival_staff table.

-- PRIMARY KEY FOR festival_staff table

ALTER TABLE festival_staff
ADD CONSTRAINT pk_festival_staff
PRIMARY KEY (festival_staff_id);

-- FOREIGN KEY FOR festival_staff table

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

ALTER TABLE festival_staff
ADD CONSTRAINT fk_fs_staff
FOREIGN KEY (staff_id)
REFERENCES staff(staff_id);
```

```
-- BEFORE ASSIGNING ANY KIND OF CONSTRAINTS IN festival_staff table.

-- INSERTING DATA INTO festival_staff table

INSERT INTO festival_staff(festival_staff_id,festival_nature_id,location_id,staff_id)
VALUES (1,1,1,1);

INSERT INTO festival_staff(festival_staff_id,festival_nature_id,location_id,staff_id)
VALUES (2,1,2,1);

-- EXPECTED / ACUTAL OUTPUT
--ORA-00001: unique constraint (CSY2038B4.PK_FESTIVAL_STAFF) violated

--ORA-02291: integrity constraint (CSY2038B4.FK_FS_FESTIVALS) violated - parent
--key not found

-- DELETING ALL ROW FROM festival_staff table
```

```
DELETE FROM festival_staff;


-- INSERTING DATA INTO festival_staff table


INSERT INTO festival_staff(festival_staff_id,festival_nature_id,location_id,staff_id)

VALUES (1,1,1,1);


-- EXPECTED / ACUTAL OUTPUT

-- 1 row created.




-- DELETING PREVIOUSLY INSERTED ROWS FROM festival_staff table

DELETE FROM festival_staff;


-- DELETING PREVIOUSLY INSERTED ROWS FROM festivals table

DELETE FROM festivals;


-- DELETING PREVIOUSLY INSERTED ROWS FROM locations table

DELETE FROM locations;
```

```
-- DELETING PREVIOUSLY INSERTED ROWS FROM festival_natures table

DELETE FROM festival_natures;


-- DELETING PREVIOUSLY INSERTED ROWS FROM staff table

DELETE FROM staff;




SELECT festival_staff_id,festival_nature_id,location_id,staff_id FROM festival_staff;

-- EXPECTED / ACUTAL OUTPUT

-- NO ROWS SELECTED




--------------------------------------------VIEW-------------------------------------------------



COLUMN constraint_name FORMAT A30;

SELECT constraint_name FROM user_constraints

WHERE constraint_name LIKE 'FK%';
```

```
COLUMN constraint_name FORMAT A30;

SELECT constraint_name FROM user_constraints

WHERE constraint_name LIKE 'PK%';


COLUMN constraint_name FORMAT A30;

SELECT constraint_name FROM user_constraints

WHERE constraint_name LIKE 'CK%';
```

6. Function_4.txt

```
/*

        FUNCTION FILE

                -Contains create commands for functions

                -These functions are used by the procedures or triggers


        GROUP 4

                18406547 - Diwas Lamsal

                18406498 - Bhuwan Khatiwada

                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal


*/
```

```
--@D:\Database_Assignment_II\script\function_4.txt


------------------------------------------------------------------------------------------------------------------------
-- Function to count total number of staff
CREATE OR REPLACE FUNCTION func_count_staff RETURN NUMBER IS
vn_count NUMBER(5);
BEGIN
        SELECT COUNT(staff_id)
        INTO vn_count
        FROM staff;
RETURN vn_count;
END func_count_staff;
/
SHOW ERRORS;


------------------------------------------------------------------------------------------------------------------------
--Function to count number of staff with salary more than provided amount
CREATE OR REPLACE FUNCTION func_count_staff_salary(in_salary NUMBER) RETURN NUMBER IS
```

```
vn_count NUMBER(5);
BEGIN
        SELECT COUNT(staff_id)
        INTO vn_count
        FROM staff
        WHERE salary>in_salary;
RETURN vn_count;
END func_count_staff_salary;
/
SHOW ERRORS;


----------------------------------------------------------------------------------------------------------------------
--Function to count number of festivals
CREATE OR REPLACE FUNCTION func_count_festivals RETURN NUMBER IS
vn_count NUMBER(5);
BEGIN
        SELECT COUNT(festival_nature_id)
        INTO vn_count
        FROM festivals;
RETURN vn_count;
```

```
END func_count_festivals;

/

SHOW ERRORS;



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

--Function to get the price of most expensive location

CREATE OR REPLACE FUNCTION func_exp_location RETURN NUMBER IS

vn_price locations.price%TYPE;

BEGIN

        SELECT MAX(price)

        INTO vn_price

        FROM locations;

RETURN vn_price;

END func_exp_location;

/

SHOW ERRORS;


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

--Function to get the price of cheapest location

CREATE OR REPLACE FUNCTION func_chp_location RETURN NUMBER IS
```

```
vn_price locations.price%TYPE;

BEGIN

            SELECT MIN(price)

            INTO vn_price

            FROM locations;

RETURN vn_price;

END func_chp_location;

/

SHOW ERRORS;



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


-- Function to Increase Salary of a Staff

-- Takes in the current salary and percentage to raise, and returns the raised salary

CREATE OR REPLACE FUNCTION func_increase_salary (in_current_salary NUMBER,in_percent IN NUMBER) RETURN NUMBER IS

vn_salary NUMBER(10,2);

BEGIN

            vn_salary:=(((in_percent)/100)*in_current_salary)+in_current_salary;

            RETURN vn_salary;

END func_increase_salary;
```

```
/
SHOW ERRORS

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

-- Function to Discount Price of a Location
-- Takes in the current price and percentage to discount, and returns the discounted price
CREATE OR REPLACE FUNCTION func_discount_location (in_current_price NUMBER, in_percent IN NUMBER) RETURN NUMBER IS
vn_price NUMBER(10,2);
BEGIN
	vn_price:= in_current_price - (((in_percent)/100)*in_current_price);
	RETURN vn_price;
END func_discount_location;
/
SHOW ERRORS

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

-- Function to retrieve staff username
-- Generates a username for the provided staff and returns it

CREATE OR REPLACE FUNCTION func_username(in_staff_id IN staff.staff_id%TYPE) RETURN VARCHAR2 IS

vc_username VARCHAR2(5);

BEGIN

        SELECT CONCAT(SUBSTR(firstname,1,3),SUBSTR(lastname,1,2))

        INTO vc_username

        FROM staff

        WHERE staff_id = in_staff_id;

RETURN vc_username;

END func_username;

/

SHOW ERRORS;


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



-- Function to retrieve staff password

-- Generates a password for the provided staff and returns it

CREATE OR REPLACE FUNCTION func_password(in_staff_id IN staff.staff_id%TYPE) RETURN VARCHAR2 IS

vc_date VARCHAR2(5);

vc_month VARCHAR2(5);
```

```
vc_name VARCHAR2(2);

vc_password VARCHAR2(5);

BEGIN

        SELECT SUBSTR(firstname,1,1)

        INTO vc_name

        FROM staff

        WHERE staff_id = in_staff_id;


        SELECT TO_CHAR(SYSDATE,'dd'),TO_CHAR(SYSDATE,'mm')

        INTO vc_date,vc_month

        FROM DUAL;

        vc_password := vc_date||vc_month||vc_name;

RETURN vc_password;

END func_password;

/

SHOW ERRORS;


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


-- Function to check if given string contains a number

-- If it contains a number, returns 1, else returns 0
```

```
CREATE OR REPLACE FUNCTION func_check_string_num(in_string VARCHAR2) RETURN NUMBER IS
        vn_flag NUMBER(1):= 0;
BEGIN
        FOR vn_count IN 0 .. 9 LOOP
                IF in_string LIKE '%'||vn_count||'%' THEN
                        vn_flag:= 1;
                END IF;
        END LOOP;
RETURN vn_flag;
END func_check_string_num;
/
SHOW ERRORS;
```

----

7. Func_test_4.txt

```
/*

        FUNC TEST FILE

                -Contains the tests for functions

                -Tests most of the functions using procedure

                -All the functions are directly implemented in procedure bodies without needing extra tests

                 and their results are visible also in the procedure tests

                -Some functions which can be tested separately are tested in this file


        GROUP 4


                18406547 - Diwas Lamsal

                18406498 - Bhuwan Khatiwada

                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal

*/


----------------------------------------THIS FILE CONTAINS FUNCTION TESTS----------------------------------------
```

```
SET SERVEROUTPUT ON

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

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




-- Procedure to test func_check_string_num

CREATE OR REPLACE PROCEDURE proc_ck_func_check_string_num AS

BEGIN

            DBMS_OUTPUT.PUT_LINE('Testing string GOOD result should be 0. Result: '||func_check_string_num('GOOD'));

            DBMS_OUTPUT.PUT_LINE('Testing string BAD999 result should be 1. Result: '||func_check_string_num('BAD999'));

            DBMS_OUTPUT.PUT_LINE('Testing string BAD00 result should be 1. Result: '||func_check_string_num('BAD00'));

END proc_ck_func_check_string_num;

/

SHOW ERRORS


--------------------- TEST PROC_CK_FUNC_CHECK_STRING_NUM ------------------


--Test the function func_check_string_num using procedure

--The output shows the test results
```

```
EXEC proc_ck_func_check_string_num;

--If the expected and obtained results differ, the function has error(s)



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

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




-- Procedure to test func_username and func_password

CREATE OR REPLACE PROCEDURE proc_test_username_password(in_staff_id IN staff.staff_id%TYPE) IS

vc_username VARCHAR2(5);

vc_password VARCHAR2(5);

BEGIN

        vc_username:= func_username(in_staff_id);

        vc_password:=func_password(in_staff_id);

        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');

        DBMS_OUTPUT.PUT_LINE('USERNAME = '||vc_username);

        DBMS_OUTPUT.PUT_LINE('PASSWORD = '||vc_password);

END proc_test_username_password;

/
```

```
SHOW ERRORS

--------------------- TEST PROC_TEST_USERNAME_PASSWORD ------------------

--Test the functions using procedure
--The output shows the test results
EXEC proc_test_username_password(5);

--Expected : The generated username and password is displayed
--Result   : The generated username and password is displayed

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



-- Procedure to test func_increase_salary
CREATE OR REPLACE PROCEDURE proc_test_func_increase_salary(in_current_salary NUMBER,in_percent IN NUMBER) IS
        vn_salary NUMBER(10,2);
BEGIN
        vn_salary:=func_increase_salary(in_current_salary, in_percent);
        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('Previous Salary : '||in_current_salary);
        DBMS_OUTPUT.PUT_LINE('Current Salary  : '||vn_salary);
END proc_test_func_increase_salary;
/
SHOW ERRORS

--------------------- TEST PROC_TEST_USERNAME_PASSWORD ------------------

--Test the functions using procedure
--The output shows the previous price and discounted price
EXEC proc_test_func_increase_salary(1000, 10);

--Expected : Correct previous and increased salary is displayed (1000 and 1100)
--Result   : Correct previous and increased salary is displayed (1000 and 1100)

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

```
-- Procedure to test func_discount_location

CREATE OR REPLACE PROCEDURE proc_test_func_discount_location(in_current_price NUMBER, in_percent IN NUMBER) IS

        vn_price NUMBER(10,2);

BEGIN

        vn_price:=func_discount_location(in_current_price, in_percent);

        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');

        DBMS_OUTPUT.PUT_LINE('Previous Price       : '||in_current_price);

        DBMS_OUTPUT.PUT_LINE('Price After Discount : '||vn_price);

END proc_test_func_discount_location;

/

SHOW ERRORS


-------------------- TEST PROC_TEST_USERNAME_PASSWORD -----------------


--Test the functions using procedure

--The output shows the previous price and discounted price

EXEC proc_test_func_discount_location(1000, 10);


--Expected : Correct previous and discounted price is displayed (1000 and 900)

--Result   : Correct previous and discounted price is displayed (1000 and 900)
```

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

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







--Drops for function testing procedures

DROP PROCEDURE proc_ck_func_check_string_num;

DROP PROCEDURE proc_test_username_password;

DROP PROCEDURE proc_test_func_increase_salary;

DROP PROCEDURE proc_test_func_discount_location;
```

8. Procedure_4.txt

```
/*


	PROCEDURE FILE

		-Contains create commands for procedures

		-Procedures using cursors are in a separate script file


	GROUP 4

		18406547 - Diwas Lamsal
```

```
                    18406498 - Bhuwan Khatiwada

                    18413710 - Sandip Kumar Subba

                    18413644 - Bishownath Dhakal

*/



--@D:\Database_Assignment_II\script\procedure_4.txt

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




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

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

----------------------------------Procedures for inserting data into tables and object tables------------------------------

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

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


------------------------ procedure for inserting into addresses object table ------------------
```

```
CREATE OR REPLACE PROCEDURE proc_insert_addresses(in_street IN VARCHAR2:=NULL, in_city IN VARCHAR2:=NULL, in_country IN
VARCHAR2:=NULL) IS

BEGIN
        INSERT INTO addresses (street,city,country)
        VALUES (in_street,in_city,in_country);
        DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCESSFULLY');
END proc_insert_addresses;
/

SHOW ERRORS;

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

------------------------ procedure for inserting into festival_natures table ------------------

CREATE OR REPLACE PROCEDURE proc_insert_festival_natures(in_name IN festival_natures.name%TYPE, in_targeted_audience IN
festival_natures.target_audience%TYPE) IS

BEGIN
        INSERT INTO festival_natures (festival_nature_id,name,target_audience)
        VALUES (seq_festival_nature_id.NEXTVAL,in_name,in_targeted_audience);

        DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCESSFULLY');

END proc_insert_festival_natures;

/


SHOW ERRORS;



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


------------------------- procedure for inserting into locations table ----------------------


CREATE OR REPLACE PROCEDURE proc_insert_locations(in_capacity IN locations.capacity%TYPE,in_price IN locations.price%TYPE,in_address_ref
IN VARCHAR2) IS


BEGIN

        INSERT INTO locations (location_id,capacity,price,address)

        SELECT seq_location_id.NEXTVAL,in_capacity,in_price, REF(a)

        FROM addresses a

        WHERE a.street = in_address_ref;

        DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCESSFULLY');
```

```
END proc_insert_locations;

/


SHOW ERRORS;



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


------------------------ procedure for inserting into staff table ----------------------



CREATE OR REPLACE PROCEDURE proc_insert_staff(in_firstname VARCHAR2, in_lastname VARCHAR2, in_gender CHAR,
                                                                                                    in_num_t1 VARCHAR2, in_type_t1 VARCHAR2,
                                                                                                    in_num_t2 VARCHAR2, in_type_t2 VARCHAR2,
                                                                                                    in_street_c VARCHAR2, in_city_c VARCHAR2,
in_country_c VARCHAR2,
                                                                                                    in_street_p VARCHAR2, in_city_p VARCHAR2,
in_country_p VARCHAR2,
                                                                                                    in_email VARCHAR2, in_leader NUMBER, in_salary
NUMBER) IS


