*** **EXPERIMENT NO: 05** ***

Author: Atharva Paliwal

Roll No: 40 [5B]

Date: 05-November-2020

AIM: To write and execute PL/SQL blocks (with exception handling) including PL/SQL subprograms using Oracle 11g.

PROBLEM STATEMENT:

Establish the database relation EMPLOYEE and populate it with sample records. The logical schema of EMPLOYEE table is-

EMPLOYEE (EID, FNAME, LNAME, BIRTHDATE, GENDER , SSN, HIREDATE, SALARY, DEPARTMENT, DESIGNATION)

Relation : EMPLOYEE

Attribute Name	**Data Description**	**Remarks**
EID	NUMBER (4)	EI-Column,Starts 7101
FNAME	VARCHAR(10)	Required
LNAME	VARCHAR(10)	Required
BIRTHDATE	DATE	Required
GENDER	CHAR(1)	Required,[M,F]
HIREDATE	DATE	Required
SALARY	NUMBER (4)	Required, minimum value 10000
DEPARTNAME	CHAR(20)	
DESIGNATION	VARCHAR(15)	Required
SSN	CHAR(10)	Required, UNIQUE

/* Ensure that you are logged in as a user "CS5xx" and not as SYSTEM or SYS or SYSDBA user. Create table named EXAM with attributes UROLL, COURSE, EXAMDT representing university roll number - an integer ranging between 1001 thru 1099, course as “DBMS” and exam date for the record – 5 days from current date. Enforce entity integrity on UROLL. Test for creation of table and various constraints on it. Before you execute any PL/SQL block, you must enable the PL/SQL output using the command: SET SERVEROUTPUT ON */

```
CREATE TABLE EXAM(
UROLL NUMBER(4) NOT NULL,
COURSE VARCHAR2(4) NOT NULL,
EXAMDT DATE NOT NULL,
CONSTRAINTS EXAM_PK_UROLL PRIMARY KEY(UROLL),
CONSTRAINT EXAM_CK_UROLL CHECK (UROLL >= 1001 AND UROLL <=1099)
);

Table created.

SELECT TABLE_NAME,CONSTRAINT_NAME,CONSTRAINT_TYPE,OWNER
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN ('EXAM');

TABLE_NAME CONSTRAINT_NAME                C OWNER
---------- ------------------------------ - ------------------------------
EXAM       SYS_C0011778                   C CS540
EXAM       SYS_C0011779                   C CS540
EXAM       SYS_C0011780                   C CS540
EXAM       EXAM_CK_UROLL                  C CS540
EXAM       EXAM_PK_UROLL                  P CS540

5 rows selected.

SET SERVEROUTPUT ON

********************************************************************************
QUERY 01: Write a SQL code to write and execute an anonymous PL/SQL block
that will insert 5 tuples into EXAM. Ensure to commit the populated records.
Test the insertion in EXAM by displaying its contents.

/*
Create a table EMPP (contains no records at creation) that includes EID,
ENAME (column combining FNAME and LNAME with embedded blank), HIREDATE,
DESIGNATION and SALARY from EMPLOYEE table. Enforce entity integrity
constraints on EID. Verify table creation, contents and constraints.
*/

********************************************************************************
```

```
BEGIN
 INSERT INTO EXAM VALUES (1001,'DBMS',SYSDATE+5);
 INSERT INTO EXAM VALUES (1002,'DBMS',SYSDATE+5);
 INSERT INTO EXAM VALUES (1003,'DBMS',SYSDATE+5);
 INSERT INTO EXAM VALUES (1004,'DBMS',SYSDATE+5);
 INSERT INTO EXAM VALUES (1005,'DBMS',SYSDATE+5);
 COMMIT;
END;
/
```

PL/SQL procedure successfully completed.

```
SELECT * FROM EXAM;
```

```
     UROLL COUR EXAMDT
---------- ---- ---------
      1001 DBMS 31-OCT-20
      1002 DBMS 31-OCT-20
      1003 DBMS 31-OCT-20
      1004 DBMS 31-OCT-20
      1005 DBMS 31-OCT-20
```

5 rows selected.