BEGIN
```

```
        INSERT INTO staff (staff_id, firstname, lastname, gender, contact, current_address, permanent_address, email, leader, salary)
        VALUES (seq_staff_id.NEXTVAL, in_firstname, in_lastname, in_gender, contact_detail_varray_type(contact_detail_type (in_num_t1,
in_type_t1),

                                        contact_detail_type (in_num_t2, in_type_t2))

        , address_type(in_street_c, in_city_c, in_country_c)

        , address_type(in_street_p, in_city_p, in_country_p)

        , in_email, in_leader, in_salary);

   DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCESSFULLY');

END proc_insert_staff;
/

SHOW ERRORS;

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

------------------------ procedure for inserting into festivals table ----------------------
```

```
CREATE OR REPLACE PROCEDURE proc_insert_festivals(in_festival_nature_id NUMBER, in_location_id NUMBER, in_name VARCHAR2,
                                                                                                                                  in_p1_name VARCHAR2, in_p1_artist VARCHAR2,
in_p1_genre VARCHAR2,
                                                                                                                                          in_p2_name VARCHAR2, in_p2_artist
VARCHAR2, in_p2_genre VARCHAR2) IS

BEGIN
            INSERT INTO festivals (festival_nature_id, location_id, festival_name, performance)
            VALUES (in_festival_nature_id, in_location_id, in_name, performance_table_type(performance_type(in_p1_name, in_p1_artist,
in_p1_genre),

              performance_type(in_p2_name, in_p2_artist, in_p2_genre)));

   DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCESSFULLY');
END proc_insert_festivals;
/

SHOW ERRORS;

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

```
------------------------ procedure for inserting into festival_staff table ---------------------


CREATE OR REPLACE PROCEDURE proc_insert_festival_staff(in_staff_id IN festival_staff.staff_id%TYPE,in_festival_nature_id IN
festival_staff.festival_nature_id%TYPE,in_location_id IN festival_staff.location_id%TYPE) IS


BEGIN


            INSERT INTO festival_staff (festival_staff_id,staff_id,festival_nature_id,location_id)

            VALUES (seq_festival_staff_id.NEXTVAL,in_staff_id,in_festival_nature_id,in_location_id);

   DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCESSFULLY');


END proc_insert_festival_staff;

/


SHOW ERRORS;
```

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

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

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

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




--PROCEDURE TO RESET SEQUENCE


-- H., Porter, D. and Heller, J. (2019).


--This procedure takes a sequence as a parameter and resets it to 0

--Usage of EXECUTE IMMEDIATE

CREATE OR REPLACE PROCEDURE proc_reset_seq(p_seq_name IN VARCHAR2)

IS

   l_val NUMBER;

BEGIN

   EXECUTE IMMEDIATE

   'select ' || p_seq_name || '.nextval from dual' INTO l_val;

   EXECUTE IMMEDIATE
```

```
    'alter sequence ' || p_seq_name || ' increment by -' || l_val || ' minvalue 0';

    EXECUTE IMMEDIATE

    'select ' || p_seq_name || '.nextval from dual' INTO l_val;

    EXECUTE IMMEDIATE

    'alter sequence ' || p_seq_name || ' increment by 1 minvalue 0';

END proc_reset_seq;

/

SHOW ERRORS






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

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

-------------------------------------------------Procedures for querying data---------------------------------------------

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

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



--Procedures using functions
```

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

-- This procedure displays the total number of staff
-- It uses the function func_count_staff to retrieve total number of staff
CREATE OR REPLACE PROCEDURE proc_count_staff AS
        vn_count NUMBER(5):= func_count_staff;
BEGIN
        IF vn_count>0 THEN
                DBMS_OUTPUT.PUT_LINE('There are '|| vn_count || ' staff in the system.');
        ELSE
                DBMS_OUTPUT.PUT_LINE('There is no staff available.');
        END IF;
END proc_count_staff;
/
SHOW ERRORS

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

-- This procedure displays the total number of staff with salary higher than provided as argument
-- It uses the function func_count_staff_salary to retrieve the number of staff
```

```
CREATE OR REPLACE PROCEDURE proc_count_staff_salary(in_salary NUMBER) AS
        vn_count NUMBER(5):= func_count_staff_salary(in_salary);
BEGIN
        IF vn_count>1 THEN
                DBMS_OUTPUT.PUT_LINE('There are '|| vn_count || ' staff in the system with salary more than '|| in_salary);
        ELSIF vn_count = 1 THEN
                DBMS_OUTPUT.PUT_LINE('There is '|| vn_count || ' staff in the system with salary more than '|| in_salary);
        ELSE
                DBMS_OUTPUT.PUT_LINE('There is no staff with salary more than '|| in_salary);
        END IF;
END proc_count_staff_salary;
/
SHOW ERRORS

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


-- This procedure displays the total number of festivals
-- It uses the function func_count_festivals to retrieve total number of festivals
CREATE OR REPLACE PROCEDURE proc_count_festivals AS
        vn_count NUMBER(5):= func_count_festivals;
BEGIN
```

```
        IF vn_count>0 THEN
                DBMS_OUTPUT.PUT_LINE('There are '|| vn_count || ' festivals in the system.');
        ELSE
                DBMS_OUTPUT.PUT_LINE('There is no festival available.');
        END IF;
END proc_count_festivals;
/
SHOW ERRORS


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


-- This procedure displays the highest price among the locations
CREATE OR REPLACE PROCEDURE proc_exp_location AS
BEGIN
        DBMS_OUTPUT.PUT_LINE('The most expensive location costs '|| func_exp_location);
END proc_exp_location;
/
SHOW ERRORS
```

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


-- This procedure displays the cheapest price among the locations

CREATE OR REPLACE PROCEDURE proc_chp_location AS

BEGIN

	DBMS_OUTPUT.PUT_LINE('The cheapest location costs '|| func_chp_location);

END proc_chp_location;

/

SHOW ERRORS



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




--Exception handling- Techonthenet.com (2019)


------------------------------- Getting location,capacity and nature of festivals------------------------

-- This procedure displays a festival's details

-- It takes festival name as the parameter

-- If the festival is not found, the error is handled and relevant message is displayed
```

```
CREATE OR REPLACE PROCEDURE proc_festival_detail(in_festival_name IN festivals.festival_name%TYPE) IS

    vc_street addresses.street%TYPE;
    vc_city addresses.city%TYPE;
    vc_country addresses.country%TYPE;
    vn_capacity locations.capacity%TYPE;
    vn_price locations.price%TYPE;
    vc_nature festival_natures.name%TYPE;
    vc_target_audience festival_natures.target_audience%TYPE;

BEGIN
        SELECT l.address.street,l.address.city,l.address.country,l.capacity,l.price, fn.name,fn.target_audience
        INTO vc_street,vc_city,vc_country,vn_capacity, vn_price, vc_nature,vc_target_audience
        FROM locations l
        JOIN festivals f ON l.location_id = f.location_id
        JOIN festival_natures fn ON f.festival_nature_id = fn.festival_nature_id
        WHERE f.festival_name = in_festival_name;

        DBMS_OUTPUT.PUT_LINE('FESTIVAL DETAILS');
        DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');
```

```
          DBMS_OUTPUT.PUT_LINE('Name              :'||in_festival_name);
          DBMS_OUTPUT.PUT_LINE('Location');
          DBMS_OUTPUT.PUT_LINE('--------');
          DBMS_OUTPUT.PUT_LINE('Street            :'||vc_street);
          DBMS_OUTPUT.PUT_LINE('City              :'||vc_city);
          DBMS_OUTPUT.PUT_LINE('Country           :'||vc_country);
          DBMS_OUTPUT.PUT_LINE('Capacity          :'||vn_capacity);
          DBMS_OUTPUT.PUT_LINE('Price             :'||vn_price);
          DBMS_OUTPUT.PUT_LINE('Nature');
          DBMS_OUTPUT.PUT_LINE('------');
          DBMS_OUTPUT.PUT_LINE('Nature            :'||vc_nature);
          DBMS_OUTPUT.PUT_LINE('Target Audience   :'||vc_target_audience);
          DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');

EXCEPTION
     WHEN no_data_found THEN
          DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');
          RAISE_APPLICATION_ERROR (-20001,'ERROR! NO SUCH DATA FOUND! DID YOU ENTER A VALID NAME?');
     WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');
          RAISE_APPLICATION_ERROR (-20001,'AN ERROR OCCURED');
```

```
END proc_festival_detail;

/

SHOW ERRORS



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




-------------------------- Querying staff permanent address and temporary address--------------------------

-- This procedure displays a staff's address details

-- It takes staff_id as the parameter

-- If the staff is not found, the error is handled and relevant message is displayed


CREATE OR REPLACE PROCEDURE proc_show_staff_address(in_staff_id staff.staff_id%TYPE) IS


-- Declaring different variable needed.


   vc_firstname staff.firstname%TYPE;

           vc_lastname staff.firstname%TYPE;

           vc_temp_street addresses.street%TYPE;
```

```
        vc_temp_city addresses.city%TYPE;

        vc_temp_country addresses.country%TYPE;

        vc_per_street addresses.street%TYPE;

        vc_per_city addresses.city%TYPE;

        vc_per_country addresses.country%TYPE;

        c_gender CHAR;

        vn_salary staff.salary%TYPE;

        vc_email staff.email%TYPE;

        vc_leader VARCHAR2(60);


BEGIN

        -- querying data from staff table


        SELECT s.firstname,s.lastname,s.gender,s.salary,s.email,CONCAT(CONCAT(l.firstname, ' '),
l.lastname),s.permanent_address.street,s.permanent_address.city ,s.permanent_address.country ,s.current_address.street
,s.current_address.city,s.current_address.country

        INTO
vc_firstname,vc_lastname,c_gender,vn_salary,vc_email,vc_leader,vc_per_street,vc_per_city,vc_per_country,vc_temp_street,vc_temp_city,vc_t
emp_country

        FROM staff s JOIN staff l

        ON s.leader = l.staff_id

        WHERE s.staff_id = in_staff_id;
```

```
-- projecting output in the screen
        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('Name              :'||vc_firstname||' '||vc_lastname);
        DBMS_OUTPUT.PUT_LINE('Gender            :'||c_gender );
        DBMS_OUTPUT.PUT_LINE('Salary            :'||vn_salary);
        DBMS_OUTPUT.PUT_LINE('-------------------------------------');
        DBMS_OUTPUT.PUT_LINE('TEMPORARY ADDRESS');
        DBMS_OUTPUT.PUT_LINE('------------------');
        DBMS_OUTPUT.PUT_LINE('STREET            :'||vc_temp_street);
        DBMS_OUTPUT.PUT_LINE('CITY              :'||vc_temp_city);
        DBMS_OUTPUT.PUT_LINE('COUNTRY           :'||vc_temp_country);
        DBMS_OUTPUT.PUT_LINE('-------------------------------------');
        DBMS_OUTPUT.PUT_LINE('PERMANENT ADDRESS');
        DBMS_OUTPUT.PUT_LINE('------------------');
        DBMS_OUTPUT.PUT_LINE('STREET            :'||vc_per_street);
        DBMS_OUTPUT.PUT_LINE('CITY              :'||vc_per_city);
        DBMS_OUTPUT.PUT_LINE('COUNTRY           :'||vc_per_country);
        DBMS_OUTPUT.PUT_LINE('-------------------------------------');
        DBMS_OUTPUT.PUT_LINE('Email             :'||vc_email);
        DBMS_OUTPUT.PUT_LINE('Leader            :'||vc_leader);
```

```
          DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');

EXCEPTION
          WHEN no_data_found THEN
                    DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');
                    RAISE_APPLICATION_ERROR (-20001,'ERROR! NO SUCH DATA FOUND! DID YOU ENTER A VALID VALUE?');
          WHEN OTHERS THEN
                    DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');
                    RAISE_APPLICATION_ERROR (-20001,'AN ERROR OCCURED');
END proc_show_staff_address;
/

SHOW ERRORS


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

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

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

----------------------------------------------Procedures with other purposes--------------------------------------------

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

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



-- This procedure increases salary of the staff provided

-- The first parameter takes staff_id for providing the staff whose salary is to be raised

-- The second parameter takes the percentage the salary is to be raised

CREATE OR REPLACE PROCEDURE proc_increase_salary(in_staff_id IN staff.staff_id%TYPE,in_percent IN NUMBER) IS


	vn_salary NUMBER(10,2);

	vn_previous_salary NUMBER(10,2);


BEGIN

	SELECT salary

	INTO vn_previous_salary

	FROM staff

	WHERE staff_id= in_staff_id;
```

```
        vn_salary:=func_increase_salary(vn_previous_salary, in_percent);

        UPDATE staff
        SET salary = vn_salary
        WHERE staff_id = in_staff_id;

        DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('Salary Sucessfully Updated');
        DBMS_OUTPUT.PUT_LINE('Previous Salary : '||vn_previous_salary);
        DBMS_OUTPUT.PUT_LINE('Current Salary  : '||vn_salary);

END proc_increase_salary;
/

SHOW ERRORS


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


-- This procedure discounts price of the location provided
-- The first parameter takes location_id for providing the location for which the price is to be discounted

-- The second parameter takes the percentage the price is to be reduced

CREATE OR REPLACE PROCEDURE proc_discount_location(in_location_id IN locations.location_id%TYPE, in_percent IN NUMBER) IS


        vn_discounted NUMBER(10,2);

        vn_price NUMBER(10,2);


BEGIN

        SELECT price

        INTO vn_price

        FROM locations

        WHERE location_id= in_location_id;


        vn_discounted:=func_discount_location(vn_price, in_percent);


        UPDATE locations

        SET price = vn_discounted

        WHERE location_id = in_location_id;


        DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');

        DBMS_OUTPUT.PUT_LINE('Location Successfully Updated');
        DBMS_OUTPUT.PUT_LINE('Previous Price       : '||vn_price);

        DBMS_OUTPUT.PUT_LINE('Price After Discount : '||vn_discounted);


END proc_discount_location;

/


SHOW ERRORS


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




-- PROCEDURE TO GET USERNAME AND PASSWORD FROM STAFF ID

-- Uses the functions func_username and func_password

CREATE OR REPLACE PROCEDURE proc_username_password(in_staff_id IN staff.staff_id%TYPE) IS


vc_username VARCHAR2(5);

vc_password VARCHAR2(5);


BEGIN

        vc_username:= func_username(in_staff_id);

        vc_password:=func_password(in_staff_id);
```

```
		DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');
		DBMS_OUTPUT.PUT_LINE('USERNAME = '||vc_username);
		DBMS_OUTPUT.PUT_LINE('PASSWORD = '||vc_password);
EXCEPTION
WHEN no_data_found THEN
		DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');
		RAISE_APPLICATION_ERROR (-20001,'ERROR! NO SUCH DATA FOUND! DID YOU ENTER A VALID VALUE?');
WHEN OTHERS THEN
		DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------');
		RAISE_APPLICATION_ERROR (-20001,'AN ERROR OCCURED');
END proc_username_password;
/
SHOW ERRORS

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

-- PROCEDURE TO FIND HOW MANY GROUPS OF SPECIFIC SIZE CAN FIT IN A LOCATION
-- The first parameter is location_id for the location to be checked
-- The second parameter is the group size
-- The procedure checks the number of groups of provided size that can fit in the provided location.

-- It also displays the number of seats not enough for adding another group of the same size


CREATE OR REPLACE PROCEDURE proc_location_capacity_ck(in_location_id NUMBER, in_group_size NUMBER) AS

	vn_num_groups NUMBER(3);

	vn_mod_groups NUMBER(3);

	vn_rem_groups NUMBER(3);

	vn_capacity locations.capacity%TYPE;


BEGIN

	SELECT capacity INTO vn_capacity FROM locations WHERE location_id = in_location_id;


	vn_num_groups := FLOOR(vn_capacity / in_group_size);

	vn_mod_groups := MOD(vn_capacity, in_group_size); -- MOD uses FLOOR

	vn_rem_groups := REMAINDER(vn_capacity, in_group_size); --Remainder uses ROUND


	IF vn_rem_groups<0 THEN

		vn_rem_groups := vn_rem_groups*-1;

	END IF;


	DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------');
```

```
            DBMS_OUTPUT.PUT_LINE(vn_num_groups||' GROUPS OF '||in_group_size||' CAN FIT IN THE PROVIDED LOCATION WITH CAPACITY
'||vn_capacity);

            DBMS_OUTPUT.PUT_LINE('NUMBER OF EMPTY SEATS: ' || vn_mod_groups);

            IF vn_rem_groups = vn_mod_groups THEN

                        vn_rem_groups := vn_num_groups * in_group_size + in_group_size - vn_capacity;

            END IF;

            DBMS_OUTPUT.PUT_LINE('NUMBER OF ADDITIONAL SEATS NEEDED FOR ADDING ANOTHER GROUP: ' || vn_rem_groups);


            DBMS_OUTPUT.PUT_LINE('---------------------------------------------------------------');

END proc_location_capacity_ck;

/

SHOW ERRORS


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


--This procedure displays first name of a staff as one character per line

--The parameter takes in staff_id

--Using loop end loop

CREATE OR REPLACE PROCEDURE proc_staff_firstname(in_staff_id NUMBER) AS

            vn_length NUMBER(3);

            vn_counter NUMBER(3):= 1;
```

```
        vc_firstname staff.firstname%TYPE;
BEGIN
        SELECT firstname INTO vc_firstname FROM staff WHERE staff_id = in_staff_id;
        vn_length := LENGTH(vc_firstname);
        LOOP
                DBMS_OUTPUT.PUT_LINE(SUBSTR(vc_firstname, vn_counter, 1));
                IF vn_counter = vn_length THEN
                        EXIT;
                END IF;
                vn_counter := vn_counter + 1;
        END LOOP;
END proc_staff_firstname;
/
SHOW ERRORS

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

--This procedure displays last name of a staff as one character per line
--The parameter takes in staff_id
--Using exit when
CREATE OR REPLACE PROCEDURE proc_staff_surname(in_staff_id NUMBER) AS
```

```
        vn_length NUMBER(3);

        vn_counter NUMBER(3):= 1;

        vc_surname staff.lastname%TYPE;

BEGIN

        SELECT lastname INTO vc_surname FROM staff WHERE staff_id = in_staff_id;

        vn_length := LENGTH(vc_surname);

        LOOP

                DBMS_OUTPUT.PUT_LINE(SUBSTR(vc_surname, vn_counter, 1));

                EXIT WHEN vn_counter = vn_length;

                vn_counter := vn_counter + 1;

        END LOOP;

END proc_staff_surname;

/

SHOW ERRORS
```

----

9. Proc_test_4.txt

```
/*

	PROC TEST FILE
		-Contains the tests for procedures
		-Tests all the procedures used (separate tests for cursor using procedures)

	GROUP 4

		18406547 - Diwas Lamsal
		18406498 - Bhuwan Khatiwada
		18413710 - Sandip Kumar Subba
		18413644 - Bishownath Dhakal

*/

--@D:\Database_Assignment_II\script\proc_test_4.txt



---------------------------------------THIS FILE CONTAINS PROCEDURE TESTS---------------------------------------

SET SERVEROUTPUT ON
```

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

--------------------- TEST PROC_INSERT_ADDRESSES ------------------


--Expected outcomes after each insert:

--1 row created.


--Insert without procedure

INSERT INTO addresses(street, city, country)

VALUES ('54 FESTIVE ROAD', 'NORTHAMPTON', 'UK');


--Expected outcomes after each inser:

--PL/SQL procedure successfully completed, and the data appears when querying


--Insert using procedure

EXEC proc_insert_addresses('111 BAY STATE ROAD', 'BOSTON', 'USA');


--Verify Two rows of data inserted

SELECT street, city, country FROM addresses;


--Expected result:

--Two rows of data with the same details provided
```

```
--Delete test data

DELETE FROM addresses WHERE street = '54 FESTIVE ROAD';

DELETE FROM addresses WHERE street = '111 BAY STATE ROAD';


--Verify Two rows of data deleted

SELECT street, city, country FROM addresses;


--Rows found: 0

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





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

--------------------- TEST PROC_INSERT_STAFF ------------------


--Verify empty table

SELECT staff_id, firstname, lastname, leader FROM staff;


--Expected outcomes after each insert:

--1 row created.
```

```
--Insert without procedure
INSERT INTO staff (staff_id, firstname, lastname, gender, contact, current_address, permanent_address, email, leader, salary)
VALUES (seq_staff_id.NEXTVAL, 'RAMESH', 'THAPA', 'M', contact_detail_varray_type(contact_detail_type('9808123457', 'MOBILE'),

        contact_detail_type('01-4216354', 'LANDLINE'))
                                                                                                                        ,
address_type('7500 IMPERIAL BLVD', 'LOS ANGELES', 'USA')
                                                                                                                        ,
address_type('333 SAN JUAN BLVD', 'SAN JUAN', 'PUERTO RICO')
                                                                                                                        ,
'RAMESHTHAPA@GMAIL.COM', seq_staff_id.CURRVAL, 12990);




--Expected outcomes after each insert:
--PL/SQL procedure successfully completed, and the data appears when querying


--Insert with procedure
EXEC proc_insert_staff('DIANE', 'BROWN', 'F', '(617)342-23442', 'LANDLINE', '981513244', 'MOBILE', '4242 MISTY LANE', 'SEATTLE', 'USA',
'KUNGSGATAN 56', 'STOCKHOLM', 'SWEDEN', 'DIANE_BROWN@HOTMAIL.COM', 1, 5000);
```

```
--Verify Two rows of data inserted

SELECT staff_id, firstname, lastname, leader FROM staff;


--Result: Two rows of data with provided details are inserted


--Delete test data

DELETE FROM staff;


--Reset Sequence for staff

EXEC proc_reset_seq('seq_staff_id');


--Verify Sequence Currval

SELECT seq_staff_id.CURRVAL FROM DUAL;


--Verify empty table

SELECT staff_id, firstname, lastname, leader FROM staff;


--Rows found: 0


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

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

--------------------- TEST PROC_INSERT_FESTIVAL_NATURES ------------------


--Verify empty table

SELECT * FROM festival_natures;


--Expected outcomes after each insert:

--1 row created.


--Insert without procedure

INSERT INTO festival_natures (festival_nature_id,name,target_audience)

VALUES (seq_festival_nature_id.NEXTVAL, 'PURAN', 'OLD');


--Expected outcomes after each insert:

--PL/SQL procedure successfully completed, and the data appears when querying


--Insert using procedure

EXEC proc_insert_festival_natures('CONCERT', 'YOUTH');
```

```
--Verify insert
SELECT * FROM festival_natures;

--Result: Inserted data appears when querying

--Delete test data
DELETE FROM festival_natures;

--Reset Sequence for festival_natures
EXEC proc_reset_seq('seq_festival_nature_id');

--Verify Sequence Currval
SELECT seq_festival_nature_id.CURRVAL FROM DUAL;

--Verify empty table
SELECT * FROM festival_natures;

--Rows found: 0
-------------------------------------------------------------------------------------------------------------------------
```

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

--------------------- TEST PROC_INSERT_LOCATIONS ------------------


--Note: Data is necessary in addresses table in order to test this procedure


--Verify empty table

SELECT location_id, capacity, price, l.address.street, l.address.city, l.address.country FROM locations l;


--Requires reference of addresses table


--Insert without procedure

INSERT INTO locations (location_id,capacity,price,address)

SELECT seq_location_id.NEXTVAL, 1500, 10000, REF(a)

FROM addresses a

WHERE a.street = '111 VALLEY WAY';


--Insert with procedure

EXEC proc_insert_locations(1200, 20500, '177 AIRPORT ROAD');


--Verify insert
```

```
SELECT location_id, capacity, price, l.address.street AS street, l.address.city  AS city, l.address.country  AS country FROM locations l ORDER BY
location_id;

--Result: Inserted data appears when querying

--Delete test data
DELETE FROM locations;

--Reset Sequence for locations
EXEC proc_reset_seq('seq_location_id');

--Verify Sequence Currval
SELECT seq_location_id.CURRVAL FROM DUAL;

--Verify empty table
SELECT location_id, capacity, price, l.address.street, l.address.city, l.address.country FROM locations l;

--Rows found: 0
-------------------------------------------------------------------------------------------------------------------------
```

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

------------------------- TEST PROC_INSERT_FESTIVALS ---------------------


--Note: Data is necessary in locations and festival_natures table in order to test this procedure


--Verify empty table

SELECT festival_nature_id, location_id, festival_name FROM festivals;


--Insert without procedure

INSERT INTO festivals (festival_nature_id, location_id, festival_name, performance)

VALUES (1, 1, 'BHAGAVATA', performance_table_type(performance_type('PRAWACHAN', 'GURU ARBINDRA NATH', 'RELIGIOUS'),

                                                                        performance_type('PRAWACHAN', 'GURU
ANUBHAVAM ACHARYA', 'RELIGIOUS')));



--Insert with procedure

EXEC proc_insert_festivals(2, 3, 'SCREAM FEST', 'SINGING', 'NEPATHYA', 'ROCK', 'MUSIC', 'ANTIM GRAHAN', 'METAL');


--Verify insert

SELECT festival_nature_id, location_id, festival_name FROM festivals;
```

```
--Result: Inserted data appears when querying


--Delete test data

DELETE FROM festivals;


--Verify empty table

SELECT festival_nature_id, location_id, festival_name FROM festivals;


--Rows found: 0

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




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

--------------------- TEST PROC_INSERT_FESTIVAL_STAFF ------------------


--Note*: Data is necessary in festivals and staff table in order to test this procedure


--Verify empty table

SELECT festival_staff_id, festival_nature_id, location_id, staff_id FROM festival_staff;
```

```
--Insert without procedure

INSERT INTO festival_staff (festival_staff_id,staff_id,festival_nature_id,location_id)

VALUES (seq_festival_staff_id.NEXTVAL, 1, 2, 3);


--Insert with procedure

EXEC proc_insert_festival_staff(3, 5, 1);


--Verify insert

SELECT festival_staff_id, festival_nature_id, location_id, staff_id FROM festival_staff;


--Result: Inserted data appears when querying


--Delete test data

DELETE FROM festivals;


--Reset Sequence for festival_staff

EXEC proc_reset_seq('seq_festival_staff_id');


--Verify Sequence Currval

SELECT seq_festival_staff_id.CURRVAL FROM DUAL;
```

```
--Verify empty table
SELECT festival_staff_id, festival_nature_id, location_id, staff_id FROM festival_staff;

--Rows found: 0
------------------------------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------------------------------
--------------------- TEST PROC_RESET_SEQ ------------------

--Increment the sequence position
SELECT seq_festival_staff_id.NEXTVAL FROM DUAL;
SELECT seq_festival_staff_id.NEXTVAL FROM DUAL;
SELECT seq_festival_staff_id.NEXTVAL FROM DUAL;

--Reset Sequence for festival_staff
EXEC proc_reset_seq('seq_festival_staff_id');

--Verify Sequence Currval
```

```
SELECT seq_festival_staff_id.CURRVAL FROM DUAL;


--Expected output: 0

--Result: 0


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


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

--------------------- Test procedures that display count  ------------------

--Note*: Execute the insert_4.txt SQL commands before testing these procedures


--Also testing the functions that these procedures use together

--Total number of staff

EXEC proc_count_staff;


--Using actual query

SELECT COUNT(staff_id) FROM staff;


--Expected : Shows total number of staff

--Result   : Shows total number of staff
```

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

--Total number of festivals
EXEC proc_count_festivals;

--Using actual query
SELECT COUNT(festival_id) FROM festivals;

--Expected : Shows total number of festivals
--Result   : Shows total number of festivals

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

--Highest price of location
EXEC proc_exp_location;

--Using actual query
SELECT MAX(price) FROM locations;

--Expected : Shows highest price for any location
--Result   : Shows highest price for any location
```

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

--Least price of location
EXEC proc_chp_location;

--Using actual query
SELECT MIN(price) FROM locations;

--Expected : Shows lowest price for any location
--Result   : Shows lowest price for any location

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



------------------------------------------------------------------------------------------------------------------------------
--------------------- TEST PROC_COUNT_STAFF_SALARY ------------------

--BOUND CHECKING

--Below lower bound
```

```
EXEC proc_count_staff_salary(-1);

--Expected : Procedure Works and total number of staff displayed

--Result   : Procedure Works and total number of staff displayed


--Minimum value

EXEC proc_count_staff_salary(0);

--Expected : Procedure Works and total number of staff displayed

--Result   : Procedure Works and total number of staff displayed


--Medium value

EXEC proc_count_staff_salary(9000);

--Expected : Procedure Works and total number of staff with salary above 9000 displayed

--Result   : Procedure Works and total number of staff with salary above 9000 displayed


--Maximum value

EXEC proc_count_staff_salary(99999999.99);

--Expected : Procedure Works and no staff with salary higher than 99999999.99 message displayed

--Result   : Procedure Works and no staff with salary higher than 99999999.99 message displayed


--Exceed upper bound

EXEC proc_count_staff_salary(1000000000);
```

```
--Expected : Procedure Works and no staff with salary higher than 1000000000 message displayed

--Result   : Procedure Works and no staff with salary higher than 1000000000 message displayed



--Using query to test medium value (to compare with output from procedure)

SELECT COUNT(staff_id) FROM staff WHERE salary > 9000;



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

--------------------- TEST PROC_STAFF_FIRSTNAME ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

EXEC proc_staff_firstname(1);


--Expected : Procedure displays the firstname of provided staff one character per line

--Result   : Procedure displays the firstname of provided staff one character per line

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



--------------------- TEST PROC_STAFF_SURNAME ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

EXEC proc_staff_surname(1);
```

```
--Expected : Procedure displays the lastname of provided staff one character per line

--Result   : Procedure displays the lastname of provided staff one character per line

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



--------------------- TEST PROC_LOCATION_CAPACITY_CK ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

EXEC proc_location_capacity_ck(3, 49);


--Expected : Procedure displays total number of groups that can fit in the provided location

--Result   : Procedure displays total number of groups that can fit in the provided location



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

--------------------- TEST PROC_USERNAME_PASSWORD ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

EXEC proc_username_password(2);


--Expected : Generated username and password combination is displayed

--Result   : Generated username and password combination is displayed
```