Creating table EMPP

```
CREATE TABLE EMPP
AS (SELECT ENO AS EID,FNAME||' '||LNAME AS ENAME,HIREDATE,DESIGNATION,SALARY
 FROM EMPLOYEE
 WHERE 1=2);
```

Table created.

```
SELECT * FROM EMPP;
no rows selected.


ALTER TABLE EMPP
ADD CONSTRAINT EMPP_PK_EID PRIMARY KEY (EID);

Table altered.

SELECT TABLE_NAME,CONSTRAINT_NAME,CONSTRAINT_TYPE,OWNER
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN ('EMPP');

TABLE_NAME CONSTRAINT_NAME               C OWNER
---------- ------------------------------ - ------------------------------
EMPP       SYS_C0011783                  C CS540
EMPP       SYS_C0011784                  C CS540
EMPP       SYS_C0011785                  C CS540
EMPP       SYS_C0011786                  C CS540
EMPP       EMPP_PK_EID                   P CS540

5 rows selected.
```

****

QUERY 02: Write a SQL code to write and execute an anonymous PL/SQL block that will use %TYPE variables to populate the EMPP table with corresponding tuples in EMPLOYEE table.

/*

Create a table MENTEE (contains no records at creation) that includes Staff number, Staff name, Student name (column combining FNAME AND LNAME WITH EMBEDDED BLANK),Roll number and registration date from STUDENT AND STAFF tables. Enforce entity integrity constraints on combination of staff number and roll number. Verify table creation, contents

*/

****

```
DECLARE
 EID EMPLOYEE.ENO%TYPE;
 ENAME EMPLOYEE.FNAME%TYPE;
 HIREDATE EMPLOYEE.HIREDATE%TYPE;
 DESIGNATION EMPLOYEE.DESIGNATION%TYPE;
 SALARY EMPLOYEE.SALARY%TYPE;
BEGIN
 INSERT INTO EMPP (EID, ENAME, HIREDATE, DESIGNATION, SALARY)
 SELECT ENO, FNAME||' '||LNAME, HIREDATE, DESIGNATION, SALARY
 FROM EMPLOYEE;
END;
/
```

PL/SQL procedure successfully completed.

SELECT * FROM EMPP;

EID	ENAME	HIREDATE	DESIGNATION	SALARY
7102	Samantha Jones	08-NOV-06	Professor	146500.0
7101	Eugene Sabatini	10-OCT-06	Professor	150000.0
7103	Alexander Lloyd	01-FEB-07	Professor	148000.0
7104	Simon Downing	01-SEP-07	Professor	138400.0
7107	Christov Plutnik	01-SEP-08	Asso. Professor	127400.0
7105	Christina Mulboro	15-JUL-08	Asso. Professor	127400.0
7106	Dolly Silverline	17-AUG-08	Asso. Professor	127400.0
7108	Ellena Sanchez	12-NOV-09	Asso. Professor	119700.0
7109	Martina Jacobson	15-NOV-09	Asst. Professor	91000.0
7110	William Smithfield	23-JUN-10	Asst. Professor	86400.0
7111	Albert Greenfield	12-JUL-16	Research Asst.	48200.0
7112	James Washington	22-AUG-17	Research Asst.	44600.0
7113	Julia Martin	01-DEC-18	Teaching Asst.	35600.0
7114	Larry Gomes	18-MAY-19	Teaching Asst.	32850.0
7115	Svetlana Sanders	15-JAN-20	Teaching Asst.	30000.0
7116	Lovelyn Brendon	17-JUL-20	Teaching Asst.	30000.0
7117	Hector Hercules	01-AUG-20	Teaching Asst.	32200.0

17 rows selected.

Creating table MENTEE

```
CREATE TABLE MENTEE
AS (SELECT SID AS STAFF_NO, NAME AS STAFF_NAME,FNAME||' '||LNAME AS STUDENT_NAME,
ROLL,REG_DT
 FROM STUDENT,STAFF
 WHERE 1=2);

Table created.

SELECT * FROM MENTEE;
no rows selected

ALTER TABLE MENTEE
ADD CONSTRAINT MENTEE_PK_STAFF_NO_ROLL PRIMARY KEY(STAFF_NO,ROLL);

Table altered.

SELECT TABLE_NAME,CONSTRAINT_NAME,CONSTRAINT_TYPE,OWNER
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN ('MENTEE');

TABLE_NAME CONSTRAINT_NAME                C OWNER
---------- ------------------------------ - ------------------------------
MENTEE     SYS_C0011788                   C CS540
MENTEE     SYS_C0011789                   C CS540
MENTEE     SYS_C0011790                   C CS540
MENTEE     MENTEE_PK_STAFF_NO_ROLL        P CS540

4 rows selected.
```

****

QUERY 03: : Write a SQL code to write and execute an anonymous PL/SQL block that will use %ROWTYPE variables to populate the MENTEE table with corresponding tuples from Academic Schema.

****

```
DECLARE
 CURSOR TOTAL IS
 SELECT SID AS STAFF_NO, NAME AS STAFF_NAME, FNAME||' '||LNAME AS
STUDENT_NAME,
 ROLL, REG_DT
 FROM STAFF INNER JOIN STUDENT ON STUDENT.ADVISOR=STAFF.SID;
 MENTEE TOTAL%ROWTYPE ;
BEGIN
 INSERT INTO MENTEE (STAFF_NO,STAFF_NAME,STUDENT_NAME,ROLL,REG_DT)
 SELECT SID,NAME,FNAME||' '||LNAME,ROLL,REG_DT
 FROM STAFF INNER JOIN STUDENT ON STUDENT.ADVISOR=STAFF.SID;
END;
/
PL/SQL procedure successfully completed.

SELECT * FROM MENTEE;
STAFF_NO STAFF_NAME           STUDENT_NAME               ROLL REG_DT
---------- -------------------- -------------------- ---------- ---------
       101 Kamalkant Marathe    Afra Sayed                    1 20-JUL-18
       101 Kamalkant Marathe    Ritul Deshmukh               11 18-JUL-18
         .
         .
         .
       105 Geetika Goenka       Yogesh Siral                 85 21-JUL-18
       107 Sanjeev Bamireddy    Atharva Paliwal              40 20-JUL-18

         . .
```

```
         .
         .

      110 Harmeet Khullar      Love Sharnagat              68 25-JUL-17
      110 Harmeet Khullar      Tushar Tipnis               89 14-AUG-19

75 rows selected.
```

****

QUERY-04 : Write a SQL code to write and execute an anonymous PL/SQL block that will display the contents of MENTEE table without using declared variables. You should format the output using RPAD() and/or LPAD(), while including proper headers in the result.

****

```
BEGIN
 DBMS_OUTPUT.PUT_LINE('STAFF_NO STAFF_NAME
 STUDENT_NAME ROLL REG_DT');
 DBMS_OUTPUT.PUT_LINE('--- -------------------- -------------------- -- --
--------');
 FOR RECORD IN(SELECT * FROM MENTEE) LOOP
 DBMS_OUTPUT.PUT_LINE(LPAD(RECORD.STAFF_NO, 5,' ')||' '||
 RPAD(RECORD.STAFF_NAME,25,' ')||' '|| RPAD(RECORD.STUDENT_NAME,20,' ')||'
'||
 LPAD(RECORD.ROLL,10,' ')||' '||LPAD(RECORD.REG_DT,9,' '));
 END LOOP;
 EXCEPTION
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
END;
/
```

STAFF_NO	STAFF_NAME	STUDENT_NAME	ROLL	REG_DT
101	Kamalkant Marathe	Afra Sayed	1	20-JUL-18
101	Kamalkant Marathe	Ritul Deshmukh	11	18-JUL-18
101	Kamalkant Marathe	Aayush Muley	31	19-JUL-18
101	Kamalkant Marathe	Ayush Gupta	41	12-JUL-18
101	Kamalkant Marathe	Nikhil Tiwari	56	04-JUL-18
101	Kamalkant Marathe	Rohit Chandani	65	08-AUG-18
102	Adishesh Vidyarthi	Ketki Fadnavis	5	14-JUL-18
102	Adishesh Vidyarthi	Simran Baheti	15	20-JUL-18
102	Adishesh Vidyarthi	Akshat Chandak	35	20-JUL-18
102	Adishesh Vidyarthi	Saurabh Khandagale	46	10-AUG-19
102	Adishesh Vidyarthi	Paritosh Dandekar	57	14-JUL-18
102	Adishesh Vidyarthi	Sankalp Pandey	72	07-JUL-18
102	Adishesh Vidyarthi	Yash Daware	81	20-JUL-18
103	Manishi Singh	Muskan Gupta	7	19-JUL-18
103	Manishi Singh	Prachi Bhanuse	18	11-AUG-19
103	Manishi Singh	Amit Ray	37	20-JUL-18
103	Manishi Singh	Manishkumar Pardhi	48	23-AUG-19
103	Manishi Singh	Rahul Agrawal	59	16-JUL-18
103	Manishi Singh	Saurabh Sushir	73	07-JUL-18
103	Manishi Singh	Yash Jain	84	03-JUL-18
103	Manishi Singh	Anujesh Soni	67	25-JUL-17
104	Aasawari Deodhar	Akansha Wasalu	2	20-JUL-18
104	Aasawari Deodhar	Sakshi Nema	12	07-JUL-18
104	Aasawari Deodhar	Abhishek Chohan	32	07-JUL-18
104	Aasawari Deodhar	Chaitanya Kapre	42	25-JUL-18
104	Aasawari Deodhar	Mehul Khandhadiya	55	19-JUL-18
104	Aasawari Deodhar	Rishikesh Kale	63	07-JUL-18
104	Aasawari Deodhar	Yash Bhageriya	80	19-JUL-18
105	Geetika Goenka	Priyal Taori	9	19-JUL-18
105	Geetika Goenka	Atharva Uplanchiwar	39	07-JUL-18
105	Geetika Goenka	Harsh Karwa	51	11-JUL-18
105	Geetika Goenka	Raunak Khandelwal	62	19-JUL-18
105	Geetika Goenka	Shashank Tapas	75	07-JUL-18
105	Geetika Goenka	Shreyas Nemani	77	20-JUL-18
105	Geetika Goenka	Yogesh Siral	85	21-JUL-18
106	Deo Narayan Mishra	Prateeksha Devikar	8	13-JUL-18