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

--------------------- TEST PROC_FESTIVAL_DETAIL ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

EXEC proc_festival_detail('BHAGAVATA');

EXEC proc_festival_detail('BHAGAVATAMA');


--Expected : Details of provided festival displayed. If no festival found, an error message is displayed (ERROR HANDLING)

--Result   : Details of provided festival displayed. If no festival found, an error message is displayed, i.e. for BHAGAVATAMA



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

--------------------- TEST PROC_SHOW_STAFF_ADDRESS ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

EXEC proc_show_staff_address(1);

EXEC proc_show_staff_address(99);


--Expected : Details of provided staff displayed. If no staff found, an error message is displayed (ERROR HANDLING)

--Result   : Details of provided staff displayed. If no staff found, an error message is displayed, i.e. for 99
```

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

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

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

                                                                                                                                                    --CURSOR TESTS

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

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

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



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

--------------------- TEST PROC_DEL_ADDRESS_CURSOR ------------------

--Note*: Execute the insert_4.txt SQL commands before testing the cursor

EXEC proc_del_address_cursor('111 BEACHCOMBER PLACE');

EXEC proc_del_address_cursor('RANDOM ADDRESS');


--Expected : Message saying address with provided street deleted is displayed if address is found, else, not found message is displayed

--Result   : Message saying address with provided street deleted is displayed if address is found, else, not found message is displayed (for second test)
```

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

--------------------- TEST PROC_NUM_LOCATIONS_PRICE ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure


--Query locations and price

SELECT location_id, price FROM locations;


--Execute procedures with cursor

EXEC proc_num_locations_price(200);

EXEC proc_num_locations_price(20000);


--Query locations and price

SELECT location_id, price FROM locations;


--Expected : Number of locations with price below provided price are displayed. Also adds 50 to the locations found.

--                              If not found, a message is displayed saying so


--Result   : Number of locations with price below provided price are displayed. Also adds 50 to the locations found.

--                              If not found, a message is displayed saying so (when providing price as 200)
```

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

--------------------- TEST PROC_VIEW_ADDRESSES_CURSOR ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

--Variable used as out parameter to display number of records

VAR TOTAL_LOCATIONS NUMBER;

EXEC proc_view_addresses_cursor(:TOTAL_LOCATIONS);


--Test of out parameter used in the cursor

PRINT TOTAL_LOCATIONS;


--Expected : Displays total number of locations, along with details

--Result   : Displays total number of locations, along with details



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

--------------------- TEST PROC_VIEW_ADDRESSES_CURSOR ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure

--Variable used as out parameter to display number of records

VAR TOTAL_FESTIVAL_NATURES NUMBER;
```

```
EXEC proc_view_festival_natures_cursor(:TOTAL_FESTIVAL_NATURES);


--Test of out parameter used in the cursor

PRINT TOTAL_FESTIVAL_NATURES;


--Expected : Displays total number of festival_natures, along with details. Also displays cursor is open or closed message

--Result   : Displays total number of festival_natures, along with details. Also displays cursor is open or closed message



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

--------------------- TEST PROC_VIEW_ADDRESSES_CURSOR ------------------

--Note*: Execute the insert_4.txt SQL commands before testing this procedure


EXEC proc_view_staff_contact_cursor;


--Expected : Displays contact details for each staff (both type of contact)

--Result   : Displays contact details for each staff (both type of contact)



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

--------------------- TEST PROC_VIEW_STAFF_SALARY_WEEKLY_CURSOR ------------------
```

```
--Note*: Execute the insert_4.txt SQL commands before testing this procedure


EXEC proc_view_staff_salary_weekly_cursor(3);

EXEC proc_view_staff_salary_weekly_cursor(99);


--Expected : Staff weekly salary details displayed if found, else, not found message is displayed

--Result   : Staff weekly salary details displayed if found, else, not found message is displayed for provided id 99





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

10. Trigger.txt

```
/*


	TRIGGER FILE

		-Contains create commands for triggers

		-Triggers for login logout information are in the log script file


	GROUP 4
```

```
                    18406547 - Diwas Lamsal

                    18406498 - Bhuwan Khatiwada

                    18413710 - Sandip Kumar Subba

                    18413644 - Bishownath Dhakal

*/

--@D:\Database_Assignment_II\script\trigger_4.txt

----------------------------------------------------------- TRIGGERS -------------------------------------------------------

--Triggers for addresses object table

--This trigger does not allow duplicate addresses to be entered into the addresses table
CREATE OR REPLACE TRIGGER trig_check_address
BEFORE INSERT ON addresses
FOR EACH ROW
DECLARE
     vn_count NUMBER(3);
```

```
BEGIN
        SELECT COUNT(street) INTO vn_count FROM addresses
        WHERE street = :NEW.street AND city = :NEW.city AND country = :NEW.country;
        IF vn_count>0 THEN
                RAISE_APPLICATION_ERROR (-20001,'ERROR! THE PROVIDED ADDRESS ALREADY EXISTS');
        END IF;
END trig_check_address;
/
SHOW ERRORS


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


--Triggers for staff


--This trigger does not allow an invalid email to be entered while adding or updating a staff
CREATE OR REPLACE TRIGGER trig_staff_email_ck
BEFORE INSERT OR UPDATE OF email ON staff
FOR EACH row
WHEN (NEW.email NOT LIKE '%@%' OR NEW.email NOT LIKE '%.%')
BEGIN
```

```
		RAISE_APPLICATION_ERROR (-20001,'ERROR! INVALID EMAIL FORMAT');
END trig_dob_ck;
/
SHOW ERRORS

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


--Switch Case
-- Tutorialspoint (2019)
-- Oracle Docs (2019)

--This trigger does not allow string with numbers provided for a staff's firstname or lastname
--It uses the function func_check_string_num which returns 1 when string contains a number or 0 when not
CREATE OR REPLACE TRIGGER trig_staff_name_ck
BEFORE INSERT OR UPDATE OF firstname, lastname ON staff
FOR EACH row
DECLARE
	vn_ck_firstname NUMBER(3);
	vn_ck_lastname NUMBER(3);
BEGIN
	vn_ck_firstname:= func_check_string_num(:NEW.firstname);
```

```
    vn_ck_lastname:= func_check_string_num(:NEW.lastname);


    --Implementation of switch case
    CASE vn_ck_firstname
        WHEN 0 THEN DBMS_OUTPUT.PUT_LINE('VALID FIRSTNAME');
        WHEN 1 THEN RAISE_APPLICATION_ERROR (-20001,'ERROR! INVALID FIRSTNAME! NAME CANNOT CONTAIN NUMBERS');
        ELSE DBMS_OUTPUT.PUT_LINE('SOMETHING WENT WRONG');
    END CASE;


    CASE vn_ck_lastname
        WHEN 0 THEN DBMS_OUTPUT.PUT_LINE('VALID LASTNAME');
        WHEN 1 THEN RAISE_APPLICATION_ERROR (-20001,'ERROR! INVALID LASTNAME! NAME CANNOT CONTAIN NUMBERS');
        ELSE DBMS_OUTPUT.PUT_LINE('SOMETHING WENT WRONG');
    END CASE;


END trig_dob_ck;
/
SHOW ERRORS

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

```
--This trigger displays a message saying you deleted this staff when a staff is deleted

CREATE OR REPLACE TRIGGER trig_del_staff

AFTER DELETE ON staff

FOR EACH row

BEGIN

	DBMS_OUTPUT.PUT_LINE('YOU DELETED THE STAFF '||:OLD.firstname || ' ' || :OLD.lastname);

END trig_del_staff;

/

SHOW ERRORS


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


--Triggers for locations


--This trigger does not allow a negative value to be entered for price or capacity of location

CREATE OR REPLACE TRIGGER trig_ck_location

BEFORE INSERT OR UPDATE OF price, capacity ON locations

FOR EACH row

WHEN (NEW.price < 0 OR NEW.capacity < 0)

BEGIN

	RAISE_APPLICATION_ERROR (-20001,'ERROR! CANNOT HAVE VALUES LESS THAN ZERO');
```

```
END trig_ck_location;
/
SHOW ERRORS

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

--Triggers for festival_natures

--This trigger displays messages as per the interactions made on festival_natures table , i.e.
--When inserting, updating or deleting a row, relevant messages are displayed
CREATE OR REPLACE TRIGGER trig_festival_natures
AFTER INSERT OR UPDATE OR DELETE ON festival_natures
FOR EACH row
BEGIN
	IF INSERTING THEN
		DBMS_OUTPUT.PUT_LINE('YOU INSERTED THE FESTIVAL NATURE '||:NEW.name || ' WITH FESTIVAL_NATURE_ID: '||
:NEW.festival_nature_id);
	ELSIF UPDATING THEN
		DBMS_OUTPUT.PUT_LINE('YOU UPDATED THE FESTIVAL NATURE '||:NEW.name || ' WITH FESTIVAL_NATURE_ID: '||
:OLD.festival_nature_id);
	ELSE
```

```
                    DBMS_OUTPUT.PUT_LINE('YOU DELETED THE FESTIVAL NATURE '||:OLD.name || ' WITH FESTIVAL_NATURE_ID: '||
:OLD.festival_nature_id);

          END IF;

END trig_festival_natures;

/

SHOW ERRORS


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


--Triggers for festival_staff


--This trigger displays messages as per the interactions made on festival_staff table, i.e.

--When inserting, updating or deleting a row, relevant messages are displayed

CREATE OR REPLACE TRIGGER trig_festival_staff

AFTER INSERT OR UPDATE OR DELETE ON festival_staff

FOR EACH row

BEGIN

          IF INSERTING THEN

                    DBMS_OUTPUT.PUT_LINE('YOU INSERTED A RECORD. FESTIVAL_STAFF_ID: '|| :NEW.festival_staff_id);

          ELSIF UPDATING THEN

                    DBMS_OUTPUT.PUT_LINE('YOU UPDATED A RECORD. FESTIVAL_STAFF_ID: '|| :OLD.festival_staff_id);
```

```
        ELSE
                DBMS_OUTPUT.PUT_LINE('YOU DELETED A RECORD. FESTIVAL_STAFF_ID: '|| :OLD.festival_staff_id);
        END IF;
END trig_festival_staff;
/
SHOW ERRORS

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

11. Trig_test_4.txt

```
/*

        TRIG TEST FILE
                -Contains the tests for triggers
                -Tests all the triggers used except the log detail triggers
                -Includes all the four types of tests needed for each trigger

        GROUP 4
```

```
            18406547 - Diwas Lamsal

            18406498 - Bhuwan Khatiwada

            18413710 - Sandip Kumar Subba

            18413644 - Bishownath Dhakal

*/



--@D:\Database_Assignment_II\script\trig_test_4.txt



--------------------------------------THIS FILE CONTAINS TRIGGER TESTS--------------------------------------



--Display dbms output

SET SERVEROUTPUT ON




--Query log details
```

```
COLUMN event_date FORMAT A20;

SELECT * FROM login_details;



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

---------------------- TEST trig_check_address


--This trigger is supposed to prevent adding an addresses that matches a previous one


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


--Insert dummy data first

INSERT INTO addresses

VALUES('RP STREET', 'RP CITY', 'RP COUNTRY');


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


--1 Fires where needed

--Try to insert the same data

INSERT INTO addresses

VALUES('RP STREET', 'RP CITY', 'RP COUNTRY');
```

```
--Expected outcome: Error is generated preventing the row from being inserted into the addressed table

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

--2 Does not fire where not needed
INSERT INTO addresses
VALUES('UNIQUE STREET', 'UNIQUE CITY', 'UNIQUE COUNTRY');
--Expected outcome: No errors are shown and the row is inserted into the addresses table

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

--3 Changes are made if it is supposed to
--Trigger should not allow repeated data to be inserted
SELECT street, city, country FROM addresses;

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

--4 Changes are not made if it is not supposed to
--Trigger should allow unique data to be inserted
SELECT street, city, country FROM addresses;
```

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

--RESET
DELETE FROM addresses WHERE street IN ('RP STREET','UNIQUE STREET');

--Test reset
SELECT street, city, country FROM addresses;
--Expected outcome: The dummy data inserted for testing trigger are removed



-----------------------------------------------------------------------------------------------------------------------------
---------------------- TEST trig_staff_email_ck

--This trigger is supposed to prevent adding a staff with invalid email address (with missing @ and . characters)

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

--First Insert a valid row
INSERT INTO staff(staff_id, firstname, lastname, email, leader)
VALUES (99999, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTVALID@VALID.COM', 99999);
```

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

--1 Fires where needed
INSERT INTO staff(staff_id, firstname, lastname, email, leader)
VALUES (99995, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTINVALID', 99995);

INSERT INTO staff(staff_id, firstname, lastname, email, leader)
VALUES (99996, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTINVALID@', 99996);

INSERT INTO staff(staff_id, firstname, lastname, email, leader)
VALUES (99997, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTINVALID.', 99997);

UPDATE staff SET email = 'UPDATEINVALIDEMAIL'
WHERE staff_id = 99999;
--Expected outcomes for each insert or update attempt: Error message shown saying invalid email and row not inserted or updated in the staff table

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

--2 Does not fire where not needed
INSERT INTO staff(staff_id, firstname, lastname, email, leader)
```

```
VALUES (99998, 'TEST FIRSTNAME', 'TEST LASTNAME', 'TESTVALID@VALID.COM', 99998);

UPDATE staff SET email = 'ANOTHERVALID@VALID.COM'
WHERE staff_id = 99998;

--Expected outcomes: The row is inserted and updated succesfully

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

--3 Changes are made if it is supposed to
--Trigger should not allow invalid data to be inserted
SELECT staff_id, firstname, lastname, email FROM staff;

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

--4 Changes are not made if it is not supposed to
--Trigger should allow valid data to be inserted
SELECT staff_id, firstname, lastname, email FROM staff;

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

```
--RESET

DELETE FROM staff WHERE staff_id IN(99995, 99996, 99997, 99998, 99999);


--Test reset

SELECT staff_id, firstname, lastname, email FROM staff;

--Expected outcome: The dummy data inserted for testing trigger are removed



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

---------------------- TEST trig_staff_name_ck


--This trigger is supposed to prevent adding a staff with invalid name (with numbers)


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


--First Insert a valid row

INSERT INTO staff(staff_id, firstname, lastname, leader)

VALUES (99999, 'TEST FIRSTNAME', 'TEST LASTNAME', 99999);


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