106	Deo Narayan Mishra	Deepali Pathe	17	10-AUG-19
106	Deo Narayan Mishra	Aryan Pandharipande	38	07-JUL-18
106	Deo Narayan Mishra	Ganesh Thakur	47	22-AUG-19
106	Deo Narayan Mishra	Rajat Chandak	60	20-JUL-18
106	Deo Narayan Mishra	Shardul Nimbalkar	74	28-JUL-17
106	Deo Narayan Mishra	Yash Dhamecha	83	21-JUL-18
107	Sanjeev Bamireddy	Rashi Chouksey	10	08-AUG-18
107	Sanjeev Bamireddy	Siddhi Tripathi	19	31-AUG-19
107	Sanjeev Bamireddy	Atharva Paliwal	40	20-JUL-18
107	Sanjeev Bamireddy	Jayesh Kapse	52	08-AUG-18
107	Sanjeev Bamireddy	Ram Agrawal	61	19-JUL-18
107	Sanjeev Bamireddy	Shivam Bagadia	76	20-JUL-18
107	Sanjeev Bamireddy	Shapath Pandey	86	27-JUL-17
107	Sanjeev Bamireddy	Ayush Singh	66	27-JUL-17
108	Jasmine Arora	Anjali Rajendran	3	19-JUL-18
108	Jasmine Arora	Shreya Agnihotri	13	07-JUL-18
108	Jasmine Arora	Adesh Kotgirwar	33	20-JUL-18
108	Jasmine Arora	Dev Paliwal	43	21-JUL-18
108	Jasmine Arora	Kunal Thorane	54	08-AUG-18
108	Jasmine Arora	Ritik Parashar	64	19-JUL-18
108	Jasmine Arora	Yaman Kushwah	79	17-JUL-18
108	Jasmine Arora	Mayank Rangari	87	25-JUL-16
109	Vallabh Pai	Aradhita Menghal	4	07-JUL-18
109	Vallabh Pai	Shrishti Shukla	14	19-JUL-18
109	Vallabh Pai	Adhney Nawghare	34	08-AUG-18
109	Vallabh Pai	Gaurav Shukla	44	17-JUL-18
109	Vallabh Pai	Keshubh Sharma	53	20-JUL-18
109	Vallabh Pai	Shubham Jha	78	12-JUL-18
109	Vallabh Pai	Renuka Soni	30	25-JUL-16
109	Vallabh Pai	Naveen Namjoshi	88	14-AUG-19
110	Harmeet Khullar	Lalita Sharma	6	10-JUL-18
110	Harmeet Khullar	Urvi Negi	16	19-JUL-18
110	Harmeet Khullar	Amey Chole	36	08-AUG-18
110	Harmeet Khullar	Gursewak Virdi	45	07-JUL-18
110	Harmeet Khullar	Pavankumar Gupta	58	03-JUL-18
110	Harmeet Khullar	Rushil Parikh	71	07-JUL-18
110	Harmeet Khullar	Yash Roy	82	07-JUL-18
110	Harmeet Khullar	Love Sharnagat	68	25-JUL-17

110 Harmeet Khullar Tushar Tipnis 89 14-AUG-19

PL/SQL procedure successfully completed.