```
--1. Fires where needed

INSERT INTO staff(staff_id, firstname, lastname, leader)

VALUES (99995, 'TEST01', 'TEST LASTNAME12', 99995);


INSERT INTO staff(staff_id, firstname, lastname, leader)

VALUES (99996, 'TEST', '12LASTNAME', 99996);


INSERT INTO staff(staff_id, firstname, lastname, leader)

VALUES (99997, 'TEST9TE', 'TEST LASTNAME', 99997);


UPDATE staff SET firstname = '12NAME'

WHERE staff_id = 99999;


UPDATE staff SET lastname = 'NLAST000'

WHERE staff_id = 99999;


-- Expected outcomes for each insert or update attempt: Error message shown saying invalid firstname or lastname

-- and row is not inserted or updated in the staff table


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

```
--2. Does not fire where not needed
INSERT INTO staff(staff_id, firstname, lastname, leader)
VALUES (99998, 'TEST FIRSTNAME', 'TEST LASTNAME', 99998);

UPDATE staff SET firstname = 'VALID FIRST', lastname = 'VALID LAST'
WHERE staff_id = 99998;

--Expected outcomes: The row is inserted and updated succesfully, also messages showing VALID FIRSTNAME and VALID LASTNAME

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

--3. Changes are made if it is supposed to
--Trigger should not allow invalid data to be inserted
SELECT staff_id, firstname, lastname FROM staff;

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

--4. Changes are not made if it is not supposed to
--Trigger should allow valid data to be inserted
SELECT staff_id, firstname, lastname FROM staff;
```

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


--RESET
DELETE FROM staff WHERE staff_id IN(99995, 99996, 99997, 99998, 99999);


--Test reset
SELECT staff_id, firstname, lastname FROM staff;
--Expected outcome: The dummy data inserted for testing trigger are removed




-----------------------------------------------------------------------------------------------------------------------------
---------------------- TEST trig_del_staff


--This trigger is supposed to notify deletion of staff


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


--Insert dummy valid data into staff
INSERT INTO staff(staff_id, firstname, lastname, leader)
VALUES (99998, 'TEST FIRSTNAME', 'TEST LASTNAME', 99998);
```

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

--Insert dummy data into locations

INSERT INTO locations(location_id, price, capacity)

VALUES (99999, 15000, 120);

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

--1 Fires where needed

DELETE FROM staff WHERE staff_id = 99998;

--Expected outcome: A message is displayed saying you deleted a staff along with the name of the staff

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

--2 Does not fire where not needed

DELETE FROM locations WHERE location_id = 99999;

--Expected outcome: No extra message is displayed

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

```
--3 Changes are made if it is supposed to

-- Makes no changes


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


--4 Changes are not made if it is not supposed to

SELECT staff_id, firstname, lastname FROM staff;

--Expected outcome: The dummy data inserted for testing trigger are removed




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

---------------------- TEST trig_ck_location


--This trigger is supposed to prevent negative values from being inserted as price or capacity for a location


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


--First enter a valid row

INSERT INTO locations(location_id, price, capacity)

VALUES (99996, 5555, 111);
```

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

--1 Fires where needed

INSERT INTO locations(location_id, price, capacity)

VALUES (99997, -5000, 120);


INSERT INTO locations(location_id, price, capacity)

VALUES (99998, 5000, -120);


UPDATE locations SET price = -200

WHERE location_id = 99996;


-- Expected outcomes for each insert or update attempt: Error message shown saying cannot have values less than zero and the row is not
inserted

-- into the locations table or not updated


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


--2 Does not fire where not needed

INSERT INTO locations(location_id, price, capacity)
```

```
VALUES (99999, 5000, 120);

--Expected outcome: No error message is displayed and the row is inserted into the locations table

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

--3 Changes are made if it is supposed to

--Trigger should not allow invalid data to be inserted

SELECT location_id, price, capacity FROM locations;

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

--4 Changes are not made if it is not supposed to

--Trigger should allow valid data to be inserted

SELECT location_id, price, capacity FROM locations;

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

--RESET

DELETE FROM locations WHERE location_id IN(99999, 99998, 99997, 99996);

--Test reset
```

```
SELECT location_id, price, capacity FROM locations;

--Expected outcome: The dummy data inserted for testing trigger are removed



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

---------------------- TEST trig_festival_natures


--This trigger is supposed to notify any insert, update or delete performed on festival_natures table


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


--1 Fires where needed

--Insert

          INSERT INTO festival_natures

          VALUES (99999, 'TEST NAME', 'TEST AUDIENCE');

--Update

          UPDATE festival_natures SET name = 'UPDATED NAME' WHERE festival_nature_id = 99999;

--Delete

          DELETE FROM festival_natures WHERE festival_nature_id = 99999;

--Expected outcome: For each insert, update or delete, a relevant message is shown about the performed action
```

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


--2 Does not fire where not needed

--Insert

        INSERT INTO locations(location_id, price, capacity)

        VALUES (99999, 5000, 120);

--Update

        UPDATE locations SET price = 9999 WHERE location_id = 99999;

--Delete

        DELETE FROM locations WHERE location_id = 99999;

--Expected outcome: No message is displayed


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


--3 Changes are made if it is supposed to

        SELECT * FROM festival_natures;

        SELECT location_id, price, capacity FROM locations;

--The attempted tasks are performed and the trigger does not affect the data


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

--4 Changes are not made if it is not supposed to

```
        SELECT * FROM festival_natures;

        SELECT location_id, price, capacity FROM locations;
```

--The attempted tasks are performed and the trigger does not affect the data

----

--RESET

```
        DELETE FROM festival_natures WHERE festival_nature_id = 99999;

        DELETE FROM locations WHERE location_id = 99999;
```

--Test Reset

```
        SELECT * FROM festival_natures;

        SELECT location_id, price, capacity FROM locations;
```

--Expected outcome: The dummy data inserted for testing trigger are removed

----

---------------------- TEST trig_festival_staff

--This trigger is supposed to notify any insert, update or delete performed on festival_staff table

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

--1 Fires where needed
--Insert
        INSERT INTO festival_staff
        VALUES (99999, 1, 1, 1);
--Update
        UPDATE festival_staff SET staff_id = 2 WHERE festival_staff_id = 99999;
--Delete
        DELETE FROM festival_staff WHERE festival_staff_id = 99999;
--Expected outcome: For each insert, update or delete, a relevant message is shown about the performed action

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

--2 Does not fire where not needed
--Insert
        INSERT INTO locations(location_id, price, capacity)
        VALUES (99999, 5000, 120);
--Update
        UPDATE locations SET price = 9999 WHERE location_id = 99999;
```

```
--Delete
            DELETE FROM locations WHERE location_id = 99999;
--Expected outcome: No message is displayed

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

--3 Changes are made if it is supposed to
            SELECT * FROM festival_staff;
            SELECT location_id, price, capacity FROM locations;
--The attempted tasks are performed and the trigger does not affect the data

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

--4 Changes are not made if it is not supposed to
            SELECT * FROM festival_staff;
            SELECT location_id, price, capacity FROM locations;
--The attempted tasks are performed and the trigger does not affect the data

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

--RESET
```

```
        DELETE FROM festival_staff WHERE festival_staff_id = 99999;

        DELETE FROM locations WHERE location_id = 99999;


--Test Reset

        SELECT * FROM festival_staff;

        SELECT location_id, price, capacity FROM locations;

--Expected outcome: The dummy data inserted for testing trigger are removed



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

12. Cursor_4.txt

```
/*



        CREATE CURSOR FILE

                -Contains commands for creating the cursors

                -Contains both implicit and explicit types of cursors

                -Contains all the Cursor Attributes
        GROUP 4

                18406547 - Diwas Lamsal

                18406498 - Bhuwan Khatiwada

                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal

*/



--@D:\Database_Assignment_II\script\cursor_4.txt



--Display dbms output

SET SERVEROUTPUT ON



---------------------------------------------------------- CURSORS ------------------------------------------------------

/*Implicit Cursor
```

```
    -This procedure uses implicit cursor while deleting an address and displays which address was deleted.

    -If address not found, it displays the message 'NO SUCH ADDRESS WAS FOUND'


    Parameter in_street: Take the street of the address to be deleted


*/

CREATE OR REPLACE PROCEDURE proc_del_address_cursor(in_street VARCHAR2) IS

BEGIN

    DELETE FROM addresses WHERE street = in_street;

    IF SQL%FOUND THEN

        DBMS_OUTPUT.PUT_LINE('ADDRESS WITH STREET "'||in_street || '" DELETED SUCCESSFULLY!');

    ELSE

        DBMS_OUTPUT.PUT_LINE('NO SUCH ADDRESS WAS FOUND!');

    END IF;

END proc_del_address_cursor;

/

SHOW ERRORS

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


/*Implicit Cursor
```

```
    -This procedure uses implicit cursor while updating all the locations with price below provided price

    -It also displays the number of found and updated locations

    -If no location was found, it displays a message saying so


    Parameter in_price: Take the maximum price for location


*/

CREATE OR REPLACE PROCEDURE proc_num_locations_price(in_price NUMBER) IS

    vn_location_id locations.location_id%TYPE;

BEGIN

    UPDATE locations SET price = price + 50 WHERE price <= in_price;

    IF SQL%NOTFOUND THEN

        DBMS_OUTPUT.PUT_LINE('LOCATION WITH PRICE LESS THAN "'|| in_price || '" NOT FOUND!');

    ELSE

        DBMS_OUTPUT.PUT_LINE('THERE ARE '||SQL%ROWCOUNT||' LOCATIONS WITH PRICE LESS THAN '||in_price|| ', and they have been updated.');

    END IF;

END proc_num_locations_price;

/

SHOW ERRORS
```

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

/*Explicit Cursor, Use of OUT parameter, Use of FOR loop

        -This procedure uses explicit cursor while displaying all the locations
        -It also displays the number of locations found

        Parameter out_num_records: This parameter returns the total number of found rows(locations)

*/
CREATE OR REPLACE PROCEDURE proc_view_addresses_cursor(out_num_records OUT NUMBER) AS
        CURSOR cur_name IS
        SELECT location_id, l.address.street AS street, l.address.city AS city, l.address.country AS country, capacity, price
        FROM locations l;
        vn_count NUMBER(3):=0;
BEGIN
        FOR rec_cur_names IN cur_name LOOP
                DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');
```

```
                    DBMS_OUTPUT.PUT_LINE(cur_name%ROWCOUNT || '. The location costs '|| rec_cur_names.price || '. The address of the
location is');

                    DBMS_OUTPUT.PUT_LINE(rec_cur_names.street || ', '|| rec_cur_names.city || ', '|| rec_cur_names.country || '. The capacity is
'||rec_cur_names.capacity);

                    DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------');

                    vn_count := cur_name%ROWCOUNT;

          END LOOP;

          out_num_records:= vn_count;


END proc_view_addresses_cursor;

/

SHOW ERRORS


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


/*Explicit Cursor, Use of OUT parameter, Use of WHILE loop


          -This procedure uses explicit cursor while displaying all the festival_natures

          -It also displays the number of festival_natures found


          Parameter out_num_records: This parameter returns the total number of found rows(festival_natures)
```

```
*/
CREATE OR REPLACE PROCEDURE proc_view_festival_natures_cursor(out_num_records OUT NUMBER) AS
        CURSOR cur_name IS
        SELECT festival_nature_id, name, target_audience
        FROM festival_natures;
        --Need to declare when using while
        rec_cur_names cur_name%ROWTYPE;
BEGIN
        OPEN cur_name;
        FETCH cur_name INTO rec_cur_names;
        IF cur_name%NOTFOUND THEN
                DBMS_OUTPUT.PUT_LINE('NO FESTIVAL NATURES WERE FOUND IN THE SYSTEM!');
                out_num_records:=0;
        END IF;
        WHILE cur_name%FOUND LOOP
                IF cur_name%ISOPEN THEN
                        DBMS_OUTPUT.PUT_LINE('THE CURSOR IS OPEN');
                END IF;

                DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');
```

```
                    DBMS_OUTPUT.PUT_LINE(cur_name%ROWCOUNT || ' The festival nature '|| rec_cur_names.name || ' targets the audiences '
|| rec_cur_names.target_audience);

                    DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------');

                    out_num_records:=cur_name%ROWCOUNT;

                    FETCH cur_name INTO rec_cur_names;

          END LOOP;

          CLOSE cur_name;


          IF NOT cur_name%ISOPEN THEN

                    DBMS_OUTPUT.PUT_LINE('THE CURSOR IS NOW CLOSED');

          END IF;


END proc_view_festival_natures_cursor;

/

SHOW ERRORS


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


/*Explicit Cursor, Use of FOR loop


          -This procedure uses explicit cursor while displaying all the staff contact numbers along with some other details
```

```
*/
CREATE OR REPLACE PROCEDURE proc_view_staff_contact_cursor AS
        CURSOR cur_name IS
        SELECT staff_id, firstname, lastname, c.contact_number AS cnumber, c.number_type AS ctype, email, salary
        FROM staff s, TABLE (s.contact) c ORDER BY staff_id;
        vn_count NUMBER(3):=0;
BEGIN
        FOR rec_staff IN cur_name LOOP
                DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');
                DBMS_OUTPUT.PUT_LINE(rec_staff.staff_id || '. Contact details for the staff: '|| rec_staff.firstname || ' ' || rec_staff.lastname);
                DBMS_OUTPUT.PUT_LINE(' Contact Number: '|| rec_staff.cnumber || ', Contact Type: '|| rec_staff.ctype || ', Email: 
'||rec_staff.email);
                DBMS_OUTPUT.PUT_LINE(' The salary of the staff is: ' || rec_staff.salary);
                DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');
                vn_count := cur_name%ROWCOUNT;
        END LOOP;
        IF vn_count = 0 THEN
                DBMS_OUTPUT.PUT_LINE('NO RECORDS WERE FOUND!');
        END IF;
END proc_view_staff_contact_cursor;
```

```
/
SHOW ERRORS

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

/*Explicit Cursor, Use of OUT parameter, Use of no loop

	-This procedure uses explicit cursor while displaying weekly salary of the provided staff
	-It uses functions such as ROUND, TRUNC, FLOOR and CEIL

	Parameter in_staff_id: The id of the staff whose salary is to be displayed

*/
--View staff weekly salary
CREATE OR REPLACE PROCEDURE proc_view_staff_salary_weekly_cursor(in_staff_id NUMBER) AS
	CURSOR cur_name IS
	SELECT staff_id, firstname, lastname, salary
	FROM staff
	WHERE staff_id = in_staff_id;
	rec_cur_names cur_name%ROWTYPE;
```

```
    --Variable for displaying salary

    vn_weekly_salary NUMBER(10,2);

BEGIN

    OPEN cur_name;

    FETCH cur_name INTO rec_cur_names;

    IF cur_name%NOTFOUND THEN

        DBMS_OUTPUT.PUT_LINE('NO STAFF WAS FOUND WITH THE PROVIDED ID!');

    ELSE

        vn_weekly_salary := (rec_cur_names.salary*7)/(30); -- Assuming 30 days in a month


        DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');

        DBMS_OUTPUT.PUT_LINE('Staff id       :  '|| rec_cur_names.staff_id);

        DBMS_OUTPUT.PUT_LINE('Name           :  '|| rec_cur_names.firstname || ' ' || rec_cur_names.lastname);

        DBMS_OUTPUT.PUT_LINE('Monthly Salary  :  '|| rec_cur_names.salary);

        DBMS_OUTPUT.PUT_LINE('Weekly Salary   :  '|| vn_weekly_salary);

        DBMS_OUTPUT.PUT_LINE('Rounded Weekly  :  '|| ROUND(vn_weekly_salary, -1) || ' - to tens place'); --Round

        DBMS_OUTPUT.PUT_LINE('Trunc Weekly    :  '|| TRUNC(vn_weekly_salary, -1) || ' - to tens place'); --Trunc

        DBMS_OUTPUT.PUT_LINE('Floored Weekly  :  '|| CEIL(vn_weekly_salary)); -- Ceil

        DBMS_OUTPUT.PUT_LINE('Ceiled Weekly   :  '|| FLOOR(vn_weekly_salary)); -- Floor

        DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');
```

```
                FETCH cur_name INTO rec_cur_names;
        END IF;
        CLOSE cur_name;
END proc_view_staff_salary_weekly_cursor;
/
SHOW ERRORS


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

13. Insert_4.txt

```
/*

        INSERT FILE
                -Contains commands for inserting data to the tables
                -Shows usage of procedures for inserting

        GROUP 4

                18406547 - Diwas Lamsal
```

```
                    18406498 - Bhuwan Khatiwada

                    18413710 - Sandip Kumar Subba

                    18413644 - Bishownath Dhakal


*/


--@D:\Database_Assignment_II\script\insert_4.txt


--Display dbms output

SET SERVEROUTPUT ON



------------------------------PLEASE CREATE ALL THE PROCEDURES FOR INSERTING BEFORE USING THEM HERE--------------------------------


--Some dummy data have been taken from the tutorials



-------------------- ADDRESSES OBJECT TABLE ------------------


 --Format the table

COLUMN street FORMAT A30;
COLUMN city FORMAT A20;

COLUMN country FORMAT A20;


--Verify empty table

SELECT street, city, country FROM addresses;


--Insert without procedure

INSERT INTO addresses(street, city, country)

VALUES ('54 FESTIVE ROAD', 'NORTHAMPTON', 'UK');


--Insert using procedure

EXEC proc_insert_addresses('111 BAY STATE ROAD', 'BOSTON', 'USA');

EXEC proc_insert_addresses('111 VALLEY WAY', 'SAN FERNANDO', 'USA');

EXEC proc_insert_addresses('4545 ORACLE DRIVE', 'SAN FRANCISCO', 'USA');

EXEC proc_insert_addresses('77 ELM STREET', 'NEWARK', 'USA');

EXEC proc_insert_addresses('544 42ND STREET', 'NEW YORK', 'USA');

EXEC proc_insert_addresses('111 BEACHCOMBER PLACE', 'OCEANSIDE', 'USA');

EXEC proc_insert_addresses('25 MISSION WAY', 'SAN DIEGO', 'USA');

EXEC proc_insert_addresses('177 AIRPORT ROAD', 'NEWARK', 'USA');

EXEC proc_insert_addresses('SHREE ADARSHA MARG', 'KATHMANDU', 'NEPAL');
```

```
--Verify insert
SELECT street, city, country FROM addresses;

--Expected : The inserted data appears in the table
--Result   : The inserted data appears in the table

--------------------- STAFF TABLE ------------------

--Verify empty table
SELECT staff_id, firstname, lastname, leader FROM staff;

--Requires address_type and contact_detail_varray_type

--Insert without procedure
INSERT INTO staff (staff_id, firstname, lastname, gender, contact, current_address, permanent_address, email, leader, salary)
VALUES (seq_staff_id.NEXTVAL, 'RAMESH', 'THAPA', 'M', contact_detail_varray_type(contact_detail_type('9808123457', 'MOBILE'),

               contact_detail_type('01-4216354', 'LANDLINE'))
                                                                                                                                    ,
address_type('7500 IMPERIAL BLVD', 'LOS ANGELES', 'USA')
                                                                                                                                    ,
address_type('333 SAN JUAN BLVD', 'SAN JUAN', 'PUERTO RICO')
```

,

```
'RAMESHTHAPA@GMAIL.COM', seq_staff_id.CURRVAL, 12990);



--Insert with procedure

EXEC proc_insert_staff('DIANE', 'BROWN', 'F', '(617)342-23442', 'LANDLINE', '981513244', 'MOBILE', '4242 MISTY LANE', 'SEATTLE', 'USA', 'KUNGSGATAN 56', 'STOCKHOLM', 'SWEDEN', 'DIANE_BROWN@HOTMAIL.COM', 1, 5000);

EXEC proc_insert_staff('BAN', 'FORHAN', 'M', '98080421345', 'NCELL', '9841354657', 'NTC', '42 SALTY LAKE', 'WASHINGTON', 'RUSSIA', '42 SALTY LAKE', 'WASHINGTON', 'RUSSIA', 'FORHANBAN@GMAIL.COM', (seq_staff_id.CURRVAL+1), 2000);

EXEC proc_insert_staff('KRUNAL', 'UPRETI', 'F', '(217)342-21375', 'LANDLINE', '984132787', 'MOBILE', '12 SHREE ADARSHA MARG', '3', 'NEPAL', 'DASHRATH MARGA', '3', 'NEPAL', 'KRUNALUPRETI@YAHOO.COM', 1, 20000);

EXEC proc_insert_staff('KEVIN', 'COX', 'M', '(619)433-6845', 'LANDLINE', '9153421542', 'MOBILE', '5567 KNIGHTSBRIDGE COURT', 'LONDON', 'UK', '5567 KNIGHTSBRIDGE COURT', 'LONDON', 'UK', 'KEVINCOX@CMP.CO.UK', 1, 3155.55);

EXEC proc_insert_staff('RANJAN', 'KHANAL', 'M', '(124)448-4214', 'LANDLINE', '9842142144', 'MOBILE', '123 CHUCCHWAL TOLE', 'KATHAMNDU', 'NEPAL', '10 SANKHAMUL', 'KATHAMNDU', 'NEPAL', 'RANJAN@TEST.COM', 1, 4510.55);

EXEC proc_insert_staff('ARUNA', 'TIWARI', 'F', '(142)485-4231', 'LANDLINE', '9841144242', 'MOBILE', '15 VICTORY ROAD', 'HO CHI MIN', 'VIETNAM', '15 VICTORY ROAD', 'HO CHI MIN', 'VIETNAM', 'ARUNAT@GMAIL.COM', 3, 5510.55);


--Verify insert

SELECT staff_id, firstname, lastname, leader FROM staff;


--Expected : The inserted data appears in the table

--Result   : The inserted data appears in the table
```

```
--------------------- FESTIVAL_NATURES TABLE ------------------


--Verify empty table

SELECT * FROM festival_natures;


--Insert without procedure

INSERT INTO festival_natures (festival_nature_id,name,target_audience)

VALUES (seq_festival_nature_id.NEXTVAL, 'PURAN', 'OLD');


--Insert using procedure

EXEC proc_insert_festival_natures('CONCERT', 'YOUTH');

EXEC proc_insert_festival_natures('CONCERT', 'OLD');

EXEC proc_insert_festival_natures('DANCE', 'YOUTH');

EXEC proc_insert_festival_natures('KIDS FESTIVAL', 'KIDS');



--Verify insert

SELECT * FROM festival_natures;


--Expected : The inserted data appears in the table
```

```
--Result   : The inserted data appears in the table



--------------------- LOCATIONS TABLE ------------------


--Verify empty table

SELECT location_id, capacity, price, l.address.street, l.address.city, l.address.country FROM locations l;


--Requires reference of addresses table


--Insert without procedure

INSERT INTO locations (location_id,capacity,price,address)

SELECT seq_location_id.NEXTVAL, 1500, 10000, REF(a)

FROM addresses a

WHERE a.street = '111 VALLEY WAY';


--Insert with procedure

EXEC proc_insert_locations(1200, 20500, '177 AIRPORT ROAD');

EXEC proc_insert_locations(500, 5500, '25 MISSION WAY');

EXEC proc_insert_locations(750, 2500, '4545 ORACLE DRIVE');

EXEC proc_insert_locations(200, 1000, '544 42ND STREET');
```

```
--Verify insert

SELECT location_id, capacity, price, l.address.street AS street, l.address.city  AS city, l.address.country  AS country FROM locations l ORDER BY
location_id;


--Expected : The inserted data appears in the table

--Result   : The inserted data appears in the table




------------------------ FESTIVALS TABLE ---------------------


--Verify empty table

SELECT festival_nature_id, location_id, festival_name FROM festivals;


--Insert without procedure

INSERT INTO festivals (festival_nature_id, location_id, festival_name, performance)

VALUES (1, 1, 'BHAGAVATA', performance_table_type(performance_type('PRAWACHAN', 'GURU ARBINDRA NATH', 'RELIGIOUS'),

                                                                                                performance_type('PRAWACHAN', 'GURU
ANUBHAVAM ACHARYA', 'RELIGIOUS')));
```

```
--Insert with procedure

--Format of Arguments: Festival Nature Id, Location Id, Festival name, Performance1 - name, artist, genre, Performance 2- name, artist, genre

EXEC proc_insert_festivals(2, 3, 'SCREAM FEST', 'SINGING', 'NEPATHYA', 'ROCK', 'MUSIC', 'ANTIM GRAHAN', 'METAL');

EXEC proc_insert_festivals(5, 1, 'PHOENIX FESTIVAL', 'QUIZ', 'PE ENTERTAINERS', 'KIDS', 'GAME', 'PE ENTERTAINERS', 'KIDS');

EXEC proc_insert_festivals(3, 2, 'MUSIC FEST', 'METAL PERFORMANCE', 'METALLICA', 'METAL', 'ROCK PERFORMANCE', 'BLINK 182', 'PUNK ROCK');

EXEC proc_insert_festivals(4, 3, 'DANCE FEST', 'SALSA', 'CARTOONZ CREW', 'LINDY HOP', 'SWING DANCE', 'CARTOONZ', 'SWING');



--Verify insert

SELECT festival_nature_id, location_id, festival_name FROM festivals;


--Expected : The inserted data appears in the table

--Result   : The inserted data appears in the table



--------------------- FESTIVAL_STAFF TABLE ------------------



--Verify empty table

SELECT festival_staff_id, festival_nature_id, location_id, staff_id FROM festival_staff;
```

```
--Insert without procedure

INSERT INTO festival_staff (festival_staff_id,staff_id,festival_nature_id,location_id)

VALUES (seq_festival_staff_id.NEXTVAL, 1, 2, 3);


--Insert with procedure

EXEC proc_insert_festival_staff(3, 5, 1);

EXEC proc_insert_festival_staff(2, 3, 2);

EXEC proc_insert_festival_staff(1, 5, 1);

EXEC proc_insert_festival_staff(7, 4, 3);


--Verify insert

SELECT festival_staff_id, festival_nature_id, location_id, staff_id FROM festival_staff;


--Expected : The inserted data appears in the table

--Result   : The inserted data appears in the table
```

14. Update_4.txt

```
/*


        UPDATE FILE
```

```
		-Contains commands for updating existing data in tables

	GROUP 4

		18406547 - Diwas Lamsal

		18406498 - Bhuwan Khatiwada

		18413710 - Sandip Kumar Subba

		18413644 - Bishownath Dhakal

*/


--@D:\Database_Assignment_II\script\update_4.txt

--------------------------------------THIS FILE CONTAINS UPDATES--------------------------------------

--Display dbms output
SET SERVEROUTPUT ON


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

```
--Update addresses object table

UPDATE addresses a SET a.city = 'NEW BANESHWOR' WHERE a.street = 'SHREE ADARSHA MARG';

--Query

SELECT * FROM addresses WHERE street = 'SHREE ADARSHA MARG';


--Change it back

UPDATE addresses a SET a.city = 'KATHMANDU' WHERE a.street = 'SHREE ADARSHA MARG';

--Query

SELECT * FROM addresses WHERE street = 'SHREE ADARSHA MARG';


--Expected outcome: City for the provided street SHREE ADARSHA MARG is first changed to NEW BANESHWOR and then back to KATHMANDU


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

--Update staff table


--Update firstname and lastname

UPDATE staff SET firstname = 'SURESH', lastname = 'BHUJEL'

WHERE staff_id = 1;


--Query
```

```
SELECT staff_id, firstname, lastname FROM staff WHERE staff_id = 1;

--Expected outcome: Firstname and lastname for staff id 1 are changed to SURESH BHUJEL

--Update contact detail
UPDATE staff s
SET s.contact = contact_detail_varray_type(contact_detail_type('123-451-523', 'TELEPHONE'),
        contact_detail_type('9811541542', 'MOBILE'))
WHERE s.staff_id = 3;

--Query
SELECT contact FROM staff WHERE staff_id = 3;

--Expected outcome: Contact details are changed for the staff id 3

--Update temporary address
UPDATE staff s
SET s.current_address = address_type('12 DELHI ROAD', 'DELHI', 'INDIA')
WHERE s.staff_id = 2;

--Query
```

```
SELECT current_address FROM staff WHERE staff_id = 2;

--Expected outcome: Current addrress of the staff is changed to 12 DELHI ROAD, DELHI, INDIA

--Increase staff salary
EXEC proc_increase_salary(2, 11.5);
EXEC proc_increase_salary(1, 15);

--Query
SELECT staff_id, salary FROM staff;

--Expected outcome: Staff salary are raised by 11.5 and 15 percent for staff ids 2 and 1 respectively

------------------------------------------------------------------------------------------------
--Update locations table

--Update address ref and capacity together
UPDATE locations l
SET l.address =(
            SELECT REF(a)
            FROM addresses a
```

```
        WHERE a.street = 'SHREE ADARSHA MARG'), capacity = 1250
WHERE l.location_id = 2;


--Query
SELECT location_id, l.address.street AS street, l.address.city AS city, l.address.country AS country, capacity FROM locations l WHERE l.location_id
= 2;


--Expected outcome: address is updated to shree adarsha marg and capacity is changed to 1250 for location id 2


--Discount location
EXEC proc_discount_location(2, 25);


--Query
SELECT location_id, price FROM location;


--Expected outcome: Price is reduced by 25 percent in location id 2


------------------------------------------------------------------------------------------------
--Update festival_natures table


--Update name and target audience
```

```
UPDATE festival_natures

SET name = 'OLD CONCERT', target_audience = 'AGE 50+'

WHERE festival_nature_id = 3;


--Query

SELECT * FROM festival_natures WHERE festival_nature_id = 3;