****

QUERY-05 : Write a SQL code to write and execute an anonymous PL/SQL block that will display the system date. Use exception (exception VALUE_ERROR) to check if the variable holding the system date is large enough in size. Re-execute the block with appropriate modification to test the exception.

****

```
DECLARE
 CURR_DATE VARCHAR2(10);
BEGIN
 SELECT SYSDATE INTO CURR_DATE FROM DUAL;
 DBMS_OUTPUT.PUT_LINE('CURRENT DATE : '|| CURR_DATE);
EXCEPTION
 WHEN VALUE_ERROR THEN
 DBMS_OUTPUT.PUT_LINE('Too large value');
END;
/
```

CURRENT DATE : 05-NOV-20

PL/SQL procedure successfully completed.

Re-executing the query to check exception-

```
DECLARE
  CURR_DATE VARCHAR2(5);
BEGIN
  SELECT SYSDATE INTO CURR_DATE FROM DUAL;
  DBMS_OUTPUT.PUT_LINE('CURRENT DATE : '|| CURR_DATE);
EXCEPTION
  WHEN VALUE_ERROR THEN
  DBMS_OUTPUT.PUT_LINE('Exception occurred: Too large value');
END;
/
```

```
Exception occurred: Too large value

PL/SQL procedure successfully completed.
```

****

QUERY-06 : Write a SQL code to create and execute an anonymous PL/SQL block that will check (say, for employee number 7108) whether an employee is entitled to receive the longevity bonus. Longevity bonus is given to employees with minimum 12 years of service. Now, re-execute the block to extend longevity bonus to employees with 10 years of service.

****

```
DECLARE
 EID_INPUT NUMBER := &NUM;
 BONUS_YEAR NUMBER(2);
 HIREDATE_INPUT EMPP.HIREDATE%TYPE;
BEGIN
 SELECT HIREDATE INTO HIREDATE_INPUT FROM EMPP WHERE EID = EID_INPUT;
 BONUS_YEAR := MONTHS_BETWEEN(SYSDATE,HIREDATE_INPUT)/12;
 IF BONUS_YEAR >= 12 THEN
 DBMS_OUTPUT.PUT_LINE('EMPLOYEE '||EID_INPUT||' IS ENTITLED FOR BONUS');
 ELSE
 DBMS_OUTPUT.PUT_LINE('EMPLOYEE '||EID_INPUT||' IS NOT ENTITLED FOR BONUS');
 END IF;
END;
/
```

```
Enter value for num: 7108
old   2:  EID_INPUT NUMBER := &NUM;
new   2:  EID_INPUT NUMBER := 7108;
EMPLOYEE 7108 IS NOT ENTITLED FOR BONUS

PL/SQL procedure successfully completed.
```

Re-executing for employees with minimum 10 years of service

```
DECLARE
 EID_INPUT NUMBER := &NUM;
 BONUS_YEAR NUMBER(2);
 HIREDATE_INPUT EMPP.HIREDATE%TYPE;
BEGIN
 SELECT HIREDATE INTO HIREDATE_INPUT FROM EMPP WHERE EID = EID_INPUT;
 BONUS_YEAR := MONTHS_BETWEEN(SYSDATE,HIREDATE_INPUT)/12;
 IF BONUS_YEAR >= 10 THEN
 DBMS_OUTPUT.PUT_LINE('EMPLOYEE '||EID_INPUT||' IS ENTITLED FOR BONUS');
 ELSE
 DBMS_OUTPUT.PUT_LINE('EMPLOYEE '||EID_INPUT||' IS NOT ENTITLED FOR BONUS');
 END IF;
END;
/
```

```
Enter value for num: 7108
old   2:  EID_INPUT NUMBER := &NUM;
new   2:  EID_INPUT NUMBER := 7108;
EMPLOYEE 7108 IS ENTITLED FOR BONUS

PL/SQL procedure successfully completed.
```

****

QUERY-07 : Write a SQL code to create and execute an anonymous PL/SQL block that will locate the first August born employee. Re-write and execute an anonymous PL/SQL block that will locate the first August born employee, when EMPLOYEE is searched in reversed order.