--Expected outcome: name and target audience of the festival nature is changed to OLD CONCERT and AGE 50+ respectively


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

--Update festivals table


--Domscheit, W. (2014)

--Update performance nested table

UPDATE TABLE(SELECT f.performance

            FROM   festivals f

            WHERE  f.festival_nature_id = 3 AND f.location_id = 2) p

  SET p.name = 'METAL'

           WHERE p.artist = 'METALLICA';


--Query
```

```
SELECT performance FROM festivals f
WHERE  f.festival_nature_id = 3 AND f.location_id = 2;

--Expected outcome: name of the performance is changed to METAL for artist metallica

----------------------------------------------------------------------------------------------
--Update festival_staff table

--Update the staff associated with festival_staff
UPDATE festival_staff SET staff_id = 2
WHERE festival_staff_id = 5;

--Query
SELECT * FROM festival_staff WHERE festival_staff_id = 5;

--Expected outcome: Staff id is 2  for festival_staff_id 5

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

15. Query_4.txt

```
/*

        QUERY FILE

                -Contains the commands to query the data added to the database

                -Includes usage of procedures and cursors

                -Shows usage of out parameter

                -Shows usage of almost all the features from Term I for example: nested queries, UNION, LEFT JOIN, MIN, AVG etc.


        GROUP 4


                18406547 - Diwas Lamsal

                18406498 - Bhuwan Khatiwada

                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal


*/



--@D:\Database_Assignment_II\script\query_4.txt


---------------------------------------THIS FILE CONTAINS QUERIES---------------------------------------
```

```
--Set linesize to display all the columns in a nice format without title line breaks

SET linesize 125


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

--Query log details

COLUMN event_date FORMAT A20;

SELECT * FROM login_details;


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


--------------------- ADDRESSES OBJECT TABLE ------------------


--Format the columns

COLUMN street FORMAT A30;

COLUMN city FORMAT A20;

COLUMN country FORMAT A20;


--Query whole table

--INITCAP makes the text appear in Camel Case

SELECT INITCAP(street) AS street, city, country FROM addresses;
```

```
--Other queries

--Querying addresses where no locations exist or no staff live
--Usage of MINUS and UNION
--Help taken from tutorial week 10
SELECT a.city, a.street, a.country FROM addresses a
MINUS(
SELECT s.current_address.city, s.current_address.street, s.current_address.country FROM staff s
UNION
SELECT ss.permanent_address.city, ss.permanent_address.street, ss.permanent_address.country FROM staff ss
UNION
SELECT l.address.city, l.address.street, l.address.country FROM locations l);
--As expected, the addresses that exist in the locations are only removed as locations used in staff table are object types and unique to
--data in the addresses table.

--Querying addresses where locations exist
--Usage of INTERSECT
SELECT a.city, a.street, a.country FROM addresses a
INTERSECT
SELECT l.address.city, l.address.street, l.address.country FROM locations l;
```

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

--------------------- STAFF TABLE ------------------

--Query the whole table

--Format the columns
COLUMN firstname FORMAT A10
COLUMN lastname FORMAT A10
COLUMN contact_number FORMAT A15
COLUMN number_type FORMAT A12
COLUMN current_street FORMAT A15 WOR
COLUMN current_city FORMAT A15 WOR
COLUMN current_country FORMAT A15 WOR
COLUMN permanent_street FORMAT A16 WOR
COLUMN permanent_city FORMAT A15 WOR
COLUMN permanent_country FORMAT A18 WOR


--Query with current address
```

```
SELECT staff_id, firstname, lastname, gender, c.contact_number, c.number_type, s.current_address.street AS current_street,
s.current_address.city AS current_city, s.current_address.country AS current_country

FROM staff s, TABLE (s.contact) c ORDER BY staff_id;


--Query with permanent address

SELECT staff_id, firstname, lastname, gender, c.contact_number, c.number_type, s.permanent_address.street AS permanent_street,
s.permanent_address.city AS permanent_city, s.permanent_address.country AS permanent_country

FROM staff s, TABLE (s.contact) c ORDER BY staff_id;



--Other queries


--Staff with leader names

--Usage of INNER JOIN

SELECT s.staff_id, s.firstname, s.lastname, l.staff_id AS leader_id, l.firstname AS leader_firstname, l.lastname AS leader_lastname

FROM staff s JOIN staff l ON

s.leader = l.staff_id

ORDER BY s.staff_id;



--Staff having higher salary than leaders using subquery
```

```
SELECT s.firstname, s.lastname, s.salary
FROM staff s
WHERE EXISTS(
        SELECT l.staff_id FROM staff l
        WHERE s.leader = l.staff_id
        AND s.salary>l.salary);


--Average salary of staff
SELECT AVG(salary) FROM staff;

--Highest salary of staff
SELECT MAX(salary) FROM staff;

--Staff having highest salary
SELECT s.staff_id, s.firstname, s.lastname, s.salary
FROM staff s
WHERE s.salary IN(
        SELECT MAX(salary) FROM staff);
```

```
--Lowest salary of staff
SELECT MIN(salary) FROM staff;


--Staff having lowest salary
SELECT s.staff_id, s.firstname, s.lastname, s.salary
FROM staff s
WHERE s.salary IN(
        SELECT MIN(salary) FROM staff);


--Rest of the staff (Staff not having lowest salary)
SELECT s.staff_id, s.firstname, s.lastname, s.salary
FROM staff s
WHERE s.salary NOT IN(
        SELECT MIN(salary) FROM staff);



--Total budget of company allocated in staff salary
SELECT SUM(salary) FROM staff;


--Number of staff
SELECT COUNT(staff_id) FROM staff;
```

```
--Staff having leaders

SELECT s.staff_id, s.firstname, s.lastname

FROM staff s

WHERE s.leader != s.staff_id;


--Staff having no leaders

SELECT s.staff_id, s.firstname, s.lastname

FROM staff s

WHERE s.leader = s.staff_id;



--Query using procedure


--Get username and password

EXEC proc_username_password(2);

EXEC proc_username_password(&staff_id);



--Total number of staff

EXEC proc_count_staff;
```

```
--Total number of staff with salary more than provided number
EXEC proc_count_staff_salary(10000);
EXEC proc_count_staff_salary(15000);
EXEC proc_count_staff_salary(&amount);

--Query staff details using id
EXEC proc_show_staff_address(1);
EXEC proc_show_staff_address(2);
EXEC proc_show_staff_address(3);
EXEC proc_show_staff_address(4);
EXEC proc_show_staff_address(5);
EXEC proc_show_staff_address(&staff_id);

--Query staff firstname and lastname
EXEC proc_staff_firstname(1);
EXEC proc_staff_surname(1);

EXEC proc_staff_firstname(3);
EXEC proc_staff_surname(3);
EXEC proc_staff_firstname(&staff_id);

EXEC proc_staff_surname(&staff_id);


--Query using cursor

--View staff contact details

EXEC proc_view_staff_contact_cursor;


--Query staff weekly salary using cursor

EXEC proc_view_staff_salary_weekly_cursor(1);

EXEC proc_view_staff_salary_weekly_cursor(3);

EXEC proc_view_staff_salary_weekly_cursor(4);

EXEC proc_view_staff_salary_weekly_cursor(&staff_id);



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


--------------------- FESTIVAL_NATURES TABLE ------------------


--Query the whole table
```

```
COLUMN name FORMAT a20
COLUMN target_audience FORMAT a15

SELECT * FROM festival_natures;

--Other queries

--Festival natures for old people
SELECT * FROM festival_natures WHERE target_audience = 'OLD';

--Cartesian product (NOT READABLE)
SELECT fn.name, fn.target_audience, l.capacity, l.price FROM festival_natures fn, locations l;

--Using cursor
--Usage of OUT parameter
VAR TOTAL_FESTIVAL_NATURES NUMBER;
EXEC proc_view_festival_natures_cursor(:TOTAL_FESTIVAL_NATURES);
PRINT TOTAL_FESTIVAL_NATURES;

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

```
--------------------- LOCATIONS TABLE ------------------

--Query the whole table

COLUMN street FORMAT A15 WOR
COLUMN city FORMAT A15 WOR
COLUMN country FORMAT A15 WOR

SELECT location_id, capacity, price, l.address.street AS street, l.address.city  AS city, l.address.country  AS country FROM locations l ORDER BY location_id;

--Other queries



--Most Expensive Location
SELECT l.location_id, l.address.street AS street, l.address.city  AS city, l.address.country  AS country, l.capacity, l.price
FROM locations l
WHERE l.price IN(
        SELECT MAX(price) FROM locations);

--Cheapest Location
```

```
SELECT l.location_id, l.address.street AS street, l.address.city  AS city, l.address.country  AS country, l.capacity, l.price
FROM locations l
WHERE l.price IN(
            SELECT MIN(price) FROM locations);

--Location with highest capacity
SELECT l.location_id, l.address.street AS street, l.address.city  AS city, l.address.country  AS country, l.capacity, l.price
FROM locations l
WHERE l.capacity IN(
            SELECT MAX(capacity) FROM locations);

--Location with lowest capacity
SELECT l.location_id, l.address.street AS street, l.address.city  AS city, l.address.country  AS country, l.capacity, l.price
FROM locations l
WHERE l.capacity IN(
            SELECT MIN(capacity) FROM locations);

--Query using procedure

--Price of most expensive location
EXEC proc_exp_location;
```

```
--Price of cheapest location
EXEC proc_chp_location;



--Query the number of groups that can fit in a location
--The format:- First Argument: locaion_id, Second Argument: Group Size
EXEC proc_location_capacity_ck(3, 50);
EXEC proc_location_capacity_ck(5, 70);
EXEC proc_location_capacity_ck(2, 111);
EXEC proc_location_capacity_ck(2, 5);

--Query using cursor

--Variable used as out parameter to display number of records
VAR TOTAL_LOCATIONS NUMBER;
EXEC proc_view_addresses_cursor(:TOTAL_LOCATIONS);
PRINT TOTAL_LOCATIONS;



--Displays Number of locations below given price
```

```
--Also increases the location prices by 50

EXEC proc_num_locations_price(200);

EXEC proc_num_locations_price(20000);



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


------------------------ FESTIVALS TABLE ---------------------


--Query the whole table


COLUMN festival_name FORMAT A20 WOR

COLUMN performance_name FORMAT A20 WOR

COLUMN artist FORMAT A20 WOR

COLUMN genre FORMAT A15


SELECT festival_nature_id, location_id, festival_name, p.name AS performance_name, p.artist, p.genre

FROM festivals f, TABLE (f.performance) p;



--Other queries
```

```
--Festivals with location details and festival_nature details

--Formatting

COLUMN festival_name FORMAT A20 WOR

COLUMN festival_nature FORMAT A20 WOR

COLUMN target_audience FORMAT A15

COLUMN location FORMAT A30 WOR


--Query

SELECT f.festival_name AS festival_name, fn.name AS festival_nature, fn.target_audience, (CONCAT(CONCAT(CONCAT(CONCAT(l.address.street,
', '), l.address.city), ', '), l.address.country)) AS location, l.price

FROM festivals f JOIN festival_natures fn

ON f.festival_nature_id = fn.festival_nature_id

JOIN locations l

ON f.location_id = l.location_id;



--Get location id and total festivals involved in each location

--This query is displaying the location ids and number of festivals booked in that location

SELECT location_id, COUNT(festival_nature_id)

FROM festivals
```

```
GROUP BY location_id

HAVING COUNT(festival_nature_id)>1

ORDER BY COUNT(festival_nature_id) ASC;



--Query using procedure


--Total number of festivals

EXEC proc_count_festivals;



--Query festival details using festival name

EXEC proc_festival_detail('BHAGAVATA');

EXEC proc_festival_detail('SCREAM FEST');

EXEC proc_festival_detail('PHOENIX FESTIVAL');

EXEC proc_festival_detail('MUSIC FEST');

EXEC proc_festival_detail('DANCE FEST');

EXEC proc_festival_detail('&festival_name');
```

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

--------------------- FESTIVAL_STAFF TABLE ------------------

--Query the whole table

SELECT festival_staff_id, festival_nature_id, location_id, staff_id FROM festival_staff;



--Other queries

--Festival_staff with location details and festival_nature details along with staff details
--Formatting
COLUMN festival_nature FORMAT A20 WOR
COLUMN staff FORMAT A20 WOR
COLUMN location FORMAT A40 WOR

--Query
SELECT fs.festival_staff_id, CONCAT(CONCAT(s.firstname, ' '),  s.lastname) AS staff, fn.name AS festival_nature,
CONCAT(CONCAT(CONCAT(CONCAT(l.address.street, ', '), l.address.city), ', '), l.address.country) AS location
```

```
FROM festival_staff fs JOIN festivals f

ON fs.festival_nature_id = f.festival_nature_id

JOIN festival_natures fn

ON fn.festival_nature_id = f.festival_nature_id

JOIN locations l

ON f.location_id = l.location_id

JOIN staff s

ON s.staff_id = fs.staff_id

ORDER BY fs.festival_staff_id;



--Query using left join with staff

SELECT s.staff_id, CONCAT(CONCAT(s.firstname, ' '),  s.lastname) AS staff, fs.staff_id AS FS_staff_id, fs.festival_staff_id, fs.festival_nature_id,
fs.location_id

FROM staff s

LEFT JOIN festival_staff fs

ON fs.staff_id = s.staff_id;



--Same query using right join (Should display same result)
```

```
SELECT s.staff_id, CONCAT(CONCAT(s.firstname, ' '),  s.lastname) AS staff, fs.staff_id AS FS_staff_id, fs.festival_staff_id, fs.festival_nature_id,
fs.location_id

FROM festival_staff fs

RIGHT JOIN staff s

ON fs.staff_id = s.staff_id;



--Get staff id and total festivals staff is involved with in each location

--This query returns only the records of staff except the staff with id 2

SELECT staff_id, COUNT(festival_staff_id)

FROM festival_staff

GROUP BY staff_id

HAVING staff_id != 2;



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

16. Drop_4.txt

```
/*

	DROP FILE

		-Contains commands for dropping all the constraints and tables

		-The drops are ordered such that integrity violation does not occur

		-First the foreign key constraints are dropped, then primary key and check constraints followed by table drops

		 After the table drops, the types and sequences are dropped.


	GROUP 4


		18406547 - Diwas Lamsal

		18406498 - Bhuwan Khatiwada

		18413710 - Sandip Kumar Subba

		18413644 - Bishownath Dhakal

*/

--@D:\Database_Assignment_II\script\drop_4.txt
```

```
--Display dbms output

SET SERVEROUTPUT ON


-------------------------------------------CONSTRAINTS-----------------------------------------------



-------------------------------------------FOREIGN KEYS-----------------------------------------------


ALTER TABLE festival_staff

DROP CONSTRAINT fk_fs_festivals;


ALTER TABLE festivals

DROP CONSTRAINT fk_f_locations;


ALTER TABLE festivals

DROP CONSTRAINT fk_f_festival_natures;


ALTER TABLE festival_staff

DROP CONSTRAINT fk_fs_staff;
```

```
ALTER TABLE staff

DROP CONSTRAINT fk_s_staff;



-------------------------------------------PRIMARY KEYS-----------------------------------------------


ALTER TABLE festival_staff

DROP CONSTRAINT pk_festival_staff;


ALTER TABLE festivals

DROP CONSTRAINT pk_festivals;


ALTER TABLE locations

DROP CONSTRAINT pk_locations;


ALTER TABLE festival_natures

DROP CONSTRAINT pk_festival_natures;


ALTER TABLE staff

DROP CONSTRAINT pk_staff;
```

```
---------------------------------------------CHECK--------------------------------------------------

--Staff table
ALTER TABLE staff
DROP CONSTRAINT ck_staff_gender;

ALTER TABLE staff
DROP CONSTRAINT ck_staff_firstname;

ALTER TABLE staff
DROP CONSTRAINT ck_staff_lastname;


--Festival_natures table
ALTER TABLE festival_natures
DROP CONSTRAINT ck_festival_natures_name;


--Festivals table
```

```
ALTER TABLE festivals

DROP CONSTRAINT ck_festivals_festival_name;



-------------------------------------------TABLES------------------------------------------------


--Drop child tables

DROP TABLE festival_staff;

DROP TABLE festivals;


--Drop parent tables

DROP TABLE locations;

DROP TABLE festival_natures;

DROP TABLE staff;




-------------------------------------------TYPES-----------------------------------------------



DROP TYPE performance_table_type;
DROP TYPE performance_type;


DROP TYPE contact_detail_varray_type;

DROP TYPE contact_detail_type;


DROP TABLE addresses;

DROP TYPE address_type;



------------------------------------------SEQUENCES-----------------------------------------------


--Drop sequences

DROP SEQUENCE seq_staff_id;

DROP SEQUENCE seq_festival_nature_id;

DROP SEQUENCE seq_location_id;

DROP SEQUENCE seq_festival_staff_id;




PURGE RECYCLEBIN;
```

```
-------------------------------------------VIEW------------------------------------------------

--View objects
COLUMN object_name FORMAT A30;
COLUMN object_type FORMAT A12;
SELECT object_name, object_type FROM user_objects
WHERE object_type = 'TYPE';

--View tables
COLUMN tname FORMAT A30;
SELECT * FROM TAB;

--View Sequences
COLUMN sequence_name FORMAT A30;
SELECT sequence_name FROM user_sequences;



--View constraints
COLUMN constraint_name FORMAT A30;
SELECT constraint_name FROM user_constraints
WHERE constraint_name NOT LIKE 'SYS%';
```

17. Drop_test_4.txt

```
/*


        DROP TEST FILE
                -Contains the tests for drops
                -Checks whether drop commands from the drop script file remove all the intended objects

        GROUP 4

                18406547 - Diwas Lamsal
                18406498 - Bhuwan Khatiwada
                18413710 - Sandip Kumar Subba
                18413644 - Bishownath Dhakal

*/
```

```
--@D:\Database_Assignment_II\script\drop_test_4.txt



-------------------------------------------DROPPING CONSTRAINTS-----------------------------------------------



------------------------------------------- DROPPING FOREIGN KEYS-----------------------------------------------


-- TEST

--View constraints

COLUMN constraint_name FORMAT A30;

SELECT constraint_name FROM user_constraints

WHERE constraint_name LIKE 'FK%';


-------------------------------- TEST RESULT-------------------------------


--------- BEFORE DROPPING ALL FOREIGN KEYS------------


--EXPECTED OUTPUT

-- 5 ROWS SELECTED
-- ACTUAL OUTPUT

-- 5 ROWS SELECTED


--------- AFTER DROPPING ALL FOREIGN KEYS------------


--EXPECTED OUTPUT

-- NO ROWS SELECTED


-- ACTUAL OUTPUT

-- NO ROWS SELECTED








-------------------------------------------- DROPPING PRIMARY KEYS-------------------------------------------------


-- TEST

--View constraints

COLUMN constraint_name FORMAT A30;
```

```
SELECT constraint_name FROM user_constraints

WHERE constraint_name LIKE 'PK%';



---------------------------------- TEST RESULT---------------------------------


--------- BEFORE DROPPING ALL PRIMARY KEYS------------


--EXPECTED OUTPUT

-- 5 ROWS SELECTED


-- ACTUAL OUTPUT

-- 5 ROWS SELECTED


--------- AFTER DROPPING ALL PRIMARY KEYS------------


--EXPECTED OUTPUT

-- NO ROWS SELECTED


-- ACTUAL OUTPUT

-- NO ROWS SELECTED
```

```
------------------------------------------- DROPPING CHECK CONSTRAINTS-----------------------------------------------

-- TEST
--View constraints
COLUMN constraint_name FORMAT A30;
SELECT constraint_name FROM user_constraints
WHERE constraint_name LIKE 'CK%';


--------------------------------- TEST RESULT--------------------------------

--------- BEFORE DROPPING ALL CHECK CONSTRAINTS ------------

--EXPECTED OUTPUT
-- 5 ROWS SELECTED
-- ACTUAL OUTPUT

-- 5 ROWS SELECTED


--------- AFTER DROPPING ALL CHECK CONSTRAINTS ------------


--EXPECTED OUTPUT

-- NO ROWS SELECTED


-- ACTUAL OUTPUT

-- NO ROWS SELECTED






------------------------------------------- DROPPING TABLES ------------------------------------------------


-- TEST

--View ALL TABLES

COLUMN tname FORMAT A30;

SELECT * FROM TAB;
```

```
--------------------------------- TEST RESULT--------------------------------

--------- BEFORE DROPPING ALL TABLES ------------

--EXPECTED OUTPUT
-- 7 ROWS SELECTED

-- ACTUAL OUTPUT
-- 7 ROWS SELECTED

--------- AFTER DROPPING ALL TYPES ------------

--EXPECTED OUTPUT
-- NO ROWS SELECTED

-- ACTUAL OUTPUT
-- NO ROWS SELECTED
```

```
-------------------------------------------- DROPPING TYPES -------------------------------------------------

-- TEST
--View Object TYPES
COLUMN object_name FORMAT A30;
COLUMN object_type FORMAT A12;
SELECT object_name, object_type FROM user_objects
WHERE object_type = 'TYPE';



--------------------------------- TEST RESULT--------------------------------

--------- BEFORE DROPPING ALL TYPES ------------

--EXPECTED OUTPUT
-- 5 ROWS SELECTED

-- ACTUAL OUTPUT
-- 5 ROWS SELECTED

--------- AFTER DROPPING ALL TYPES ------------

--EXPECTED OUTPUT
-- NO ROWS SELECTED

-- ACTUAL OUTPUT
-- NO ROWS SELECTED




------------------------------------------- DROPPING SEQUENCE -----------------------------------------------

-- TEST
--View SEQUENCE
COLUMN sequence_name FORMAT A30;
SELECT sequence_name FROM user_sequences;
```

```
---------------------------------- TEST RESULT---------------------------------

--------- BEFORE DROPPING ALL SEQUENCE ------------

--EXPECTED OUTPUT

-- 4 ROWS SELECTED

-- ACTUAL OUTPUT

-- 4 ROWS SELECTED

--------- AFTER DROPPING ALL SEQUENCE ------------

--EXPECTED OUTPUT

-- NO ROWS SELECTED

-- ACTUAL OUTPUT

-- NO ROWS SELECTED
```

18. Other_drop_4.txt

```
/*

        OTHER DROPS FILE

                -Contains the drop commands for functions, procedures, triggers and procedures using cursor


        GROUP 4

                18406547 - Diwas Lamsal

                18406498 - Bhuwan Khatiwada

                18413710 - Sandip Kumar Subba

                18413644 - Bishownath Dhakal


*/



--@D:\Database_Assignment_II\script\other_drops_4.txt


---------------------------------------- FUNCTION DROPS --------------------------------------

DROP FUNCTION func_count_staff;

DROP FUNCTION func_count_staff_salary;

DROP FUNCTION func_count_festivals;
DROP FUNCTION func_exp_location;

DROP FUNCTION func_chp_location;

DROP FUNCTION func_increase_salary;

DROP FUNCTION func_discount_location;

DROP FUNCTION func_username;

DROP FUNCTION func_password;

DROP FUNCTION func_check_string_num;



--------------------------------------- PROCEDURE DROPS ---------------------------------------


DROP PROCEDURE proc_insert_addresses;

DROP PROCEDURE proc_insert_festival_natures;

DROP PROCEDURE proc_insert_locations;

DROP PROCEDURE proc_insert_staff;

DROP PROCEDURE proc_insert_festivals;

DROP PROCEDURE proc_insert_festival_staff;

DROP PROCEDURE proc_reset_seq;

DROP PROCEDURE proc_count_staff;

DROP PROCEDURE proc_count_staff_salary;

DROP PROCEDURE proc_count_festivals;
DROP PROCEDURE proc_exp_location;

DROP PROCEDURE proc_chp_location;

DROP PROCEDURE proc_festival_detail;

DROP PROCEDURE proc_show_staff_address;

DROP PROCEDURE proc_increase_salary;

DROP PROCEDURE proc_discount_location;

DROP PROCEDURE proc_username_password;

DROP PROCEDURE proc_location_capacity_ck;

DROP PROCEDURE proc_staff_firstname;

DROP PROCEDURE proc_staff_surname;



---------------------------------------- TRIGGER DROPS ----------------------------------------


DROP TRIGGER trig_check_address;

DROP TRIGGER trig_staff_email_ck;

DROP TRIGGER trig_staff_name_ck;

DROP TRIGGER trig_del_staff;

DROP TRIGGER trig_ck_location;

DROP TRIGGER trig_festival_natures;

DROP TRIGGER trig_festival_staff;
```

```
-------------------------------------- CURSOR DROPS ---------------------------------------

DROP PROCEDURE proc_num_locations_price;

DROP PROCEDURE proc_del_address_cursor;

DROP PROCEDURE proc_view_addresses_cursor;

DROP PROCEDURE proc_view_festival_natures_cursor;

DROP PROCEDURE proc_view_staff_contact_cursor;

DROP PROCEDURE proc_view_staff_salary_weekly_cursor;
```

19. Log_4.txt

```
/*

        LOG FILE

                -Contains the create and drop commands for log table and triggers

                -This script should be run once the user has been created

                -It allows storing login and logout information


        GROUP 4
```

```
            18406547 - Diwas Lamsal

            18406498 - Bhuwan Khatiwada

            18413710 - Sandip Kumar Subba

            18413644 - Bishownath Dhakal

*/



--Display dbms output
SET SERVEROUTPUT ON

--@D:\Database_Assignment_II\script\log_4.txt

DROP TABLE login_details;

--------------------------------------------------------------------------------------------------------------------
--Table for storing information of logging in and out
CREATE TABLE login_details(
        performed_by VARCHAR2(20),
  event_type VARCHAR2(20),
    event_date DATE,
    event_time VARCHAR2(14)
);



--------------------------------------------------------------------------------------------------------------------------
--Triggers for checking the login and logout
-- RebellionRider (2019).
-- Matthias Hoys (2012).
--Trigger for recording logs of logging in to the system
CREATE OR REPLACE TRIGGER trig_record_login
AFTER LOGON ON SCHEMA
DECLARE
        vc_message VARCHAR2(30);
        vn_hour NUMBER(2);
BEGIN

----------------------------------------------------------------------------------------
--Not able to implement because SET SERVEROUTPUT ON cannot be called from within a trigger
----------------------------------------------------------------------------------------
  SELECT TO_CHAR(SYSDATE, 'hh24') INTO vn_hour FROM DUAL;
```

```
IF vn_hour>= 5 AND vn_hour <12 THEN
      vc_message := 'GOOD MORNING!';
ELSIF vn_hour >= 12 AND vn_hour < 17 THEN
      vc_message := 'GOOD AFTERNOON!';
ELSIF vn_hour >= 17 AND vn_hour < 21 THEN
      vc_message := 'GOOD EVENING!';
ELSE
      vc_message := 'IT IS LATE ALREADY! YOU SHOULD GET SOME SLEEP';
END IF;

DBMS_OUTPUT.PUT_LINE('-----------------------------');
DBMS_OUTPUT.PUT_LINE('HELLO THERE '||USER);
DBMS_OUTPUT.PUT_LINE(vc_message);
DBMS_OUTPUT.PUT_LINE('-----------------------------');
-----------------------------------------------------------------------------------------



INSERT INTO login_details VALUES(
  USER, ora_sysevent, SYSDATE, TO_CHAR(SYSDATE, 'hh24:mi:ss')
);
COMMIT;
```

```
END trig_record_login;
/
SHOW ERRORS
------------------------------------------------------------------------------------------------------------------------
--Trigger for recording logs of logging out of the system
CREATE OR REPLACE TRIGGER trig_record_logoff
BEFORE LOGOFF ON SCHEMA
BEGIN
  INSERT INTO login_details VALUES(
    USER, ora_sysevent, SYSDATE, TO_CHAR(SYSDATE, 'hh24:mi:ss')
  );
  COMMIT;
END trig_record_logoff;
/
SHOW ERRORS

------------------------------------------------------------------------------------------------------------------------
--DROP TRIGGER trig_record_login;
--DROP TRIGGER trig_record_logoff;
------------------------------------------------------------------------------------------------------------------------
```

20. Reset_4.txt

```
/*

        Automates the reset process by dropping everything, recreating all the tables, types, sequences, functions,
        procedures, triggers and cursors

        GROUP 4

                18406547 - Diwas Lamsal
                18406498 - Bhuwan Khatiwada
                18413710 - Sandip Kumar Subba
                18413644 - Bishownath Dhakal

*/

--@D:\Database_Assignment_II\script\reset_4.txt
```

```
-------------------------------------------OTHER DROPS-------------------------------------------------------------
@D:\Database_Assignment_II\script\other_drops_4.txt
----------------------------------------------DROP-----------------------------------------------------------------
@D:\Database_Assignment_II\script\drop_4.txt
---------------------------------------------INSERT----------------------------------------------------------------
@D:\Database_Assignment_II\script\create_4.txt
-------------------------------------------CONSTRAINT--------------------------------------------------------------
@D:\Database_Assignment_II\script\constraint_4.txt
-------------------------------------------FUNCTIONS---------------------------------------------------------------
@D:\Database_Assignment_II\script\function_4.txt
-------------------------------------------PROCEDURE---------------------------------------------------------------
@D:\Database_Assignment_II\script\procedure_4.txt
--------------------------------------------TRIGGER----------------------------------------------------------------
@D:\Database_Assignment_II\script\trigger_4.txt
--------------------------------------------CURSOR-----------------------------------------------------------------
@D:\Database_Assignment_II\script\cursor_4.txt
------------------------------------------------------------------------------------------------------------------

SET LINESIZE 120
```