****

```
DECLARE
 EMP_REC EMPLOYEE%ROWTYPE;
 i NUMBER := 7101;
BEGIN
 LOOP
 SELECT * INTO EMP_REC FROM EMPLOYEE WHERE ENO = i;
i := i + 1;
 IF EXTRACT(MONTH FROM EMP_REC.BIRTHDATE) = 8 THEN
 DBMS_OUTPUT.PUT_LINE('FIRST AUGUST BORN EMPLOYEE RECORD : '||
 EMP_REC.ENO ||' '||EMP_REC.FNAME||' '||EMP_REC.LNAME);
 EXIT;
 END IF;
 IF i> 7117 THEN
 EXIT;
 END IF;
 END LOOP;
END;
/
```

```
FIRST AUGUST BORN EMPLOYEE RECORD : 7114 Larry Gomes


PL/SQL procedure successfully completed.
```

```
DECLARE
 EMP_REC EMPLOYEE%ROWTYPE;
i NUMBER := 7117;
BEGIN
 LOOP
 SELECT * INTO EMP_REC FROM EMPLOYEE WHERE ENO = i;
i := i - 1;
 IF EXTRACT(MONTH FROM EMP_REC.BIRTHDATE) = 8 THEN
 DBMS_OUTPUT.PUT_LINE('FIRST AUGUST BORN EMPLOYEE RECORD : '||
 EMP_REC.ENO ||' '||EMP_REC.FNAME||' '||EMP_REC.LNAME);
 EXIT;
 END IF;
 IF i< 7101 THEN
 EXIT;
 END IF;
 END LOOP;
END;
/
```

```
FIRST AUGUST BORN EMPLOYEE RECORD : 7114 Larry Gomes


PL/SQL procedure successfully completed.
```

****

QUERY-08 : Write a SQL code to create and execute an anonymous PL/SQL block that accept Staff ID from the console and will display staff details for said staff. A system exception, NO_DATA_FOUND should be cached when the mentioned staff does not exist.

```
/*
Create table PAYSCALE, that includes fields- DESIGNATION(15 alphanumeric characters),MINPAY(6 digits), MAXPAY(6 digits).Entity Integrity is maintained
```

on DESIGNATION, with plausible values – Professor, Research Asst., Asso. Professor, Teaching Asst. and Asst. Professor.
Add following tuples to PAYSCALE table-
  Professor,140000,200000
  Asso. Professor,100000,140000
  Asst. Professor,50000,90000
  Teaching Asst.,20000,32500
  Research Asst.,30000,45000
*/

****

```
DECLARE
 STAFF_RECORD STAFF%ROWTYPE;
 STAFF_INPUT NUMBER := &NUM;
BEGIN
 SELECT * INTO STAFF_RECORD FROM STAFF WHERE SID = STAFF_INPUT;
 DBMS_OUTPUT.PUT_LINE(STAFF_RECORD.SID||' '|| STAFF_RECORD.NAME||' '||
 STAFF_RECORD.BRANCH||' '||STAFF_RECORD.DESG||' '||
 STAFF_RECORD.JOIN_DT);
EXCEPTION
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('STAFF DOES NOT EXIST.');
END;
/
```

```
Enter value for num: 107
old   3:  STAFF_INPUT NUMBER := &NUM;
new   3:  STAFF_INPUT NUMBER := 107;
107 Sanjeev Bamireddy CSEC Associate 12-MAY-18


PL/SQL procedure successfully completed.
```

```
SQL> /
Enter value for num: 120
old   3: STAFF_INPUT NUMBER := &NUM;
new   3: STAFF_INPUT NUMBER := 120;
STAFF DOES NOT EXIST.

PL/SQL procedure successfully completed.
```

Creating table PAYSCALE

```
CREATE TABLE PAYSCALE(
 DESIGNATION VARCHAR2(15) NOT NULL,
 MINPAY NUMBER(6) NOT NULL,
 MAXPAY NUMBER(6) NOT NULL,
 CONSTRAINT PAYSCALE_PK_DESIGNATION PRIMARY KEY(DESIGNATION),
 CONSTRAINTS PAYSCALE_CK_DESIGNATION CHECK
 (DESIGNATION IN('Professor','Research Asst.','Asso. Professor','Teaching
Asst.','Asst. Professor'))
 );
```

Table created.

```
BEGIN
 INSERT INTO PAYSCALE VALUES ('Professor',140000,200000);
 INSERT INTO PAYSCALE VALUES ('Asso. Professor',100000,140000);
 INSERT INTO PAYSCALE VALUES ('Asst. Professor',50000,90000);
 INSERT INTO PAYSCALE VALUES ('Teaching Asst.',20000,32500);
 INSERT INTO PAYSCALE VALUES ('Research Asst.',30000,45000);
END;
/
```

PL/SQL procedure successfully completed.

```
SELECT * FROM PAYSCALE;
```

```
DESIGNATION         MINPAY     MAXPAY
--------------- ---------- ----------
```

```
Professor           140000     200000

Asso. Professor     100000     140000

Asst. Professor      50000      90000

Teaching Asst.       20000      32500

Research Asst.       30000      45000


5 rows selected.
```

****

QUERY-09 : Write a SQL code to create and execute an anonymous PL/SQL block that defines user-defined exceptions- BELOW_PAY_RANGE and ABOVE_PAY_RANGE. Your script should accept an employee number from the console and check for the salary to fall within the payscale [minpay,maxplay].

If salary is less than minpay, BELOW_PAY_RANGE exception is raised and when cached an appropriate message-
'<EmpNo> Receives Salary Below Scale [minpay,maxplay].'
is displayed; otherwise ABOVE_PAY_RANGE exception is raised and cached to display appropriate message accordingly.

You must appropriately catch the NO_DATA_FOUND exception also. When there are no violations, display for the employee the salary drawn. Test the above anonymous block for input employee numbers –
7101, 7104, 7106, 7109, 7111, 7114 and 7117.

****

```
DECLARE
 EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;
 MINPAY_INPUT PAYSCALE.MINPAY%TYPE;
 MAXPAY_INPUT PAYSCALE.MAXPAY%TYPE;
 SAL EMPLOYEE.SALARY%TYPE;
 ABOVE_PAY_RANGE EXCEPTION;
 BELOW_PAY_RANGE EXCEPTION;
BEGIN
 SELECT EMPLOYEE.SALARY,PAYSCALE.MINPAY,PAYSCALE.MAXPAY
```

```
 INTO SAL,MINPAY_INPUT,MAXPAY_INPUT
 FROM EMPLOYEE INNER JOIN PAYSCALE USING (DESIGNATION)
 WHERE EMPLOYEE.ENO= EMP_INPUT;
 IF SAL > MAXPAY_INPUT THEN
 RAISE ABOVE_PAY_RANGE;
 ELSIF SAL < MINPAY_INPUT THEN
 RAISE BELOW_PAY_RANGE;
 ELSE
 DBMS_OUTPUT.PUT_LINE('EMPLOYEE NUMBER IS '||EMP_INPUT||
 ' AND EMPLOYEE SALARY IS '|| SAL);
 END IF;
EXCEPTION
 WHEN ABOVE_PAY_RANGE THEN
 DBMS_OUTPUT.PUT_LINE(EMP_INPUT||' Receives Salary Above Scale '||'['||
 MINPAY_INPUT||','||MAXPAY_INPUT||']');
 WHEN BELOW_PAY_RANGE THEN
 DBMS_OUTPUT.PUT_LINE(EMP_INPUT||' Receives Salary Below Scale '||'['||
 MINPAY_INPUT||','||MAXPAY_INPUT||']');
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
END;
/

Enter value for input: 7101
old   2:  EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;
new   2:  EMP_INPUT EMPLOYEE.ENO%TYPE := 7101;
EMPLOYEE NUMBER IS 7101 AND EMPLOYEE SALARY IS 150000

PL/SQL procedure successfully completed.
```

SQL> /

Enter value for input: 7104

old   2: EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;

new   2: EMP_INPUT EMPLOYEE.ENO%TYPE := 7104;

7104 Receives Salary Below Scale [140000,200000]

PL/SQL procedure successfully completed.

SQL> /

Enter value for input: 7106

old   2: EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;

new   2: EMP_INPUT EMPLOYEE.ENO%TYPE := 7106;

EMPLOYEE NUMBER IS 7106 AND EMPLOYEE SALARY IS 127400

PL/SQL procedure successfully completed.

SQL> /

Enter value for input: 7109

old   2: EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;

new   2: EMP_INPUT EMPLOYEE.ENO%TYPE := 7109;

7109 Receives Salary Above Scale [50000,90000]

PL/SQL procedure successfully completed.

SQL> /

Enter value for input: 7111

old   2: EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;

new   2: EMP_INPUT EMPLOYEE.ENO%TYPE := 7111;

7111 Receives Salary Above Scale [30000,45000]

```
PL/SQL procedure successfully completed.

SQL> /
Enter value for input: 7114
old   2:  EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;
new   2:  EMP_INPUT EMPLOYEE.ENO%TYPE := 7114;
7114 Receives Salary Above Scale [20000,32500]

PL/SQL procedure successfully completed.

SQL> /
Enter value for input: 7117
old   2:  EMP_INPUT EMPLOYEE.ENO%TYPE := &INPUT;
new   2:  EMP_INPUT EMPLOYEE.ENO%TYPE := 7117;
EMPLOYEE NUMBER IS 7117 AND EMPLOYEE SALARY IS 32200

PL/SQL procedure successfully completed.
```

****

QUERY-10 : Write a SQL code to create and execute an anonymous PL/SQL block that will modify Query-09 to process all records of EMPLOYEE table. You need not acquire employee number from console. You should only report the violations.

****

```
DECLARE
 EMP_IN EMPLOYEE.ENO%TYPE;
 DESG_IN EMPLOYEE.DESIGNATION%TYPE;
 SAL EMPLOYEE.SALARY%TYPE;
 MINP PAYSCALE.MINPAY%TYPE;
```

```
 MAXP PAYSCALE.MAXPAY%TYPE;
 BELOW_PAY_RANGE EXCEPTION;
 ABOVE_PAY_RANGE EXCEPTION;
BEGIN
 FOR i IN (SELECT ENO AS EMP_IN, DESIGNATION AS DESG_IN, SALARY AS SAL,
 MINPAY AS MINP, MAXPAY AS MAXP FROM EMPLOYEE NATURAL JOIN PAYSCALE)
 LOOP
SAL:=i.SAL;
 DESG_IN:=i.DESG_IN;
MINP:=i.MINP;
MAXP:=i.MAXP;
 EMP_IN:=i.EMP_IN;
 BEGIN
 IF SAL > MAXP THEN
 RAISE ABOVE_PAY_RANGE;
 ELSIF SAL < MINP THEN
 RAISE BELOW_PAY_RANGE;
 ELSE
 DBMS_OUTPUT.PUT_LINE('');
 END IF;
 EXCEPTION
 WHEN BELOW_PAY_RANGE THEN
 DBMS_OUTPUT.PUT_LINE(EMP_IN||' Receives Salary Below Scale '||
 '['||MINP||','||MAXP||']');
 WHEN ABOVE_PAY_RANGE THEN
 DBMS_OUTPUT.PUT_LINE(EMP_IN||' Receives Salary Above Scale '||
 '['||MINP||','||MAXP||']');
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
 END;
```

```
 END LOOP;
END;
```

```
7104 Receives Salary Below Scale [140000,200000]
7109 Receives Salary Above Scale [50000,90000]
7111 Receives Salary Above Scale [30000,45000]
7113 Receives Salary Above Scale [20000,32500]
7114 Receives Salary Above Scale [20000,32500]

PL/SQL procedure successfully completed.
```

****

VIVA VOICE

****

Question-01: What is an anonymous block?

****

Answer : The PL/SQL anonymous block statement is an executable statement that can contain PL/SQL control statements and SQL statements. It can be used to implement procedural logic in a scripting language. The optional exception section can be inserted near the end of the BEGIN-END block.

Question-02: What is an exception? List the standard PL/SQL exceptions.

****

Answer : An exception is a PL/SQL error that is raised during program execution, either implicitly by TimesTen or explicitly by your program. Handle an **exception** by trapping it with a handler or propagating it to the calling environment.

Standard PL/SQL exceptions are-

ACCESS_INTO_NULL,CASE_NOT_FOUND,COLLECTION_IS_NULL,DUP_VAL_ON_INDEX,INVALID_CURSOR,INVALID_NUMBER,LOGIN_DENIED,NO_DATA_FOUND,NOT_LOGGED_ON,PROGRAM_ERROR,ROWTYPE_MISMATCH,SELF_IS_NULL,STORAGE_ERROR,TOO_MANY_ROWS,VALUE_ERROR,ZERO_DIVIDE

Question-03: Differentiate between ‘&’ and ‘&&’ in SQL.

****

Answer : & - is used to create a temporary substitution variable that will prompt you for a value every time it is reffered. && - is used to create a permanent substitution variable.

Question-04: Why it is good practice to use %TYPE when declaring variables?

****

Answer : The %TYPE attribute lets you declare a constant, variable, field, or parameter to be of the same data type a previously declared variable, field, record, nested table, or database column. If the referenced item changes, your declaration is automatically updated.

Use of this attribute ensures that type compatibility between table columns and PL/SQL variables is maintained.

Inference :

- All the queries were executed successfully.
- PL/SQL concepts were learnt and queries were executed.
- User-defined and standard exceptions were also handled.

********** **END** **********